Institut für Informatik
Lehrstuhl für Programmierung und Softwaretechnik

LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Minecraft in MicroPsi

Eine populäre Videospielwelt als Simulationsumgebung für eine kognitive künstliche Intelligenz

Jonas Kemper

Medieninformatik Bachelor

Aufgabensteller: Prof. Dr. Martin Wirsing
Betreuer: Joscha Bach PhD, Annabelle Klarl
Abgabetermin: 19. September 2013

Ich versichere hiermit eidesstattlich, dass ich die vorliegende Arbeit selbstständig angefertigt, alle Zitate als solche kenntlich gemacht sowie alle benutzten Quellen und Hilfsmittel angegeben habe.

München, den 19. September 2013

. . . .

(Unterschrift des Kandidaten)

Zusammenfassung

Ziel der Arbeit ist die Entwicklung und das Testen einer Simulationsumgebung für einen kognitiven Agenten auf Basis des populären Videospiels Minecraft.

Minecraft bietet sich als Grundlage für eine Simulationsumgebung aufgrund der kompositionalen Semantik der Spielwelt besonders an, da das Agentensystem durch das Erforschen seiner Umwelt Wissen über die Spielwelt aufbauen und ähnliche Strukturen wieder erkennen kann. Objekte der Spielwelt sind in Minecraft keine bloßen Hindernisse sondern werden mit variablen Eigenschaften prozedural erzeugt und ähneln so eher einer realen Umgebung als andere virtuelle Welten.

Da Minecraft von Anfang an mit einem Mehrspielermodus ausgestattet wurde, lässt es sich zudem für Multiagenten-Umgebungen und so für kollaborative Agenten verwenden. Des Weiteren sind Minecraft-Lizenzen günstig zu erwerben, erhältlich für viele Plattformen und es gibt eine äußerst große und aktive Community für selbsterstellte Spiel-Inhalte und -Modifikationen.

Angestrebt wird, dass die kognitive Architektur MicroPsi2 sich an einen Minecraft-Server anmelden kann, ihre Umgebung wenigstens in Ansätzen wahrnimmt (Objekte, Terraintypen), sich fortbewegen kann und einfache Interaktionsmöglichkeiten besitzt (z.B. Objekt aufnehmen und ablegen).

Daneben soll eine Visualisierung der Umgebung aus Sicht des Agenten entstehen, z.B. als zweidimensionale Übersicht, in deren Zentrum der Agent steht. Diese Visualisierung soll live im Browser angezeigt werden (wahlweise mit WebGL oder Canvas/D3).

Zum Schluss der Arbeit soll zum Testen der Funktionalität ein virtuelles Braitenbergvehikel in die Simulationsumgebung gesetzt werden, welches sich daraufhin auf die nächstgelegene Lichtquelle zubewegen soll. Dieses Experiment wird als Teil der Arbeit umfangreich dokumentiert.

Abstract

What I cannot create, I do not understand Richard P. Feynman, 1988

Danksagung

Ich danke meinen Betreuern Joscha Bach und Annabelle Klarl sowie Dominik und Professor Wirsing.

Contents

Chapter 1

Introduction

The hunt for artificial intelligence started many years ago. Dividing the subject into the strong and weak part means seperating its goals into usefull applications and those that try to learn about the nature of intelligence itself. The ultimate task of strong A.I. — recreating human intelligence — admitably still seems to be science fiction, though.

But then again: a new generation of cognitive scientists, psychologists and computer scientists strives to implement new Ideas for simulated cognition — building cognitive architectures. Many of them do so by simulating, in one way or the other, what could be called ”neural networks(/nodenets)”.

One of these architectures is MicroPsi 2. Developed by Joscha Bach, it is based on the Psi Theory by Dietrich Dörner, who is a german Professor for theoretical psychology. It aims at providing a solid and complete implementation of that theory of cognition and at beeing easily accesible, understandable and modifiable for the research and applications to come.

To test the functionality of such cognitive architectures and to figure out their capabilities and potential, we need to research their behavior inside defined environemts. As implementing AI into the physical world (as robots for examples) requires building apropriate hardware and patience, computer-simulated environments therefore play an important role.

MicroPsi 2 is both: a cognitive architecture but also a set of and an interface to simulation environments.

1.1 Motivation

Video games are natural applications of artificial intelligence. The quality of a games A.I. can make all the difference in between a great title and an uninspired demo of computer graphics technology.

