|  |
| --- |
| Unreal Cup  **STUDIENARBEIT**    im Studiengang Informationstechnik  an der DHBW Ravensburg  Campus Friedrichshafen  von  Michael Kekeisen,  Michael Möbius,  Daniel Rapp,  Maximilian Schmitz  09.01.2015  Bearbeitungszeitraum 13.10.2014 – 09.01.2015  Matrikelnummer 4468889,  2459628,  6014276  9322432      Betreuer Prof. Erwin Fahr |

|  |
| --- |
|  |

# Eidesstattliche Erklärung

gemäß § 5 (3) der „Studien- und Prüfungsordnung DHBW Technik“ vom   
22. September 2011.

Hiermit erkläre ich, dass ich die vorliegende Arbeit mit dem Titel

**Unreal Cup**

selbständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt, keine anderen als die angegebenen Hilfsmittel benutzt und wörtliche sowie sinngemäße Zitate als solche gekennzeichnet habe.

xx, den xx.xx.2015

Unterschrift

# Kurzfassung

(Todo: Hier mal die Imagine Cup variante, mal drüber gehen…)

Moderne Computerspiele bieten unterschiedlichsten Zielgruppen eine abwechslungsreiche Freizeitbeschäftigung. Unter der Vielzahl dieser Spiele existieren auch solche, die unter Verwendung von strategischen Elementen den Spieler herausfordern, mit verschiedenen Herangehensweisen ihren Erfolg zu maximieren.

UnrealCup bietet mit der Möglichkeit, eigene künstliche Intelligenzen für eine Fußballsimulation zu erstellen, eine Vertiefung dieser strategischen Elemente. Das Ziel für die Anwender ist es, ihr so zusammengestelltes Team in einem kompetitiven Wettstreit gegen Teams anderer Nutzer zum Sieg zu führen.  
Neben der spielerischen Funktion hat UnrealCup außerdem den Effekt, dass die Anwender ihre Fähigkeiten in der Programmierung und KI-Entwicklung erweitern können.

Der UnrealCup-Simulator bietet für dieses Ziel neben moderner 3D-Grafik und einer komfortablen Schnittstelle zum Einbinden der KI alle wichtigen Funktionen und Regeln, um dem Nutzer eine realistische Simulation und ein packendes Erlebnis zu bieten. UnrealCup beinhaltet einen graphischen Editor, um auch Nutzern, die wenig Erfahrung mit Programmierung besitzen, die Möglichkeit zu bieten, ein Team nach ihren Wünschen zu erstellen.

Neben der Bereitstellung aller notwendigen Spielerfunktionen zum Erstellen einer leistungsfähigen KI wird auch dafür gesorgt, dass alle für Simulationszwecke sinnvollen Fußballregeln eingehalten werden.  
Der UnrealCup-Simulator besitzt also Ähnlichkeiten zur Simulation League des bekannten RoboCups, hebt sich allerdings durch diverse Alleinstellungsmerkmale wie eine moderne Grafik, realitätsnahe Physik und einen graphischen Editor von vergleichbaren Projekten ab.

# Abstract

(Imagine Cup Kurzfassung, drüber gehen…)

Modern video games offer a varying leisure-time activity for diverse target groups. Among the variety of video games there are games that challenge the player to maximize his success by offering the possibility to use strategic elements.

UnrealCup deepens those strategic parts of the game by allowing the user to create own artificial intelligence for his teams players.  
The aim for the user is to lead their assembled team to victory in competitive contests against other players.

Aside of this playful function UnrealCup allows its users to expand their skills in programming and AI-development.

The UnrealCup-Simulator offers, aside of modern 3D-graphics and a comfortable interface to integrate the AI, important functions and football rules to give a realistic simulation and a thrilling experience. UnrealCup also contains a graphical WYSIWYG-Editor to enable users without lots of experience in programming to create their own team.

Besides offering all player functions needed to create an efficient AI the UnrealCup-Simulator also makes sure that all soccer rules which are reasonable in a simulation are considered.  
The Simulator has similarities with the simulation league of the known RoboCup, yet it stands out from similar projects with its modern graphics, realistic physics and the graphical editor.

# Inhalt

[Eidesstattliche Erklärung I](#_Toc418083318)

[Kurzfassung II](#_Toc418083319)

[Abstract III](#_Toc418083320)

[Inhalt IV](#_Toc418083321)

[1 Einleitung 1](#_Toc418083322)

[2 Aufgabenstellung 2](#_Toc418083323)

[3 Stand der Technik 3](#_Toc418083324)

[3.1 Einleitung 3](#_Toc418083325)

[3.2 Game Engine 3](#_Toc418083326)

[3.2.1 Grundsätzliches 3](#_Toc418083328)

[3.2.2 Source Engine 3](#_Toc418083329)

[3.2.3 Unity 4](#_Toc418083330)

[3.2.4 Unreal Engine 4 4](#_Toc418083331)

[3.2.5 Verwendung im Projekt 5](#_Toc418083332)

[3.3 Scripting-Ansatz 5](#_Toc418083333)

[3.3.1 Grundsätzliches 5](#_Toc418083334)

[3.3.2 Lua 5](#_Toc418083335)

[3.4 Robo Cup (Kurz Halten) 6](#_Toc418083336)

