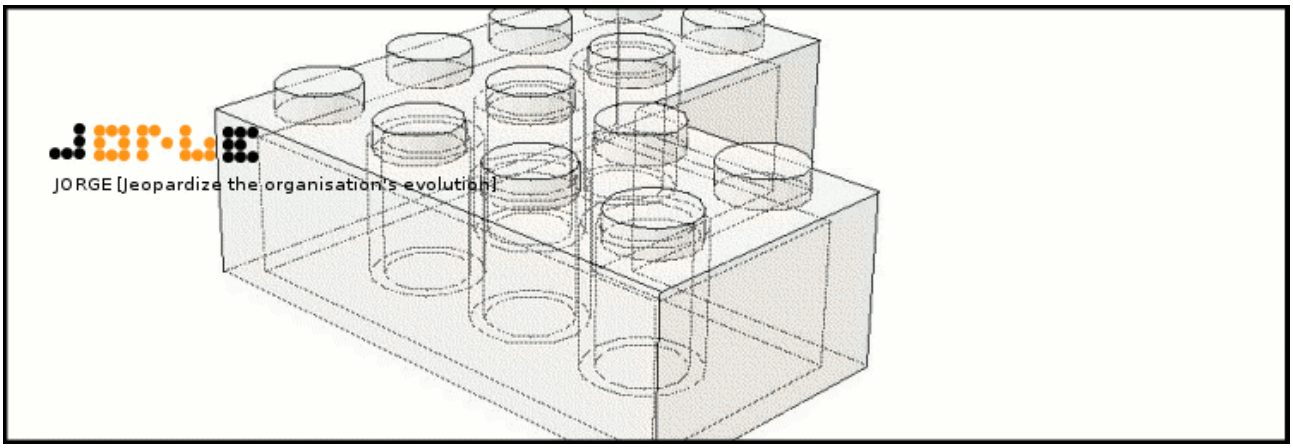
Berner Fachhochschule
Hochschule für Technik und Informatik HTI



Pflichtenheft

[JORGE - LEGO MindStorms Simulator]

Autoren	Nik Lutz [lutzn@hti.bfh.ch]
	Stefan Feissli [feiss@hti.bfh.ch]
	Christof Seiler [seilc@hti.bfh.ch]
Version	1.0
Datum	19.11.2005
Status	Freigegeben
Webseite	jorge.hta-bi.bfh.ch
Projektauftraggeber	Claude Fuhrer
Projektarbeit	Wintersemester 2005, HTI Biel, Abt. Informatik

Inhaltsverzeichnis

1 Einleitung

Das Pflichtenheft basiert auf der Aufgabenstellung der Diplomarbeit [1] und dem Projektantrag [2]. JORGE steht für *Jeopardize the organisation's evolution* und ist eine Simulationsumgebung für Legoroboter.

1.1 Zweck des Dokuments

Das Pflichtenheft beschreibt die Ziele, welche mit der angestrebten Lösung zu erreichen sind sowie die Anforderungen und Wünsche an das zukünftige System.

1.2 Erwähnte Dokumente und Referenzen

Nr.	Referenzen	Version	Autor
[1]	Aufgabenstellung Diplomarbeit		Claude Fuhrer
[2]	Projektantrag	1.05	Team JORGE
[3]	LeJOs: www.lejos.org	2.1.0	
[4]	Ogre: www.ogre3d.org	1.0	
[5]	LeoCAD:		

2 Ausgangslage

Der in der Semesterarbeit erstellte Prototyp demonstriert das Zusammenspiel der von uns gewählten Technologien. Der Benutzer unseres Prototypen kann ein selbst erstelltes LeJos-Programm in den virtuellen Roboter laden, es ausführen und den Roboter in der virtuellen Welt beobachten.
Die Bedienung des Prototypen ist sehr eingeschränkt: Zum Beispiel muss das auszuführende Programm dem Prototypen an der Kommandozeile übergeben werden und die physikalischen Eigenschaften der virtuellen Welt und des Roboters können nur in XML-Dateien verändert werden.
Der produktive Einsatz unseres Prototypen können wir niemandem zumuten, da dessen Benutzung wenig intuitiv ist.

3 Ziele

Am Ende unserer Diplomarbeit wollen wir den interessierten Benutzern ein lauffähiges Programm anbieten können, das über ein intuitive Benutzeroberfläche verfügt. Zudem soll die Dokumentation und die Architektur der Software verständlich und übersichtlich sein. Die genauen Ziele sind in diesem Dokument unter Anforderungen ausführlicher beschrieben.