Building an interface between the cognitive architecture MicroPsi 2 and a video game world is what this Thesis is about.

1.2 Objective

1.3 Approach

1.4 Outcome

1.5 Outline

...

Chapter 2

Background

A widely accepted definition of the field of A.I. is that it is "the study and design of intelligent agents",[1] where an intelligent agent is a system that perceives its environment and takes actions that maximize its chances of success.[2] (TODO find original sources)

2.1 A! Artificial Intelligence / Examples for A.I. applications

It took many years for A.I. research to evolve from the early ideas of thinking machines over Deep Blue, the computer that could beat mankinds best chess players to Watson, the A.I. that beats the champions of Jeopardy, the game show, that is about asking the apropriate question to a given answer. There are many other applications of A.I. out there. Self-driving cars and online-shopping recommendation systems to name a few.

... related work ...

2.1.1 weak AI

The mentioned examples for A.I. have one thing incommon. They are applications of technology that serve an immediate, or at least foreseeable purpose. For A.I. in scenarios of this kind. The term "weak A.I." (or "applied A.I." has been coined.

2.1.2 strong AI

"Strong A.I.", in contrast, is about researching the nature of intelligence and consciousnes themselves. An actual (hypothetical) implementation of a Strong A.I. would mean building a machine, that is capable of acting like a human beeing. Not just in a specialized and defined problem field, but in all of them.

2.2 A! Cognitive AI / cognitive AI

Cognitive A.I. can be thought of as architectures that implement findings and theories in cognitive science and psychology as well as the neuro-sciences for the sake of prooving, if the theories hold against what they promise.

... Examples for cognitive architectures are ...

2.2.1 Concepts

Many cognitive architectures share characteristics with or implement artificial neural nodenets.

... Approaches to cognitive AI ... ... neural node nets ... ... related work ...

2.3 A! Psi Theory

The Psi theory in its foundations was described by german psychologist Dietrich Dörner in his book "Bauplan für eine Seele" and "Die Mechanik des Seelenwagens" from 1998 and 2002. It's main ideas are that it thinks of cognition as a graph-like structure (e.g. node-net) of relationships that strives to maintain homeostatic balance.[Bac09]

Basic components of the theory are Representation, Memory, Perception, Drives, Cognitive modulation and emotion, Motivation, Learning and Problem solving as well as Language and consciousness. (wikipedia)

"The PSI theory is a theory of human action regulation by psychologist Dietrich Doerner [1, 2]. It is an attempt to create a body-mind link for virtual agents. It aims at the integration of cognitive processes, emotions and motivation. This theory is unusual in cognitive psychology in that emotions are not explicitly defined but emerge from modulation of perception, action-selection, planning and memory access. Emotions become apparent when the agents interact with the environment and display expressive behavior, resulting in a configuration that resembles emotional episodes in biological agents. The agents react to the environment by forming memories, expectations and immediate evaluations. This short presentation is a good overview.

PSI agents possess a number of modulators and built-in motivators that lie within a range of intensities. These parameters combined to produce complex behavior that can be interpreted as being emotional. Additionally, by regulating these parameters, the agent is able to adapt itself to the different circumstances in the environment.This theory has been applied to different virtual agent simulations in different types of environments [3, 4, 5, 6, 7, 8] and has proven to be a promising theory for creation of biologically plausible agents." (wikipedia)

... Basics of Psi Theorie of Dörner ...

2.3.1 Joschas Contribution: MicroPsi

Joscha Bach adapted that theory to bring it in a contemporary from with his own slight modifications.

... explanation of Joschas Dissertation ...

2.4 A? Summary

Even though building a conscious machine that thinks and acts like we do is still mehre science-fiction, it is this kind of foundational research, that leads no new ways of thinking of the world, that give us our most import leaps.

... still a lot to do in AI ...

Chapter 3

Foundations

This work is a work of combining existing technologies. In particular, the two most important ones are Micro Psi 2, the most ambitous framework for the Psi theory, and Minecraft, the super-popular sandbox-videogame.

To understand how and why they were chosen, a brief history of their creation as well as explanations of their basic ideos and relevant insights to their architecture are what this chapter is about.

3.1 A! Psi Implementations

Psi has been implemented by different groups at different times.

3.1.1 Dörners Implementation

The first implementations are by Dörner and his associates themselves. The used Pascal and developed it for windows environments. This implementation can still be downloaded and runs on Windows 7 installtions, for example.