[3.5 Künstliche Intelligenz 7](#_Toc418083337)

[3.6 Zusammenfassung 9](#_Toc418083338)

[4 Planung / Struktur 10](#_Toc418083339)

[4.1 Projektmanagement 10](#_Toc418083340)

[4.2 Analyse der Aufgabenstellung 10](#_Toc418083341)

[4.2.1 Muss-Kriterien 10](#_Toc418083342)

[4.2.2 Soll-Kriterien 11](#_Toc418083343)

[4.2.3 Kann-Kriterien 11](#_Toc418083344)

[4.2.4 Analyse der Kriterien 11](#_Toc418083345)

[4.3 Architektur 12](#_Toc418083346)

[5 Umsetzung 14](#_Toc418083347)

[5.1 Umsetzung der Simulationsumgebung 14](#_Toc418083348)

[5.1.1 Grundsätzliches 14](#_Toc418083349)

[5.1.2 Regelimplementierung 14](#_Toc418083350)

[5.1.3 Scripschnittstelle 14](#_Toc418083351)

[5.2 Entwicklung der künstlichen Intelligenz 14](#_Toc418083352)

[6 Ausblick 15](#_Toc418083353)

[7 Fazit 16](#_Toc418083354)

[Literaturverzeichnis (Grundlagenliteratur) I](#_Toc418083355)

[Abbildungsverzeichnis II](#_Toc418083356)

[Abkürzungsverzeichnis III](#_Toc418083357)

[Glossar IV](#_Toc418083358)

# Einleitung

Simulationen wie die Robocup Simulation League bieten Interessierten eine Fläche zur Entwicklung und Forschung an Künstlicher Intelligenz. Nur was hinter der Simulation an Aufwand für den reibungslosen Betrieb getätigt wird oder was an Technologien verwendet wird ist weitestgehend unbekannt.

Wir entschieden uns daher eine eigene Fußballsimulation auf die Beine zu stellen, setzen dabei aber auf aktuelle Technologien. Hierbei fallen bestehende Beschränkungen durch z.B. Einfügen einer Höhenachse weg. Dies erlaubt eine wesentlich flexiblere Ausarbeitung von Strategien.

Das Thema der Echtzeit wird eine sehr große Rolle in der Entwicklung des Projektes spielen: Alle Spieler müssen ihre Skripte pseudo parallel ausführen können und dürfen sich gegenseitig nicht beeinflussen oder gar blockieren. Dafür muss auf erlerntes der Vorlesungen Echtzeit und Software Engineering zurückgegriffen werden.

Das fertige Projekt soll auch dem Spieler ein Gefühl für Logik vermitteln, die von der Aufmachung von Robocup zuerst abgeschreckt waren. Damit wird die Zielgruppe erweitert und man erreicht auch Spieler im Casualbereich.

# Aufgabenstellung

Das Ziel dieser Arbeit ist es, eine grafische Fußballsimulation zu erstellen, in welcher der Spieler eigene Logiken für sein Fußballteam hinterlegt und somit deren Verhalten beeinflusst. Dabei besteht ein Team aus jeweils 10 Feldspielern und einen Torwart, wobei jeder dieser Spieler seine eigene Logik haben kann. Während eines Fußballspiels werden die gängigen Regeln beachtet und bei einer Missachtung je nach Stärke des Vergehens geahndet. Innerhalb der Simulation werden die Bewegungen des Balls physikalisch korrekt simuliert. Die Spieler besitzen als Limitierung ein Ausdauersystem, welches verhindern soll, dass ein Spieler dauerhaft sprintet oder den Ball in kurzer Zeit mit maximaler Kraft kickt.

Die Simulation ist komplett dreidimensional und bietet animierte Fußballspieler, wobei der Aktionsumfang der Spieler zuerst auf ein Basisrepertoire beschränkt wird. Dazu gehört die Aktionen „Laufen“, „Schießen“ und „Ball annehmen“. Die Ausarbeitung der grafischen Komponente wird in der Priorität hinter den funktionalen Teil der Arbeit gestellt.

# Stand der Technik

## Einleitung

Die Arbeit baut auf unterschiedlichen Technologien auf, deren Verständnis für die Projektbearbeitung unumgänglich ist. Da die Simulation, wie sie in Kapitel 2 beschrieben wurde, nicht im gegebenen Zeitrahmen von Grund auf Programmiert werden kann, muss eine Game-Engine gewählt werden, die Grundlagen wie die 3D-Animation oder die Physik-Berechnungen bereitstellt.

Des Weiteren muss eine Technologie gewählt werden, die die Steuerung der Charaktere über im Voraus erstellte Abläufe erlaubt.

In diesem Kapitel sollen die Technologien, die für die genannten Zwecke in Betracht gezogen wurden, kurz dargestellt werden und die Technologieentscheidung begründet werden

## Game Engine

### Grundsätzliches

Wie in Kapitel 3.1 angesprochen ist eine Game-Engine notwendig, die die Grundfunktionalitäten einer 3D-Simulation bereitstellt. Es existieren heutzutage mehrere solcher Frameworks, auf die im Folgenden kurz eingegangen werden soll.

### Source Engine

Source Engine wird von Valve entwickelt und ist durch die weit verbreiteten Spiele, die mit ihr Entwickelt wurden, bekannt. Dazu gehört neben Half Life 2 oder Counter Strike: Source auch Portal.