4 Anforderungen

4.1 Musskriterien

Am Ende der Projektarbeit soll ein lauffähiges System erstellt worden sein, das unter Microsoft Windows fehlerfrei läuft. Beim Programmieren ist darauf zu achten, dass die Software später auch auf andere Betriebssysteme portiert werden kann, d.h. es sollen nur Platformunabhänige Bibliotheken (Libraries) und Standard C++ Funktionen verwendet werden. Dort wo dies nicht möglich ist, muss es im Code und in der Dokumentation vermerkt sein.

- Wir entwickeln eine intuitive und verständliche Benutzeroberfläche mit Standardkomponenten wie Textfeldern, Menus, usw.
- Es können mehrere Roboter in die virtuellen Welt eingefügt, konfiguriert und gestartet werden.
- Der Benutzer kann verschiedene Hindernisse in die virtuelle Welt einfügen, um die Interaktion seines Programs zu testen.
- Wir Dokumentieren und zeigen anhand eines Beispiels, wie neue Sensoren erstellt werden können.
- Roboter sollen untereinander kommunizieren können. D.h. die Infrarot-Funktionen des LeJos-API’s können vom Programmierer genutzt werden.
- JORGE soll auch auf weniger performanten Rechnern möglichst ruckfrei laufen. Um dies zu gewährleisten, können Einstellungen in den Bereichen Physik, Ogre und Emulator vorgenommen werden.
- Der Benutzer kann eigene Hindernisse definieren, die er dann in die virtuelle Welt einfügen kann.

Zudem soll eine ausführliche Dokumentation erstellt werden. Sie soll einerseits dem Anwender zeigen wie unsere Applikation genutzt werden kann, andererseits soll sie künftigen Entwicklern, die unsere Applikation erweitern möchten, den Einstieg so einfach wie möglich machen.

4.2 Nice to have

Falls die Implementation der Musskriterien optimal verläuft sind mehrere Erweiterungen möglich.

- Ein Program, um Lego-Roboter aus LeoCAD zu exportieren. Damit könnte der Benutzer ein eigenes Roboter-Modell kreieren und in die Simulation aufnehmen.
- Ein Framework um neue Sensoren zu erstellen. So erstellte Sensoren können zur Laufzeit als dynamische Bibilothek geladen werden und in der Simulation eigesetzt werden.
- Skins (optische Hüllen die dem Roboter ein anderes Aussehen geben) für Roboter.
- Der Benutzer kann eine eigene Welt innerhalb unserer Applikation erstellen.