... psi screenshot ...

3.1.2 Joschas Implementation

The work on Dörners teams implementation has not been continued, so Joscha Bach and his associates built new implementations of Psi.

3.1.2.1 MicroPsi in Java

From 2003 to 2009 they built an implementation in Java. It consisted of a graphical editor and a 3D simulation-environment. [Bac12]

"The first implementation of the MicroPsi framework spanned the years 2003 to 2009, and was built in Java as a set of plugins for the Eclipse IDE. The graphical editor was built on SWT. It comprised about 60000 lines of code, and although a lot of effort went towards platform independence (with the exception of a DirectX/.Net based 3D viewer component), deployment on the various operating systems and across several versions of Eclipse became support intensive, especially after its adop- tion by teams outside of our group." [Bac12]

3.1.2.2 MicroPsi in Python

Aiming at better understandability and to maintain platform independence, MicroPsi has been built ground up again using Python. What is remarkable about the new implementation called MicroPsi, is that the graphical interface is completely rendered inside a webbrowser — using state-of-the-art internet- and webapplication-technologies. [Bac12]

Even though there have been more complex simulation enviroments (e.g. 3D-worlds) for previous implementations of Psi-architectures, the relatively new MicroPsi 2 has only two fairly simple ones: a 2D-Island and a map of the public transportation system of Berlin. Instead of building a new 3D-world, we set out for something a little more innovative.

"The cognitive architecture MicroPsi builds on a framework for simulating agents as neuro-symbolic spreading activation networks. These agents are situated in a simulation environment or fitted with robotic bodies. The current implementation of MicroPsi has been re-implemented from the ground up and is described here." [Bac12]

"Gradual changes in the formalization of MicroPsi and the emergence of new software development methodologies and tool chains, especially the move from Java design patterns and XML tools towards lightweight and agile Python code, prompted a complete rewrite of the MicroPsi framework, starting in 2011. The following section describes the overall structure of the framework, followed by detailed definitions of the node net formalism and the structure of simulation worlds that enable running MicroPsi agents." [Bac12]

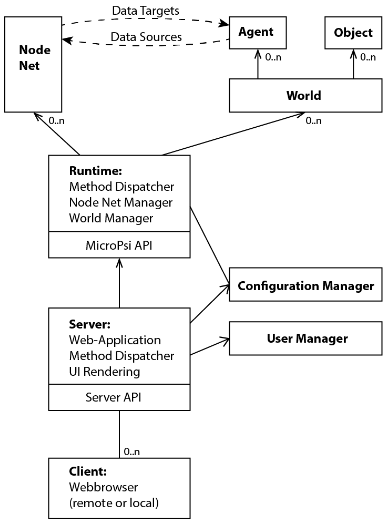
The MicroPsi 2 user interface is rendered completely inside a web browser and the simulation is deployed as a web application. The UI components are based upon HTML/Javascript and "and facilitates the communication between the browser based renderer and the agent simulator via JSON and JSON remote procedure calls. Rendering is supported by Twitter's widget library Bootstrap (2012) and the Javascript library PaperJS (Lehni and Puckey, 2011)." [Bac12]

3.1.2.3 Module Overview / Architecture

MicroPsi modular structure makes it fairly easy to understand. At first, one can diffentiate between the server component (or the web-interface) and the actual simulation code (called "core").

In a minimal setup microPsi runs three threads. One thread for the webserver, one for the world simulation and for every world-adapter (or agent). If more then one agent or more than one world are launched, they are instantiated as additional threads.

"MicroPsi consists of a server (the web application), a runtime component, a set of node nets, a set of simulation worlds, a user manager and a configuration manager (figure 2). The server is built on the micro web framework Bottle (Hellkamp 2011) and communicates with all current users via their web browsers through the Server API. User sessions and access rights are handled by the user manager component. On startup, the server invokes the runtime component, which interfaces to the server with the MicroPsi API. The runtime is designed to work independently of the server and does not need to be deployed as a web application (command line interac- tion or OS based user interfaces are possible as well). The runtime supplies a manager for MicroPsi node nets (see section 4), and a manager for simulation worlds (or interfaces to outside environments, such as robotic bodies, remote data providers, etc.). Standard simulation worlds (section 6) provide agents (node net embodiments) and objects as situated state machines." [Bac12]



Agents MicroPsi defines agents as node nets. [Bac12]