Die Source Engine kann neben den Betriebssystemen Windows, Linux und MAC auch auf der Playstation 3 oder der Xbox360 verwendet werden.

Neben der Grafik-Engine beinhaltet dieses Framework auch eine Physik-Engine.

Ein Problem mit der Source Engine ist, dass sie nicht frei verwendet werden kann sondern kostenpflichtig lizensiert werden muss.

### Unity

Die Engine Unity wird von Unity Technologies entwickelt und unterstützt, wie die Source Engine, sowohl Grafik als auch Physikberechnungen. Ein großer Vorteil von Unity ist die große Anzahl der unterstützten Plattformen. Neben den Betriebssystemen Windows, Mac und Linux und den bekanntesten Konsolen (Playstation 3, Xbox360, Nintendo Wii) unterstützt Unity auch Smartphones (iOS, Android, Windows Phone) und weitere Plattformen.

Unity bietet eine Gratisversion und eine Pro-Version an, die für 1500$ erworben werden kann.

### Unreal Engine 4

Unreal Engine wird von Epic Games entwickelt und ist auf allen verbreiteten Computer-Betriebssystemen lauffähig. Neben diesen unterstützt es allerdings auch die neue Konsolengeneration mit Playstation 4 und Xbox One, sowie mobile Anwendungen mit iOS oder Android.

Seit der erstmaligen Verwendung der Unreal Engine im Computerspiel „Unreal“ im Jahr 1998 konnten viele, teilweise sehr bekannte Spiele erfolgreich auf der Engine aufbauen. Die Unreal Engine 4, das aktuellste Framework der Reihe, wurde in Spielen wie Tekken 7 oder Kingdom Hearts 3 eingesetzt.

Lizensiert wurde die Engine anfänglich für 19€ pro Monat, wobei sie für Studenten und Hochschulen kostenlos zur Verfügung gestellt wurde. Während des Projektzeitraums entfiel die monatliche Gebühr generell, was die Verbreitung der Engine weiter förderte. Bei einem finanziellen Erfolg eines mit der Unreal Engine 4 erstelltem Produkts fordert Epic Games eine prozentualeGewinnbeteiligung.

### Verwendung im Projekt

Da die erwähnten Frameworks sich im Funktionsumfang kaum bis überhaupt nicht unterscheiden war die Auswahl der geeigneten Engine nicht ohne weiteres möglich.

Eines der Hauptmerkmale, die für die Entscheidung relevant waren, war der Preis und die Lizensierung des Produkts.  
Durch diese Betrachtung wurde schnell klar, dass die Source Engine für dieses Projekt nicht in Frage kommt, da hier keine Möglichkeit zur freien Verwendung existiert.  
  
Ein weiterer wichtiger Punkt war die Erweiterbarkeit des Codes zum Einbinden der externen KI. Da ein solcher Ansatz natürlich nicht den Anforderungen an normale Spiele entspricht, bietet keine der von uns in Betracht gezogenen Frameworks eine vorgefertigte Lösung für dieses Problem.  
Aus diesem Grund muss der Code der Game-Engine um eine Schnittstelle erweitert werden, die die Befehle von der KI entgegennimmt. Bei der Unity-Engine ist dies nicht möglich, da dieses Framework in der Gratis-Version keine Erweiterung des Quellcodes um Plugins zulässt.

Aus den oben genannten Gründen fiel die Wahl auf die Unreal Engine 4.

## Scripting-Ansatz

### Grundsätzliches

Da die Steuerung der Spieler, wie in der Aufgabenstellung erwähnt, von außerhalb der Simulation stattfindet, muss ein geeigneter Ansatz gewählt werden, über den eine programmierte KI in die Simulation eingespeist werden kann.

Hierbei fiel die Wahl auf eine Scriptsprache, da diese typischerweise leichtgewichtig und einfach zu erlernen sind.

### Lua

Lua ist eine leichtgewichtige Scriptsprache und kann auf verschiedenen Plattformen eingesetzt werden. Dabei bietet Lua eine Menge von Basisfunktionen die zur Behandlung eines Problems erweitert werden können. Da Lua leicht einzubinden und zu erlernen ist wird die Sprache bereits in vielen Programmen, unter anderem in Videospielen wie Garrys Mod, eingesetzt.

## Robo Cup

RoboCup ist ein Wettbewerb, bei dem es um Robotik und künstliche Intelligenz geht. Im Wettbewerb gibt es verschiedene Kategorien, in denen die Teams gegeneinander antreten. Als Gemeinsamkeit haben alle Kategorien, dass es darum geht Roboter autonom Fußball spielen zu lassen.

In den meisten Kategorien steht das Entwickeln und Konstruieren der Roboter im Vordergrund. Diese Kategorien unterscheiden sich in der Größe und den Eigenschaften der Roboter, sowie in der Größe der Teams.

Zusätzlich zu den physikalischen Wettbewerben gibt es auch die Simulation League, in der die Programmierung der künstlichen Intelligenz im Vordergrund steht. Hier entfällt die Entwicklung und Konstruktion der Roboter und die Teams fokussieren ihre Anstrengungen auf die Weiterentwicklung der Intelligenz der Spieler (vgl. [1, p. 2f]).

