|  |
| --- |
| Unreal Cup  **STUDIENARBEIT**    im Studiengang Informationstechnik  an der DHBW Ravensburg  Campus Friedrichshafen  von  Michael Kekeisen,  Michael Möbius,  Daniel Rapp,  Maximilian Schmitz  Abgabedatum  Bearbeitungszeitraum xx.xx.2014 – yy.yy.2015  Matrikelnummer Matrikelnummer Michael K,  2459628,  Matrikelnummer Daniel,  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

# Abstract

# Inhalt

[Eidesstattliche Erklärung I](#_Toc407532944)

[Kurzfassung II](#_Toc407532945)

[Abstract III](#_Toc407532946)

[Inhalt IV](#_Toc407532947)

[1 Einleitung 1](#_Toc407532948)

[2 Aufgabenstellung 2](#_Toc407532949)

[3 Stand der Technik 3](#_Toc407532950)

[3.1 Einleitung 3](#_Toc407532951)

[3.2 Unreal Engine 4 3](#_Toc407532952)

[3.3 LUA Script 3](#_Toc407532953)

[3.4 Robo Cup 3](#_Toc407532954)

[3.5 Künstliche Intelligenz 4](#_Toc407532955)

[3.6 Zusammenfassung 6](#_Toc407532956)

[4 Planung / Struktur 7](#_Toc407532957)

[4.1 Projektmanagement 7](#_Toc407532958)

[4.2 Analyse der Aufgabenstellung 7](#_Toc407532959)

[4.2.1 Muss-Kriterien 7](#_Toc407532960)

[4.2.2 Soll-Kriterien 8](#_Toc407532961)

[4.2.3 Kann-Kriterien 8](#_Toc407532962)

[4.2.4 Analyse der Kriterien 8](#_Toc407532963)

[4.3 Architektur 9](#_Toc407532964)

[5 Umsetzung 11](#_Toc407532965)

[5.1 Generell 11](#_Toc407532966)

[5.2 Spielfeld 11](#_Toc407532967)

[5.3 Funktionen über LUA Script 11](#_Toc407532968)

[5.4 Interne Umsetzung 11](#_Toc407532969)

[6 Ausblick 12](#_Toc407532970)

[7 Fazit 13](#_Toc407532971)

[Literaturverzeichnis I](#_Toc407532972)

[Abbildungsverzeichnis II](#_Toc407532973)

[Abkürzungsverzeichnis III](#_Toc407532974)

[Glossar IV](#_Toc407532975)

# 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 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 den 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

http://de.wikipedia.org/wiki/Liste\_von\_Spiel-Engines

### 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

<http://de.wikipedia.org/wiki/Unity_%28Spiel-Engine%29>

http://unity3d.com/unity/licenses

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

https://www.unrealengine.com/what-is-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 wird die Engine normalerweise für 19€ pro Monat, ein großer Vorteil der Engine ist allerdings die Tatsache, dass sie Studenten und Hochschulen kostenlos zur Verfügung gestellt wird.

### 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-Ansätze

### 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

https://en.wikipedia.org/wiki/Lua\_%28programming\_language%29

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 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 Wegen 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, wird 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 erstellen 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:



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.



# Umsetzung

## Prototypenentwicklung

### Begründung

Aufgrund der geringen Erfahrung des Teams zum Thema Spieleentwicklung war es wichtig, dass eine Einarbeitungsphase in die Technologien und die Materie stattfinden konnte.  
Aus diesem Grund wurde die Entscheidung getroffen, Teile des fünften Semesters zur Prototypenentwicklung aufzuwenden.

Zu dieser Entwicklung wurde eine grobe, grundlegende Architektur angelegt, die im Laufe des Semesters mit der gesammelten Erfahrung überarbeitet und verbessert werden sollte. Da diese Architektur eher temporärer, veränderlicher Natur war soll sie hier nicht dargestellt werden.

Der Prototyp dient neben der Bereitstellung einer Lernphase auch zur Implementierung der Grundfunktionen, die später mit geringerem Aufwand in die endgültige Version übernommen werden können.

Mit der gesammelten Erfahrung aus dem Prototypen und der so entwickelten Architektur, die in Kapitel <TODO> erläutert wird, sollte nun im sechsten Semester eine endgültige Softwareversion entwickelt werden.

### Umfang

Der Prototyp beinhaltet, wie bereits erwähnt, die Grundfunktionalitäten auf denen die spätere Simulation aufbaut. Dazu gehört neben dem Aufbau des Spielfeldes und dem Einbinden der Lua-Schnittstelle auch die Entwicklung der Basisfunktionen der Spieler, wie zum Beispiel die Implementierung eines Sichtfelds oder das Bewegen zu einer bestimmtem Position auf dem Spielfeld.

## Generell

## Spielfeld

## Funktionen über LUA Script

## Interne Umsetzung

# Ausblick

# Fazit

# Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | RoboCup Federation, „RoboCup Soccer Server Users Manual,“ RoboCup Federation, 2003. |
| [2] | Lämmel, U., Cleve, J., Künstliche Intelligenz, München: Hanser, 2012. |

# Abbildungsverzeichnis

# Abkürzungsverzeichnis

# Glossar