"MicroPsi interprets cognitive models as agents, situated in dynamic environments. MicroPsi agents are entirely defined as hierarchical spreading activation networks (SAN), which—for lack of a better name—are called node nets. Node nets are the brains of these agents—or rather, an abstraction of the information processing provid- ed by brains, and the environment provides a body and stuff to interact with. The body manifests itself as a set of data sources (which can be thought of as the terminals of sensory neurons) and data targets (the abstracted equivalent of motor neurons). By reading activation values from data sources, and sending activation into data targets, the MicroPsi agent may control its body and interact with its world. MicroPsi's node nets can be interpreted as neural networks and afford neural learn- ing paradigms. For the purposes of information storage and retrieval, they can be seen as semantic networks with a small set of typed links to express associative, partonom- ic, taxonomic and causal relationships. Since the nodes can also encapsulate state machines and arbitrary operations over the node net, they can also be understood as components of a concurrent, modularized architecture, with activation spreading as the primary means of communication be- tween the modules." [Bac12]

Environment The simulations worlds are the environments in which we can study our node nets behavior. Worlds need to provide the worldadapter — the interface in between the node net and the simulation. Data sources and data targets have to be defined, to get a functional and meaningful experiment going. [Bac12]

"Within the MicroPsi framework, agents may be embedded into an environment (world). The environment must provide a world adapter wa for each MicroPsi agent. The world adapter offers data sources, from which the agent's node net may read environmental information, and data targets, which allow the agent to effect changes in the world. Since the environment only has write access to data sources, and read access to data targets, node net and environment may be updated asynchronously. The world adapter may interface a local multi-agent simulation, a robotic body, a computer game client or simulation server, dynamically updated stock data, etc." [Bac12]

Applications At the time of the development of the original development of the framework, the priorized application was building a framework for knowledge representation. [Bac12]

"Compared with the original implementation of MicroPsi, the current iteration of the framework is still fragmentary; at the time of writing, it supports only a simple generic simulation world for multi agent experiments (instead of the various simulation environments provided in MicroPsi 1). Also, 3D viewing components for environments and facial expressions are completely absent. The current priority of MicroPsi 2 lies on affective simulation for problem solving experiments (see Bach 2012b), and its application as a general framework for knowledge representation in a hierarchical semantic network." [Bac12]

3.1.2.4 Core

Then, there are nodenets. Nodenets can be stored and edited graphically from within the web-interface.

World-adapters are, what makes nodenets and worlds interact with each other. Therefore they provide Data-Sources and Data-targets, which embody the nodenets input from the world (it's senses/sensors) and output (it's muscles?/motors/actuators).

... the core runs the heart of the simulation ...

3.1.2.5 Server

The server is built on the Python webframework Bottle. [Bac12] ... the server provides the interface ...

3.1.2.6 Simulation Environments

... so far, there are an "Island" and an "Berlin" worlds ...

3.2 A! Minecraft

... a more complex simulation environment could be fun ...

... the story of Minecraft ... ... Minecrafts poularity (and demographics) ...

The story of Minecraft has many interesting sides, but one kind it is without doubt: a story of succes.

When Markus ("Notch") Persson built and released the first public version of Minecraft it soon became clear that his creation resonated with many people. The simple concept of world entirely build of standard sized building blocks which the player is allowed to create, destroy and relocate one-by-one let many gamers employ their creativity, explore the Minecraft world and test out it's possibilities and boundaries.

One computer-game in particular stood out in the last years — not for A.I. reasons though. It is called Minecraft and is benefiting of high popularity ever since its first release in 2009. Since copies of the game can be obtained commercially for the first time in 2011, the different versions of the game sold more than 26 million times - the PC version priced at about 20 Euros. It should be noted, that the games developer studio Mojang is a so called "Indie" developer that is not associated with any classical game publisher but distributes copies of their game exclusively via their own website.

Even though the game can be downloaded and played as a single packet of software, many scenarios of playing the game consist of running a Minecraft server software as well as one client per player. It is possible to mimic the official client by implementing the reverse-engineered Client-Server-Protocol and build artificial players that way.

Minecraft is a complex yet easily accesible virtual world. It is constantly developed and new features are added regularly. It is a massive fanbase and a huge community of game-modifications.

Another interesting aspect about Minecraft is the procedural semantic the game world is generated with and. Trees in Minecraft, for example, may share a similar structure that consists of a trunk and branches and leaves spreading out as fractals, but the particular charecteristics of each tree are generated randomly. This makes a Minecraft world somewhat more realistic than most other videogames.