Im Gegensatz zu den Kategorien, bei denen die Robotik im Vordergrund steht, ist die Simulation League in Bezug auf die Aufgabenstellung sehr interessant. In der RoboCup Simulation League werden zwei Teams mit jeweils bis zu elf Spielern simuliert. Die Simulation League gibt es in zwei Varianten: Eine 2D-Version und eine 3D-Version.

In der 2D-Variante beschränkt sich die Simulation auf Blickwinkel und eine simple zweidimensionale Physik. Somit sind hier keine komplexen, realistischen Spielzüge möglich. Das mögliche Spielverhalten entspricht allerdings dem, der einfacheren Robotik-Kategorien.

Die 3D-Version bietet eine komplexere Physikengine, welche entsprechend dreidimensionale Bewegungen des Balls ermöglicht. Dadurch wird das Spielverhalten realistischer. Die RoboCup Simulation League hat jedoch weiterhin die Simulation von Fußballrobotern im Vordergrund.

Technisch basiert die Umsetzung der Simulation League auf einer Client-Server-Architektur (vgl. [1, p. 11]). Der Server kontrolliert die Simulation und Implementiert die Regeln und die Physik. Die Spielerintelligenz wird von den Clients realisiert. Pro Spieler wird ein Client benötigt. Die Clients kommunizieren über UDP/IP mit dem Server und erhalten so Informationen über das Spiel und können den entsprechenden Spieler steuern.

Die Programmierung der Clients ist in verschiedenen Programmiersprachen möglich, wodurch die Gestaltung der Spielerintelligenz sehr flexibel gestaltet. Es ist möglich das Spielverhalten individuell für einzelne Spieler zu gestalten, aber auch das Zusammenspiel der Spieler im Team kann gestaltet werden. Dazu stehen Kommunikationsmechanismen zwischen den Spielern zur Verfügung.

Viele Ansätze aus der RoboCup Simulation League können für die Aufgabenstellung übernommen werden. Zum einen ist die Kommunikation der Spieler untereinander ein interessanter und wichtiger Punkt für die Entwicklung komplexer Spielzüge. Auch die unabhängige Programmierung der Spielerlogik von einer Kontrollinstanz, die Regeln und Physik der Simulation sicherstellt, ist ein guter Ansatz um Flexibilität zu gewährleisten.

## Künstliche Intelligenz

Künstliche Intelligenz beschäftigt sich mit Möglichkeiten Computer zu nutzen um komplexe Probleme zu lösen, die normalerweise menschliche Intelligenz zur Lösung voraussetzen. Als Teilgebiet der Informatik befasst sich die künstliche Intelligenz mit der Entwicklung intelligenter Computersysteme, die ein ähnliches Verhalten aufweisen wie die menschliche Intelligenz (vgl. [2, p. 13f]). Wobei es nicht direkt um den Vergleich mit der menschlichen Denkweise geht, sondern darum mit den nachgebildeten menschlichen Fähigkeiten zur Problemlösung die Leistungsfähigkeit von Computern zu erhöhen.

Das Feld der künstlichen Intelligenz ist breit aufgestellt und umfasst viele verschiedene Einsatzgebiete.

Ansätze der künstlichen Intelligenz sind zum Beispiel im Gebiet des Data Mining zu sehen. Mit der Technik soll die Wissensdarstellung und die Wissensverarbeitung unterstützt werden. Hier werden die Methoden verwendet um Muster und Regeln aus großen Datenmengen zu extrahieren. Diese Methoden können Informationen und Zusammenhänge finden, die für den Menschen so nicht erkennbar wären.

Auch im Bereich der Spracherkennung ist die künstliche Intelligenz vertreten. Hier geht es darum das Gesprochene zu verstehen und zu verarbeiten. Dazu muss der Computer in der Lage sein nicht nur die Syntaktik der Sprache zu verstehen sondern auch die Semantik. Das bedeutet der Computer muss ein ähnliches Verständnis für die Sprache besitzen wie der Mensch um sie zu verstehen. Die Darstellung der Sprache über ihre Grammatik ist hierzu ein anzutreffender Ansatz.

Ein weiteres, sehr bekanntes Beispiel für künstliche Intelligenz sind Computerspiele. In den meisten Computerspielen gibt es computergesteuerte Gegner oder Mitspieler. Diese müssen auf das Verhalten und die Strategie des Spielers reagieren können. Dazu müssen sie eigene Entscheidungen auf der Basis von bestimmten Umgebungswerten ableiten können.

Oft wird der Begriff künstliche Intelligenz auch mit maschinellem Lernen in Verbindung gebracht. Maschinelles Lernen stellt ein Teilgebiet der künstlichen Intelligenz dar, bei dem Computer auf Basis von vorhandenen Daten neues Wissen Generieren. Das kann sowohl das automatisierte Treffen von Entscheidungen sein, als auch die Erkennung von Mustern und Zusammenhängen.

Dies wird in vielen Fällen mittels künstlicher neuronaler Netze versucht. Sie bilden menschliche Strukturen nach und lernen das gewünschte Verhalten durch die Verarbeitung von Beispielen. Ähnlich dem menschlichen Gehirn greifen sie eine Art Erinnerungsvermögen zurück, wenn sie auf ähnliche Problemstellungen treffen (vgl. [2, p. 16]).