4.3 Systemanforderungen

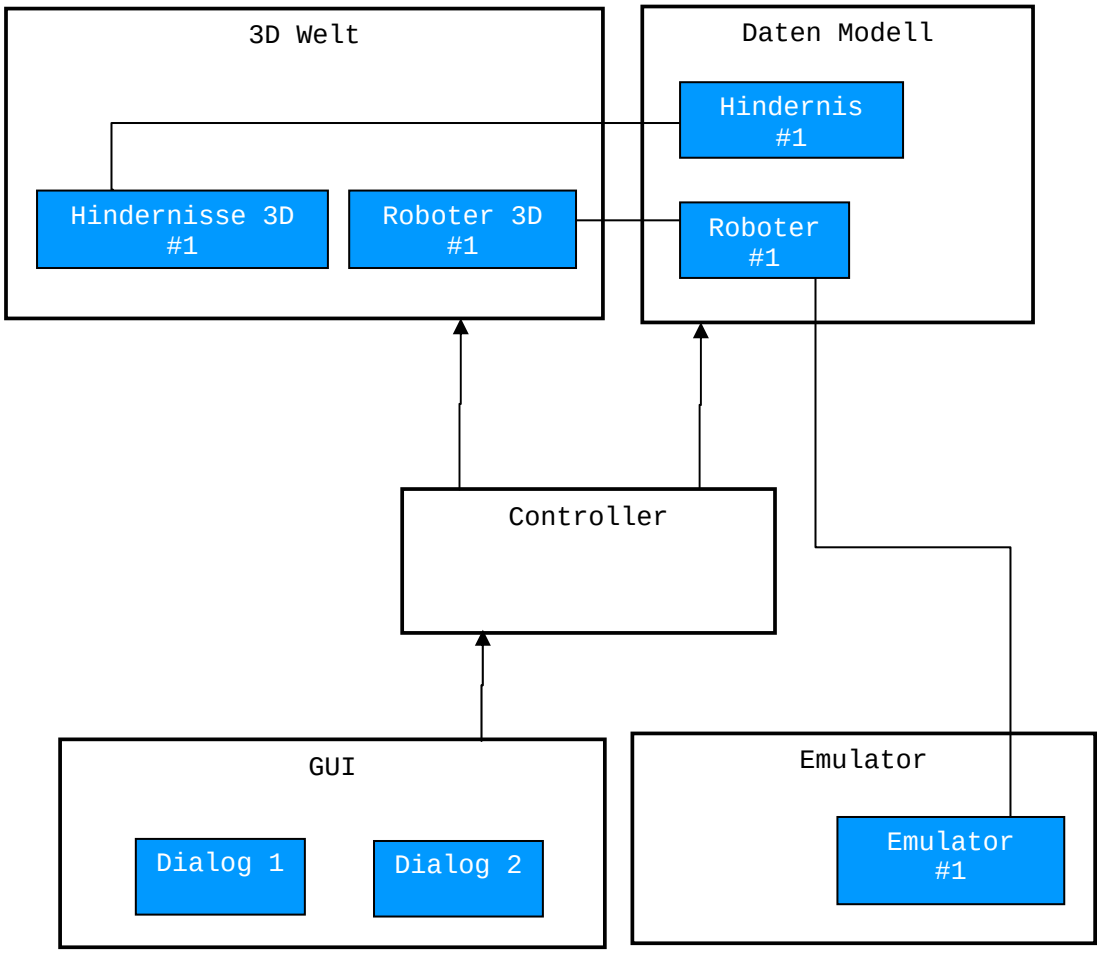
4.3.1 Konzept

Das System wird in grob 5 Komponenten aufgeteilt:

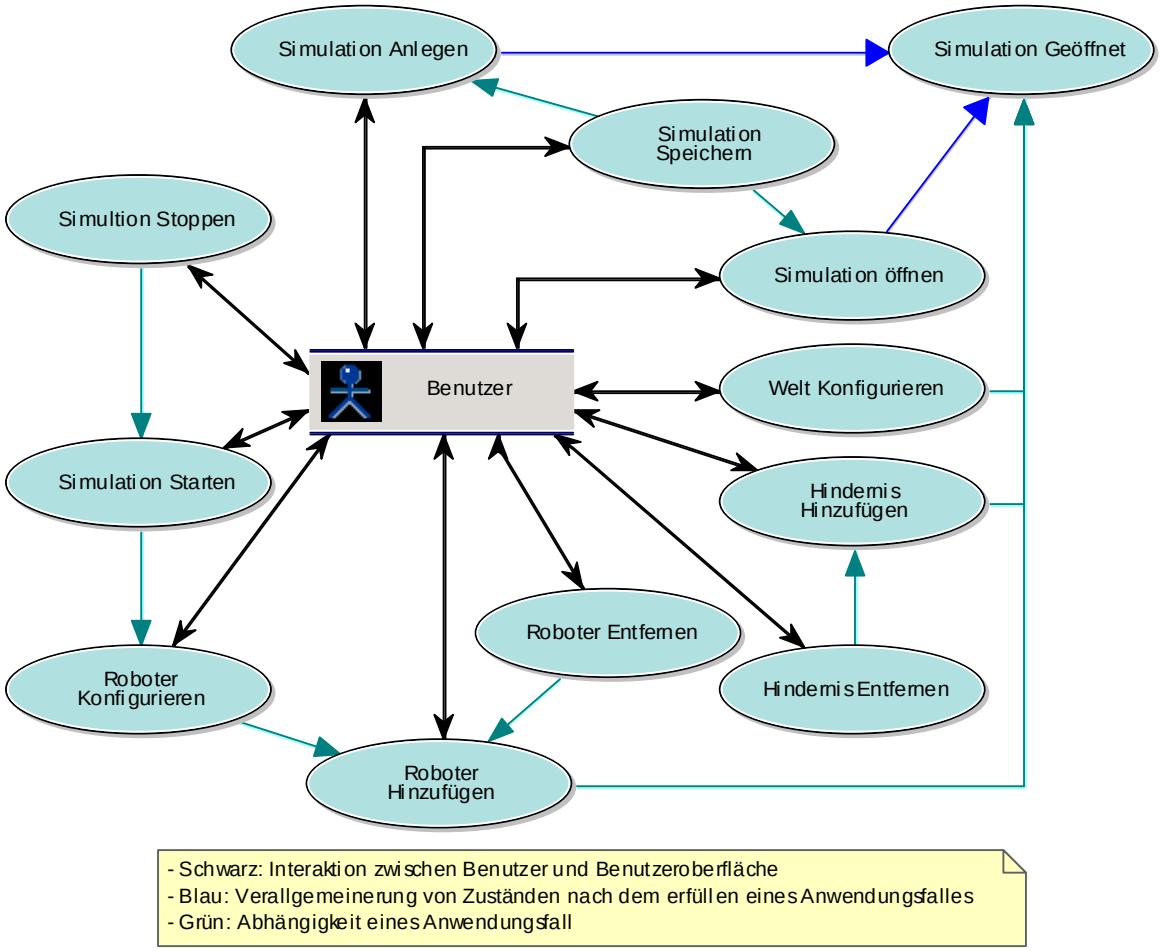
1. Eine Binär-Schnittstelle (Emulator) zu Lego-Programmen, die den Austausch von Informationen wie Sensoraktivität oder Motoren-Steuerungs-Befehlen zwischen JORGE und den Lego-Programmen regelt. Die Schnittstelle ist die primäre Eingabe- und Ausgabeeinheit für die Steuerung eines virtuellen Roboters.
2. Ein Controller, der den Ablauf und Zustand der Applikation regelt. Er wird primär von der Benutzeroberfläche angesprochen und delegiert Aufgaben an die verschiedenen anderen Komponenten, wie GUI und Modell.
3. Ein Daten-Modell, das die Konfiguration der Roboter und der virtuellen Welt auf eine geeignete Datenstrukturen abbildet, und somit ermöglicht, dass die Simulation abgespeichert und später wieder geladen werden kann.
4. Das Modell der dreidimensional Welt. Die Primäre Aufgabe des Models ist die Darstellung der 3D-Welt/Landschaft und der sich darin bewegenden Roboter. Der Benutzer sollte aber auch die Möglichkeit haben, zum Beispiel die Perspektive zu wechseln. Neben der Darstellung der Objekte soll das Modell auch die physikalischen Eigenschaften der 3D-Welt simulieren/überwachen.
5. Eine Benutzeroberfläche, die dem Benutzer die Möglichkeit bietet, neue Lego-Programme bequem zu laden, Roboter und die virtuelle Welt zu konfigurieren und schliesslich die Simulation zu starten.

4.3.2 Übersichtsmodell

Das Übersichtsmodell zeigt grob das Zusammenspiel der verschiedenen Komponenten.



4.3.3 Anwendungsfälle (Use Cases)



Anwendungsfall:	**Simulation Anlegen**
Kurzbeschreibung:	Eine neue Simulation wird erstellt.
Vorbedingung:	Applikation gestartet.
Nachbedingungen:	Der Benutzer sieht die von ihm gewählte 3D Welt in einem Fenster.
Auslöser:	Der Benutzer wählt die entsprechende Option im Menu

Anwendungsfall:	**Simulation Speichern**
Kurzbeschreibung:	Speichern der gewählten Roboter-, Hinderniss- und Welteinstellungen.
Vorbedingung:	Simulation geöffnet
Nachbedingungen:	Korrektes XML