3.2.1 What is a Minecraft world?

In this section some of the basic concepts (in regards to our A.I. interface) of the game are described. Minecraft worlds are build out of blocks (see figure 3.1). Blocks are cubes. There are different kinds (materials) of blocks and they have a single size, which converts to the basic distance unit of Minecraft. One unit can rougly be thought of as equaling one meter.

Figure 3.1: *A "grass" Minecraft Block*

A chunk (see figure 3.2) is a segment of the Minecraft world that is 16 blocks long, 16 blocks wide and 256 blocks high (or deep). [mcw13]

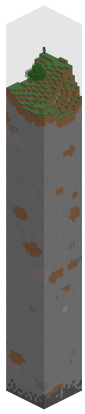


Figure 3.2: *A chunk*

"The player" (see figure 3.3) is what the playable game-character in Minecraft is called. Usually it is displayed as a humanoid.

Figure 3.3: *"the player" (CC-BY-3.0 Mojang AB) [ima13]*

... brief description of the basic mechanisms ... ... the game has now predefined "goal" ... other than monsters trying to kill you at night ...

3.2.2 The Cient Server Protokoll

Minecrafts Client-Server-Protocol is (at least not to the public) documented by the developer themselves. However, the modding-community found it's ways to gather full knowledge of its structure (probably by using reverse engineering techniques). The protocol is based on packets.

"All packets begin with a single "Packet ID" byte. Listed packet size includes this byte. Packets are either "server to client", "client to server", or "Two-Way" (both). Packets are not prefixed with their length. For variable length packets, you must parse it completely to determine its length."

To give an example of one of the easier packets, the "Client Position"-Packet is fairly straight forward. It is exclusively send form clients to servers and starts with the Packet ID (as every Packet does), followed by the X- and Y-coordinates as doubles, the stance value as a double, which is used to to "modify the players bounding box when going up stairs, crouching, etc...", another double for the Z-coordinate and eventually a boolean that describes if the player is on the ground or not. [pro13]

Knowledge of this datastructure is already sufficient to move around in the Minecraft world. To go forward, one has to figure out the players current position, calculate the absolute coordinates of the destination of the movement in regard of it and send a "Client Position" packet with these coordinates to the server. If the destination is not more than 100 blocks away from the origin, the server accepts the packet. In the official Minecraft client, a players movement from one point to the other is rendered with a smooth transition.

Other than movement, there are defined packets(-structures) for every aspect of the game. May it be the initial handshake, the creation or destruction of blocks or activity of other player- or non-player-characters

... the language an external client needs to speak, to take place in a Minecraft world ...

3.2.3 Suitability of Minecraft as a simulation environment

There are a number of reasons why using Minecraft as a simulation environment could be useful and lead to interesting results.

First, the game itself is easily accesssible. It is developed using Java (for both the client and the server software) and therefore somewhat platform independent. The "desktop computer version" is sold for Windows, Mac OS X and Linux devices. There are official ports to Android, iOS, Xbox 360, the Raspberry Pi an a version for the upcoming "Xbox One" is announced. The desktop versions are priced at 19.95 euros which makes it affordable to a large audience.

The game itself already has an enormous audience. It is (like most videogames) espacially popular with teenagers (or upcoming scientists). Minecraft being loved by so many people could benefit this project as for giving it more attention.

The game's developer has prooven many times (proof?) that it acts very friendly to other developers when it comes to building game modifications and creating content that uses and/or changes original Minecraft content. In other words: They are not at all restrictive, when it comes to user people doing all kinds of things with their creations. This led to the availability of a fairly complete community-sourced documentation and explantion of virtually every aspect of the game — including it's software architecture, data structure and protocols. This is usefull for this project, as chances are low that they will have anything against this project in the forseeable future. (In fact, the game's A.I. creator Jon Kagström seems to be fond of this project)

The Minecraft world with it's logic, semantic and functions offers possibilities for an A.I. to proof being able to interact with the environment both on very simple and primitive ways (e.g. moving around) as well as increasingly complex tasks like building, collecting ressources, craft items from blocks and interacting apropriately with both well-disposed and hostile other incarnations.

The semantics of the gameworld share charecteristics with the real world. Moving through the world one quickly realizes that it is build up out of different biomes (eg. forrest or tundra). Also trees, rivers, mountains and ore veins are not hardcoded but generated proceduarally and their structure appers to be (somewhat) fractal.