Das Thema künstliche Intelligenz hat Relevanz für diese Studienarbeit, da die Spieler basieren auf bekannten Informationen Aktionen ausführen sollen. Diese Informationen können die eigene Position, den Erschöpfungszustand, die Position des Balls, der Mitspieler und der Gegner sein. Aber auch Informationen über den Spielstand oder Beobachtungen über die gegnerische Strategie kommen in Frage. Auf Basis dieser Informationen können weitere Informationen abgeleitet werden und so Schlüsse über den Spielverlauf und die notwendige Strategie getroffen werden. Darauf aufbauend kann der Spieler Aktionen durchführen.

## Zusammenfassung

# Planung / Struktur

## Projektmanagement

Die Projektentwicklung ist in zwei Phasen unterteilt. In der ersten Phase, die während des fünften Semesters stattfindet, werden zunächst eine Planung sowie die passende Architektur und ein Prototype entwickelt. In der zweiten Phase, die während des sechsten Semesters stattfindet, wird eine Implementierung nach der entwickelten Architektur erstellt und getestet.

Das Projekt wird mit einem vier Mann staken Team umgesetzt, aus diesem Grund wird ein guter und regelmäßiger Austausch gepflegt. Hierzu findet wöchentlich ein Meeting des gesamten Teams statt um den aktuellen Vorschritt zu besprechen und neue Aufgaben zu bestimmen. Zur Sicherstellung das in die richtige Richtung gearbeitet wird findet monatlich ein Treffen mit dem Auftraggeber statt. Hierbei wird über den aktuellen Stand berichtet und weitere Ziele diskutiert. (Siehe Anhang XX Meetings-Protokolle)

Aufgrund dieser agilen Softwareentwicklung entstehen nur grobe Zeitpläne und Meilensteine.

## Analyse der Aufgabenstellung

Aus der vorgegebenen Aufgabenstellung werden zunächst die elementaren und zusätzlichen Eigenschaften herausgearbeitet und in Muss-, Kann- und Sollkriterien unterteilt. Die erstellten Kriterien werden nach ihrer Machbarkeit analysiert und eine geeignete Umsetzung ausgewählt.

### Muss-Kriterien

* Simulation eines Fußballspiels
  + Physikalische Korrektheit
  + Einhaltung der Fußballregeln
  + Ausdauersystem für Spieler
  + Kompatibel mit Windows und Linux
* Erstellen einer KI für die Fußballspieler
* Nachträgliches erstellen neuer KIs ermöglichen

### Soll-Kriterien

* Bestrafung der Spieler bei Nichteinhaltung der Regeln
* 3D-Darstellung

### Kann-Kriterien

* Grafischer Editor zur Erstellung von einer KI

### Analyse der Kriterien

Die Anforderung einer physikalisch korrekten 3D Simulation wird über eine Game-Engine realisiert, da diese eine eingebaute Physik-Engine besitzen. In diesem Projekt wird die Unreal Engine von „Epic Games“ verwendet. Die Unreal Engine erfüllt ebenso die Bedingung der Plattformunabhängigkeit, es wird Windows und Linux unterstützt.

Um die KIs der Spieler nachträglich ändern zu können kann eine Skriptsprache verwendet werden, in diesem Projekt wird auf die freie Skriptsprache LUA gesetzt. Um die Skripte den Spielern zuzuordnen oder andere Einstellungen des Teams vorzunehmen wird eine XML Schnittstelle implementiert.

## Architektur

Die Architektur baut auf der Unreal Engine auf und wird aus diesem Grund vollständig in C++ entwickelt. Das Projekt greift auf existierende Technologien wie XML und LUA zurück:



Abbildung 1: Architektur Übersicht

Der wichtigste Teil der Architektur ist die Schnittstelle zwischen den LUA-Skripten und den Robotern. Hierbei müssen 22 Threads für die 22 Spieler mit dem Gamethread synchronisiert werden.

Um die Schnittstelle zwischen den Workern und dem Hauptthread zu ermöglichen wird ein vereinfachtes Remote Procedure Call Verfahren verwendet. Hierzu wird in der Klasse RobotWorker z.B. getPosition() aufgerufen. Der Thread muss angehalten werden, da diese Funktion einen Rückgabeparameter hat. Der gewünschte Funktionsaufruf wird mit insertCall() in die RobotControl übergeben. Sobald der Hauptthread Rechenzeit zur Verfügung hat ruft dieser die Methode tick() auf. Diese Methode arbeitet dann die Queue, der Funktionsaufrufe ab. Wenn ein Aufruf auf eine Antwort wartet wird diese zurück an den Worker mit setReturnValue() gegeben. Die Busy-Waiting Schleife im Thread kann nach Erhalt des Ergebnisses dies an das Skript zurückgeben. Um spätere Erweiterungen leicht zu ermöglichen z.B. andere Skriptsprachen oder eine Implementierung direkt in C++ wird über eine Spezialisierung des RobotWorkers die LUA Schnittstelle hinzugefügt.  
Die 22 Threads müssen in ihrer Leistung beschränkt werden um eine faire Simulation zu ermöglichen. In der LUA Implementierung geschieht dies über die Funktion allowedToRun() diese wertet die bisherige Rechenzeit aus und veranlasst ein Anhalten des Threads um das zehnfache seiner Rechenzeit um seine Auslastung auf 1/10 eines Prozessorkerns zu beschränken. Dies hat zur Folge, dass für alle 22 Spieler 2,2 CPUs benötigt werden. Aufgrund der zusätzlichen Threads für die 3D Engine wird ein Vierkernprozessor benötigt.



