Benutzerhandbuch UnrealCup

# Inhalt

# Erster Start

## Linux

Unter Linux kann die Simulation nach dem Entpacken des Archivs ohne weitere Vorbereitungsschritte mit der mitgelieferten, grundlegenden KI gestartet werden.

Bei der Ausführung im Linux-Labor muss allerdings darauf geachtet werden, dass die korrekten(proprietären) Grafiktreiber verwendet werden.

Zum Starten muss lediglich die ausführbare Datei(UnrealCup/Binaries/Linux/UnrealCup-Linux-Shipping) gestartet werden, möglicherweise muss diese jedoch zunächst ausführbar gemacht werden.  
Beendet wird die Simulation unter Verwendung der Tastenkombination Alt + F4.

## Windows

Unter Windows ist es möglich, dass vor der Ausführung zuerst die UnrealEngine-Daten installiert werden müssen.  
Diese sind unter <PFAD> zu finden.

Gestartet wird die Simulation durch Ausführung von <PFAD EXE>.  
Beendet wird die Simulation, wie auch bei der Linux-Version, mit der Tastenkombination Alt + F4.

# Aufbau der Teams

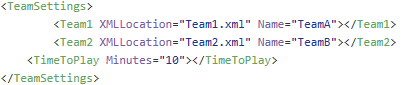
## Konfiguration

Die Teams, die in der Simulation gegeneinander antreten, können mithilfe von XML-Dateien konfiguriert werden. Die Dateien sind unter <PFAD XMLS> zu finden, die vorhandenen Dateien und deren jeweiliger Verwendungszweck wird in der folgenden Tabelle erläutert.  
Die Dateien müssen unter <PFAD> liegen.

|  |  |
| --- | --- |
| Datei | Verwendungszeck |
| Teams.xml | Verknüpfung der Teamnamen mit den jeweiligen Konfigurationsdateien,  Einstellung der Spieldauer |
| TeamX.xml | Konfiguration der einzelnen Spieler eines Teams |
| Scripte | Beinhalten die KI der einzelnen Spieler |

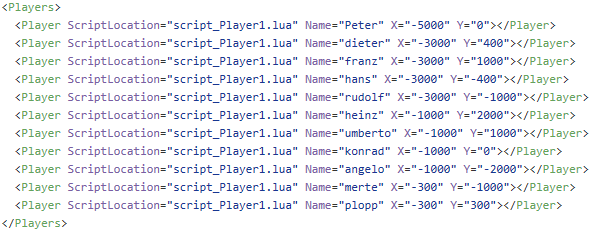
## Die Teams-XML-Datei

In der Teams XML-Datei werden die beiden Teams genannt, mit ihren Konfigurationsdateien verknüpft und die Spieldauer gesetzt. Die Datei ist wie folgt aufgebaut:



## Die Team-XML-Dateien

Die Team-XML-Dateien existieren für jedes Team und konfigurieren die einzelnen Spieler des Teams im Detail. Die Datei ist wie folgt aufgebaut:



X und Y stellen dabei die Startposition des Spielers dar.  
Die Positionen werden als kartesische Koordinaten zum Ursprung (Anstoßpunkt) angegeben.

Es ist außerdem möglich, mehreren oder gar allen Spielern eines Teams dieselbe KI zu übergeben.

# Erstellung von KI-Scripten

## Grundlagen

### Spielablauf

Nach dem Starten des Spiels werden die Spieler auf die Startpositionen, die in den XML-Dateien definiert sind, gesetzt. Daraufhin wird der Anstoß ausgeführt.

Nach jedem gefallenen Tor wird der Punktestand des jeweiligen Teams angepasst, und die Spieler des gegnerischen Teams führen einen Anstoß aus.

Weitere Fußballregeln wie Aus, Abseits oder ähnliches sind nicht implementiert.

Nach dem Ablauf der eingestellten Zeit wird das Spiel eingefroren, der endgültige Punktestand ist sichtbar, das Spiel muss nun manuell beendet werden.