... cheap licenses ... ... developer friendly community and game-studio ... ... sandbox game with many possibilieties but no pre-defined goals ... ... procedural semantic ...

3.3 A! Minecraft and MicroPsi

Using Minecraft as a simulation environment will give the Psi agent possibilities to show off what kind of sophisticated behavior it is capable of. ... a more complex simulation environment could be fun ...

Chapter 4

A! Approach / Minecraft as a Simulation Environment for MicroPsi 2

The objective of this thesis is to build and test an interface in between MicroPsi and Minecraft, so that a Minecraft world (e.g. server) can be used as a simulation environment for the MicroPsi 2 Framework, which will act as an artificial player.

The modular architecture of MicroPsi 2 allows it to add new simulation environments (or worlds, as they are called in microPsi) fairly easily. A world needs interfaces to Data Source and Data Targets and a step-function that evolves the world. Those are provided by the so called world adapter.

On the other hand, communication with a Minecraft Server typically requires a constant flow of data packets going in and out. Most third party clients, including Bots, facilitate own event loops. What has been done for this project, was breaking down the event loop of the used bot framework and rebuild it as the step function of the MicroPsi 2 world. It should be noted, that the frequency in which the framework steps the bot has to be at least chosen so, that it is able to send keep-alive-signals to Server, to not get kicked.

That being said, a big part of the project is about visualization. Inside the MicroPsi Core Application, a 3D-visualization of the Minecraft world and the agent within is aimed for. There are two main reasons for this goal. The first reason is, that the agents behavior within the simulation environment is supposed to be monitored from the MicroPsi webinterface in an aesthetically appealing way. The secons reason is, that the image data is supposed to be processed by the agent as one of it's senses.

Do obtain these goals, the Minecraft Client-Server-Protocol had to be researched, learned and imitated.

Then, artificial Minecraft players (written in Python) had to be searched, found and researched.

Eventually, building upon an existing Bot framework led to an integration with the MicroPsi framework.

4.1 A! Overview / What has been there so far?

There are many projects out there, that could be considered "bots". One has to differentiate in between two types. There are those, that mimic an entire client software and fascilitates communication with the server on it's behalf. The other ones are modifactions of the original client (or server) software and usually add non player characters — like animals and other non-human creatures. The code is usually injected through one of the popular "modloaders" (eg. Minecraft Forge).

One example (and probably the most advanced one) for an entire bot framework that replaces the client is Mineflayer. [git13] It has a high-level abstraction of the environment (eg. entity knowledge and tracking) and is written in JavaScript using node.js .

Opposed to Mineflayer, an example for a game modification bot are the "Cubebots" — fanmade cubes that aim to help Minecraft players with mondane tasks.[**?**]

... Minecraft Bots with simple as well as sophisticated AI ... ... MicroPsi 2 with Island and Berlin world ...

4.2 A! Architecture / Building the interface in between Minecraft and the simulation environment

... result: a Minecraft Bot that implements MicroPsi AI and is controlled and monitored via the MicroPsi webinterface ... ... the Webinterface holds its own visualization of the Agents worldview ...

4.2.1 A Minecraft Bot

4.2.1.1 Protocol Implementation

4.2.1.2 Control Structures

4.2.1.3 Previous own implementations with TwistedBot

4.2.1.4 other popular Bot projects and game modifications

4.2.1.5 Spockbot von Nickelpro

Developed by Nick Gamberini, spock is an open-source bot framework (and therefore a Minecraft client) written in Python. It has been chosen as an essential part of this project for two reasons: Beeing written in Python it painlessly integrates in existing MicroPsi code and the absence of dependencies (with on exception) leave the code understandable and easy to deploy.

4.2.2 Implementing the Bot in MicroPsi

To make the bot work as a simulation world two main challenges had to be overcome. First, the event loop and handling of spock had to be modified to work with MicroPsi. Therefore, the functions that were usually called from within the eventloop now have to be called from within the step function of the worldadapter. The same holds for event handling.

Second, a system for communicating "sensory data" and commands in between the bot (or Minecraft world) and the world adapter had to be implemented. Up to a certain degree, spocks plug-in system could be facilitated (well ...). In most/ cases, though, sending commands and receiving data (...) had to be be implemented on packet level.