Abbildung 2: Architektur als Klassendiagramm

# Umsetzung

## Umsetzung der Simulationsumgebung

### Grundsätzliches

### Regelimplementierung

### Scriptschnittstelle

## Entwicklung der künstlichen Intelligenz

Künstliche Intelligenz beschäftigt sich mit der Nachahmung und Simulation menschlicher, kognitiver Fähigkeiten. Das Ziel ist die effiziente Nachbildung von intelligenter Leistung und menschlichem Verhalten in Computerprogrammen (vgl. [3, p. 17f.]).

Im speziellen Fall hier geht es darum, die Intelligenz und das Verhalten eines Fußballspielers bzw. einer ganzen Fußballmannschaft nachzubilden. Dabei sind Interaktionen mit dem Ball und anderen Spielern ein wichtiger Bestandteil. Andere Spieler teilen sich in gegnerische Spieler und eigene Teammitglieder auf. Es ist notwendig, dass jeder Spieler eine individuelle Strategie verfolgt. Diese Strategie muss abhängig von der Spielposition und dem aktuellen Zustand des Spiels sein. Ziel ist es dass sich die individuellen Strategien der einzelnen Spieler zu einer Strategie für die Mannschaft ergänzen. Da Fußball ein Mannschaftssport ist, ist diese Zusammenarbeit der einzelnen Spieler ein zentraler Punkt für die Entwicklung der künstlichen Intelligenz.

Eine Simulation der Fußballmannschaft kann daher als ein verteiltes, wissensverarbeitendes System angesehen werden. Ein verteiltes System ist dadurch charakterisiert, dass mehrere unabhängige Prozesse eine gemeinsame Aufgabe erfüllen. Dabei besitzen die Prozesse keinen gemeinsamen Speicher, sondern kommunizieren über Nachrichten. Zudem wird oft die Forderung gestellt, dass ein verteiltes System für den Benutzer nicht als solches erkennbar sein darf. Das bedeutet, der Benutzer sieht nur ein einzelnes System und kann die individuellen Systeme nicht erkennen. <TODO: Quellen 🡪 Andrew Tannenbaum, Peter Löhr> Die Umsetzung einer Fußballmannschaft in künstlicher Intelligenz erfüllt im Fall dieser Studienarbeit die allgemeine Definition. Jeder Spieler wird durch einen eigenständigen Prozess repräsentiert. Die Kommunikation mit der Simulationsumgebung erfolgt über entfernte Funktionsaufrufe. Eine Kommunikation der Spieler untereinander ist ebenfalls über entfernte Funktionsaufrufe über die Simulationsumgebung möglich. Es besteht also kein gemeinsamer Speicher. Dass die einzelnen Prozesse ein gemeinsames Ziel verfolgen liegt bei einer Fußballmannschaft auf der Hand. Es geht darum als Team möglichst effektiv zu spielen um das Spiel zu gewinnen. Die Forderung nach der Erkennbarkeit der einzelnen Prozesse ist schwer zu beantworten. Auf der einen Seite ist jeder Spieler klar als Individuum mit individuellem Verhalten zu erkennen. Auf der anderen Seite ist bei korrekter Implementierung der künstlichen Intelligenzen das Verhalten als Mannschaft deutlich zu sehen.

Nach der Betrachtung als verteiltes, wissensverarbeitendes System, muss ein geeigneter Ansatz gefunden werden um dieses umzusetzen. Im Gebiet der computerbasierten Wissensverarbeitung stehen verschiedene Ansätze zur Entwicklung von intelligentem Verhalten zur Verfügung.

Es kann grundsätzlich zwischen expliziter und impliziter Wissensverarbeitung unterschieden werden. Bei der expliziten Wissensverarbeitung müssen konkrete Regeln definiert werden. Diese geben dem System ein Entscheidungsverhalten vor. Die Eingaben werden mit bereits bekannten Fakten und Regeln verarbeitet und so neue Erkenntnisse generiert.

Im Gegensatz dazu kommen implizite Ansätze ohne eine explizite Definition von Regeln aus. Sie sind in der Lage aus Beispielen selbst Regeln zu generieren. Sie benötigen allerdings Beispiele aus denen sie die Regeln erlernen können. Daher eignen sich solche Verfahren besonders gut zu Mustererkennung. Der wichtigste Vertreter für einen solchen Ansatz ist das neuronale Netz. Neuronale Netze bilden die Neuronenstruktur des menschlichen Gehirns nach. Es werden die Neuronen mit ihren Verbindungen, den Synapsen, simuliert. Neuronale Netze und andere implizite Ansätze werden in diesem Dokument nicht weiter betrachtet, da sie für diese Studienarbeit zu weit führen würden. Außerdem ist die Umsetzung der künstlichen Intelligenz eines Fußballspielers auf diese Weise nicht trivial. Es muss zunächst die Frage nach dem Erlernen der Regeln bzw. dem Spielverhalten geklärt werden.