Auslöser:	Der Benutzer wählt die entsprechende Option im Menu
Anwendungsfall:	**Simulation Öffnen**
Kurzbeschreibung:	Der Benutzer kann eine Simulation via Datei-Öffnen-Dialog laden.
Vorbedingung:	Applikation gestartet.
Nachbedingungen:	Der Benutzer sieht abgespeicherte 3D Welt mit allen Robotern und Hindernissen.
Auslöser:	Der Benutzer wählt die entsprechende Option im Menu.
Zustand:	**Simulation Geöffnet**
Kurzbeschreibung:	Vorbedingung für andere Anwendungsfälle.
Vorbedingung:	Applikation gestartet.
Nachbedingungen:	Die 3D Welt wird in einem Fenster angezeigt
Auslöser:	Dieser zustand wird mit den Anwendungsfällen „Simulation Öffnen“ und “Simulation Anlegen“ erreicht
Anwendungsfall:	**Welt Konfigurieren**
Kurzbeschreibung:	Die physikalischen Eigenschaften der virtuellen Welt können definiert werden.
Vorbedingung:	Simulation geöffnet.
Nachbedingungen:	Die gewählten Eigenschaften werden in der virtuelle Welt gesetzt.
Auslöser:	Der Benutzer drückt den entsprechenden Eintrag in der Toolbar.
Anwendungsfall:	**Roboter Hinzufügen**
Kurzbeschreibung:	Es können Roboter zur Simulation hinzugefügt werden.
Vorbedingung:	Simulation geöffnet.
Nachbedingungen:	Dem Benutzer wird der neu erstellte Roboter mit Standard-Einstellungen angezeigt.
Auslöser:	Der Benutzer drückt den entsprechenden Eintrag in der Toolbar.
Anwendungsfall:	**Roboter Konfigurieren**
Kurzbeschreibung:	Die physikalischen eigenschaften und das auszuführende Program können definiert werden.
Vorbedingung:	Mindestens ein Roboter wurde hinzugefügt.
Nachbedingungen:	Die gewählten einstellungen werden auf das 3D-Roboter-Modell übertragen.
Auslöser:	Der setzt die entsprechende Option.
Anwendungsfall:	**Roboter Löschen**
Kurzbeschreibung:	Ein Roboter kann aus der Simulation entfernt werden.
Vorbedingung:	Mindestens ein Roboter wurde hinzugefügt.

Nachbedingungen:	Der Roboter und seine Einstellungen werden gelöscht.
Auslöser:	Der Benutzer drückt den entsprechenden Eintrag in der Toolbar.

Anwendungsfall:	**Hindernis Hinzufügen**
Kurzbeschreibung:	Der Benutzer kann Hindernisse auswählen und in die virtuelle Welt einfügen.
Vorbedingung:	Simulation geöffnet.
Nachbedingungen:	Das gewählte Hindernis wird in der virtuellen Welt dargestellt.
Auslöser:	Der Benutzer wählt ein Hindernis aus, und plaziert es in der virtuellen Welt.

Anwendungsfall:	**Hindernis Entfernen**
Kurzbeschreibung:	Ein Hindernis kann aus der Simulation entfernt werden.
Vorbedingung:	Mindestens ein Hindernis wurde in die Welt eingefügt.
Nachbedingungen:	Das gewählte Hindernis wird aus der Simulation entfernt.
Auslöser:	Der Benutzer wählt das entsprechende Hindernis aus, und wählt entfernen.

Anwendungsfall:	**Simulation Starten**
Kurzbeschreibung:	Die in die Roboter geladenen Programme werden gestartet.
Vorbedingung:	Roboter konfiguriert (auszuführendes Program ausgewählt)
Nachbedingungen:	Der/Die Roboter bewegen sich gemäss den Anweisungen der geladenen Programme. Der Roboter und die virtuelle Welt können nicht mehr konfiguriert werden.
Auslöser:	Der Benutzer drückt den entsprechenden Eintrag in der Toolbar.

Anwendungsfall:	**Simulation Stoppen**
Kurzbeschreibung:	Die Simulation kann abgebrochen werden.
Vorbedingung:	Simulation wurde gestartet.
Nachbedingungen:	Die Simulation stoppt, der Roboter und die Welt können wieder konfiguriert werden.
Auslöser:	Der Benutzer drückt den entsprechenden Eintrag in der Toolbar.

4.4 Anforderung an unsere Algorithmen

Auf den uns zur Verfügung stehenden Computern der Schule wollen wir mindestens 12 fps erreichen. Im allgemeinen muss sicher auch darauf geachtet werden, dass die Geschwindigkeit der Abläufe in der Simulation nicht nur von der Hardware eines Systems abhängt, sondern auch konfigurierbar ist, da sonst auf schnellen Systemen Einzelheiten der Simulation nicht mehr wahrnehmbar wären.

4.5 Minimale Hardwareanforderung

Durch die 3D-Simulation und das Verwenden neuster Softwaretechnologien, ist JORGE eine rechenintensive Anwendung. JORGE wurde auf Rechnern mit folgender Hardware entwickelt und getestet:

Prozessor: Intel Pentium 4 Prozessor
Grafikkarte: Nvidia NV45GL 256 MB
Arbeitsspeicher: DDR 2 GB
Auf Hardwaresystemen dieser Art oder höher kann eine einwandfreie Ausführung gewährleistet werden.