Sending packets in spock is fairly easy. (see figure 4.1

... spocks plugin system ...

```
            'x': (client.position['x'] + 1)  / 1,
               'y': client.position['y'] / 1,
            'z': client.position['z'] / 1,
            'on_ground': False,
            'stance': client.position['y'] + 0.11
            }))
```

Figure 4.1: *Sending a packet that moves the agent on block in x direction*

4.2.3 The MicroPsi side

4.2.4 necessary Modifications and Additons in core/worldrunner

Data Targets and sources

4.2.5 necessary Modifications and Additons in server/control and monitoring interface

4.2.6 The Visualization

4.2.6.1 Requirements and necessity of a visualization

... why a visiualization and what is it supposed to do? ...

monitoring the bot from the webinterface ... make it aesthetically appealing as well as easily accessible ...

using visualization output as a Datasource / as the bots eyes

4.2.6.2 Implementation

used Data ... required Data for the visualization ... ... and how to obtain it (first attempts: telnets/then sockets) ...

4.2.6.3 3D Visualisierung mit Pyglet

... foundation: that minecraft pyglet clone ...

4.2.6.4 Earlier attempts using JavaScript / AJAX

... worked well but a little slow ...

4.3 A! Implementation

4.4 A! Case Study

The functionality is to be tested with a simple Braitenberg-vehicle experiment.

4.4.1 Experiment

... experiment to test functionality of the system ... ... scope: only a simple test for time reasons ...

4.4.1.1 Braitenberg Vehicle

... simplest proof of concept of a microPsi agent ...

Chapter 5

Conclusion

After several iterations and trying out different approaches and technologies the Interface is now functional.

The experiment with the simulated Braitenbergvehikel resulted in proofing that Minecraft is usable as a simulation environment.

5.1 What's next

In the future, multiple agents shall interact with the same environment and collaborate with each other. ... what has been learned ... ... what can be done with the new environment ... ... what can be improved? ... ... what other simulation environments could be of interest? ...

Appendix A

Auszug aus dem Buch

Beispielhaft wird hier gezeigt, wie Block-Zitate eingeführt werden können. Hier der Beginn des Buchs "'Per Anhalter durch die Galaxis"' von Douglas Adams [Ada98]:

> "'Das Haus stand auf einer kleinen Anhöhe genau am Rand des Ortes. Es stand alleine da und überblickte das weite Ackerland im Westen. Absolut kein bemerkenswertes Haus - es war ungefährt dreißig Jahre alt, plump, viereckig, aus Ziegelsteinen erbaut und hatte vier Fenster an der Vorderseite, der es nach Größe und Proportion mehr oder weniger mißlang, das Auge zu erfreuen."'

Appendix B

Implementierungen

Beispielhaft wird hier gezeigt, wie Code-Beispiele in den Text eingefügt werden können. Die Pseudocode-Umgebung wird von *macros.tex* bereitgestellt und kann dort entsprechend angepasst werden.

```
public static Object answeringMachine() {
   Thread.sleep(1000);
   return 42;
}
```

Figure B.1: *Implementierung einer Maschine zur Beantwortung der Fragen aller Fragen.*

List of Figures

List of Tables

Inhalt der beigelegten CD

Die beigelegte CD enthält folgenden Inhalt:

- diese Masterarbeit in PDF Format,
- Videos mit Interview von Fans,
- den Source-Code der Implementierung einer Maschine zur Beantwortung der Fragen aller Fragen. Der Source-Code ist im Ordner *src* zu finden.

Bibliography

[Ada98] In: Adams, Douglas: *Per Anhalter durch die Galaxis: Roman*. Heyne Verlag, 1998, S. 1

[Bac09] Bach, Joscha: *Principles of Synthetic Intelligence PSI: An Architecture of Motivated Cognition*. 1st. New York, NY, USA : Oxford University Press, Inc., 2009. – ISBN 0195370678, 9780195370676

[Bac12] Bach, Joscha: MicroPsi 2: The Next Generation of the MicroPsi Framework. In: Bach, Joscha (Hrsg.) ; Goertzel, Ben (Hrsg.) ; Iklé, Matthew (Hrsg.): *AGI* Bd. 7716, Springer, 2012 (Lecture Notes in Computer Science). – ISBN 978–3–642–35505–9, 11-20

[git13] `https://github.com/superjoe30/mineflayer`

[ima13] `http://www.minecraftwiki.net/wiki/File:Mob1.png`

[mcw13] `http://www.minecraftwiki.net/wiki/Chunks`

[pro13] `http://wiki.vg/Protocol`