Im Weiteren spielen daher nur noch explizite, wissensverarbeitende Systeme eine Rolle. Ansätze die auf reiner Logik basieren sind wegen der Implementierung problematisch. Durch die implementierte Scriptschnittstelle wird mit LUA-Script eine prozedurale Sprache verwendet. Darin lassen sich logische Aussagen und Regeln nur schwer formulieren. Deklarative Sprachen wie Prolog oder Lisp sind zum Umsetzen von Aussagenlogik deutlich besser geeignet. Das gilt auch für die Prädikatenlogik.

In dieser Studienarbeit wurde mit einfachen Entscheidungsbäumen gearbeitet. Die Entscheidungen basieren auf Daten, die die Spieler von der Simulationsumgebung erhalten. Die Spieler sind zusätzlich in der Lage selbst generierte Daten zu speichern und für Entscheidungen zu nutzen.

Wie im Kapitel Scriptschnittstelle bereits beschrieben, stehen für die Programmierung der künstlichen Intelligenz wichtige Informationen über den aktuellen Spielstatus zur Verfügung. In folgender Tabelle ist eine Übersicht über die von der Scriptschnittstelle angebotenen Funktionen zu sehen. Eine ausführliche Übersicht mit Übergabeparametern und Verwendungshinweisen findet sich im Kapitel Scriptschnittstelle.

|  |  |
| --- | --- |
| Funktion | Beschreibung |
| AllowedToRun | Gibt zurück, ob die künstliche Intelligenz ausgeführt werden darf |
| TimePlayed | Gibt die aktuelle Spielzeit in Prozent zurück |
| IsKickoff | Gibt zurück, ob ein Anstoß stattfindet |
| GetTeamId | Gibt die Team-Id des Spielers zurück |
| HasBall | Gibt zurück, ob der Spieler am Ball ist |
| GetStamina | Gibt die aktuelle Ausdauer des Spielers zurück |
| GetOwnLocation | Gibt die aktuelle Position des Spielers zurück |
| GetBallLocation | Gibt die aktuelle Ballposition zurück, wenn sichtbar |
| GetVisiblePlayers | Gibt Positionen der sichtbaren Spieler zurück |
| GetGoal1Position | Gibt die Position des 1. Tors zurück |
| GetGoal2Position | Gibt die Position des 1. Tors zurück |

Tabelle 1: Passive Funktionen der Scriptschnittstelle

Es ist zu sehen, dass der künstlichen Intelligenz Informationen über das Spiel selbst in Form der aktuellen Spielzeit oder der Information über den Anstoß bereitgestellt werden. Zusätzlich werden auch noch statische Daten wie die Position der Tore angegeben.

Besonders wichtig um Entscheidungen treffen zu können sind die Daten über den Spieler selbst. Dazu gehören seine Teamzugehörigkeit, sein aktueller Ausdauerwert und seine aktuelle Position auf dem Spielfeld. Außerdem werden noch Informationen über vom Spieler gesehene Objekte bereitgestellt. Das sind der Ball und die anderen Spieler. Diese Informationen stehen nur zur Verfügung, wenn sich die Objekte im Sichtfeld des Spielers befinden.

Auf Basis dieser Informationen kann die künstliche Intelligenz weitere Daten wie die Aufstellung der Teammitglieder oder die Bewegungsrichtung des Balls ableiten. Damit ist es möglich auch komplexere Spielentscheidungen zu realisieren. Einige Beispiele für die Umsetzung eines solchen Entscheidungsbaums folgen später in diesem Kapitel.

Abgesehen von den Informationen bietet die Scriptschnittstelle natürlich auch die Möglichkeit Aktionen des Spielers über die künstliche Intelligenz zu steuern. In der nachfolgenden Tabelle sind die zu Verfügung stehenden Funktionen aufgelistet. Die ausführliche Übersicht ist wieder im Kapitel Scriptschnittstelle zu finden.

|  |  |
| --- | --- |
| Funktion | Beschreibung |
| MoveForward | Vorwärtsbewegen des Spielers |
| MoveTo | Spieler an eine bestimmte Position bewegen |
| Rotate | Spieler um seine Achse drehen |
| Kick | Den Ball in eine Richtung kicken |
| StopBall | Den Ball stoppen |
| Speak | Mit anderen Spielern kommunizieren |
| Listen | Hören, ob ein anderer Spieler kommuniziert |

Tabelle 2: Aktive Funktionen der Scriptschnittstelle

Über die integrierte Schnittstelle kann die künstliche Intelligenz Bewegungen des Spielers auslösen. So ist es möglich den Spieler geradeaus oder auch zu einer bestimmten Position zu bewegen. Ebenso ist es möglich den Spieler zu rotieren.

Zudem hat die künstliche Intelligenz die Möglichkeit den Spieler mit dem Ball interagieren zu lassen. Wenn der Ball in der Nähe ist, kann der Ball angenommen und in eine bestimmte Richtung gespielt werden.

Auch die Kommunikation zwischen den Spielern kann gesteuert werden. Der Spieler kann entweder Nachrichten an andere Spieler weitergeben oder aktiv auf Nachrichten anderer Spieler hören. Der Inhalt der Nachrichten kann dabei frei gewählt werden und zum Beispiel taktische Anweisungen enthalten. Nachrichten können von allen Spielern gehört werden. Über diese Funktionen ist die künstliche Intelligenz eines Spielers in der Lage mit der künstlichen Intelligenz eines anderen Spielers zusammenzuarbeiten.

Mit den oben vorgestellten Funktionen ist es möglich Entscheidungsbäume aufzubauen. Es kann sowohl auf Daten über den Spieler, als auch auf allgemeine Daten über den Spielzustand zugegriffen werden. Durch die Verknüpfung dieser Daten kann entschieden werden welche Aktion des Spielers ausgelöst werden sollte.

Der Vorteil eines Entscheidungsbaums liegt darin, dass er einfach verständlich und nachvollziehbar ist. Somit lässt er sich auch verhältnismäßig leicht formulieren. Dieser Vorteil relativiert sich allerdings durch die Tatsache, dass auch Entscheidungsbäume bei steigender Komplexität an Übersichtlichkeit verlieren. Vor allem dadurch, dass hier nicht nur binäre Entscheidungsbäume eingesetzt werden, lassen unüberschaubare, stark verzweigte Bäume zu. Binäre Entscheidungsbäume verzweigen sich an jedem Knoten in genau zwei Richtungen. Dadurch entstehen in der Regel schmale, tiefe Bäume.

Die hier für die Entscheidungen herangezogenen Informationen lassen sich oft nicht binär darstellen. Ein wichtiges Beispiel dafür ist die aktuelle Position des Spielers.

Im Folgenden wird beispielhaft ein Teil eines Entscheidungsbaums vorgestellt, der feststellt, wie der Spieler mit dem Ball interagieren soll. Abbildung 1 zeigt den dazugehörigen Codeausschnitt.

Zunächst wird geprüft, ob der Ball in einer bestimmten Entfernung zum Spieler ist. Der Spieler soll daraufhin den Ball annehmen. Danach wird für jedes Teammitglied in Sichtweite geprüft, ob es näher am gegnerischen Tor steht als der Spieler. Falls es einen Spieler gibt, der näher am generischen Tor befindet, wird die Position des am nächsten zum Tor gelegenen Spielers gespeichert. Diese Position wird für einen späteren Pass benötigt. Die Überlegung dabei ist, dass ein Spieler, der schon näher am gegnerischen Tor steht, nicht mehr so weit laufen muss mit dem Ball. Dabei sinkt das Risiko den Ball zu verlieren und somit steigt die Chance ein Tor zu erzielen.

Wenn ermittelt wird, dass der Spieler bereits derjenige ist, der am nächsten am Tor ist, wird geprüft, ob ein gegnerischer Spieler in unmittelbarer Nähe befindet. Falls das der Fall ist, wird der nächstgelegene Mitspieler für einen Pass gesucht. Dieser hat vermutlich größere Chancen mit dem Ball zum gegnerischen Tor zu kommen als der hier gesteuerte Spieler.

Wenn die obigen Prüfungen ergeben haben, dass der Spieler abspielen soll, wird noch geprüft wie der aktuelle Ausdauerwert des Spielers ist. Ist dieser hoch genug, passt der Spieler den Ball zu seinem Mannschaftskameraden.

Wenn der Spieler nicht abspielen soll, prüft er seine Distanz zum Tor und wagt daraufhin einen Torschuss oder spielt den Ball leicht Richtung Tor um dann nachzulaufen.



Abbildung 3: Beispiel einer künstlichen Intelligenz

Der oben beschriebene Entscheidungsbaum ist verhältnismäßig klein und übersichtlich. Außerdem arbeitet er nur eine einzelne Situation ab. Um die gesamt Intelligenz des Spielers zu beschreiben sind deutlich mehr Prüfungen zu machen und mehr Informationen zu berücksichtigt. Der oben gezeigt Ausschnitt ist deshalb nur als Beispiel für einen möglichen Entscheidungsbaum zu sehen und soll die grundlegenden Möglichkeiten demonstrieren. Dieses Beispiel berücksichtigt zwar schon Interaktionen zwischen den Spielern, es kann aber noch nicht von einer gemeinsamen Strategie oder gar von einer verteilten künstlichen Intelligenz gesprochen werden.

Die Programmierung ist nicht auf die Erstellung von Entscheidungsbäumen beschränkt. Wie zu Beginn des Kapitels beschrieben gibt es verschiedene andere Formen der künstlichen Intelligenz, die zwar im Rahmen dieser Studienarbeit als ungeeignet eingestuft wurden, aber bei Arbeiten mit dem Schwerpunkt auf die Entwicklung einer künstlichen Intelligenz durchaus interessant sind.

# Ausblick

# Fazit

# Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | RoboCup Federation, „RoboCup Soccer Server Users Manual,“ RoboCup Federation, 2003. |
| [2] | Lämmel, U., Cleve, J., Künstliche Intelligenz, München: Hanser, 2012. |
| [3] | Norvig, R., Russel, S., Künstliche Intelligenz - Ein moderner Ansatz - 2. Auflage, München: Pearson Studium, 2004. |



# Abbildungsverzeichnis

* [Abbildung 1: Architektur Übersicht 12](#_Toc422166434)
* [Abbildung 2: Architektur als Klassendiagramm 13](#_Toc422166435)
* [Abbildung 3: Beispiel einer künstlichen Intelligenz 20](#_Toc422166436)

# Abkürzungsverzeichnis

# Glossar