### Anstoß

Die Spieler werden mit dem Befehl „IsKickoff“ über die verschiedenen Anstoßzustände informiert:

|  |  |
| --- | --- |
| Zustand | Erklärung |
| Kein Anstoß | Derzeit muss kein Anstoß ausgeführt werden. Das Spiel wird normal ausgeführt, die Spieler können sich frei bewegen. Fällt ein Tor, wird der Zustand auf „Anstoß wird vorbereitet“ gesetzt. |
| Anstoß wird vorbereitet | Der Anstoß wird vorbereitet. Die Spieler dürfen den Ball nicht berühren und müssen sich in ihre eigene Spielfeldhälfte zurückziehen. Nach TODO Sekunden wird der Ball am Anstoßpunkt platziert und der Anstoß wird mit dem Zustand „Anstoß kann durchgeführt werden“ freigegeben. |
| Anstoß kann durchgeführt werden | Das Team, das den Anstoß ausführen darf, darf den Ball nun berühren. Sobald der Ball berührt wurde, wird der Zustand wieder auf „Kein Anstoß“ gesetzt und alle Spieler dürfen sich wieder frei bewegen. |

Zwar werden die Spieler über den Zustand des Anstoßes informiert, die Ausführung des korrekten Verhaltens für jede Situation muss jedoch in der KI selbst gestaltet werden.  
Bei Nichtbeachtung des Verhaltens wird der Spieler bestraft, indem er aus dem Spiel entfernt wird (TODO)

### Körperliche Eigenschaften der Spieler

Die Spieler sollen in ihren Eigenschaften annähernd menschlichen Fußballspielern gleichen. Das bedeutet, sie besitzen ein eingeschränktes Sichtfeld und keine unerschöpfliche Ausdauer.

Die Spieler können alles in einem FOV von 120 Grad vor sich sehen, die Entfernung spielt dabei keine Rolle. Außerdem ist es den Spieler möglich alles in einem Radius von 5 Metern wahrzunehmen. Die Position der beiden Tore ist den Spielern zu jeder Zeit bekannt, unabhängig davon, ob das Tor sich derzeit im Sichtfeld befindet.

Neben der Fähigkeit, Dinge zu sehen ist es den Spielern auch möglich, Gesprochenes von anderen Spielern zu hören. Auch hierbei spielt die Entfernung keine Rolle, jegliche Gespräche können auf dem gesamten Spielfeld vernommen werden.

Neben diesen Eigenschaften besitzen die Spieler auch eine Ausdauer. Diese Ausdauer wird für Aktionen wie das Kicken oder die Bewegung an eine bestimmte Position verbraucht. Dabei spielt es eine Rolle, wie stark bzw. schnell diese Aktion ausgeführt wird:  
Ein Spieler kann problemlos auf dem Spielfeld auf und ab gehen, beim rennen dieser Strecke wird ihm hingegen die Ausdauer ausgehen.

Besitzt der Spieler keine Ausdauer oder zu wenig Ausdauer, um eine gewisse Aktion auszuführen, so wird diese Aktion mit verringerter, maximal möglicher Stärke bzw. Geschwindigkeit ausgeführt.

Die Ausdauer der Spieler wird kontinuierlich regeneriert, nach einer kurzen Pause besitzt der Spieler also wieder seine volle Ausdauer.

## Funktionen der Spieler

Die Scripte der Spieler werden in der Scriptsprache LUA erstellt. Neben allen standardmäßigen LUA-Funktionen werden die folgenden weiteren Funktionen angeboten:

|  |  |  |
| --- | --- | --- |
| Funktion | Inputwerte | Rückgabewerte / Ergebnisse |
| TimePlayed() | - | Spielzeit in Prozent (0 - 100) |
| GetTeamId() | - | Integer: ID des eigenen Teams (1 oder 2) |
| IsKickoff() | - | Integer: Wert, der den Zustand des Anstoßes beschreibt:   * -1: Kein Anstoß * TODO: Anstoß wird vorbereitet * TODO: TeamId des Teams, das den Anstoß ausführen darf. |
| hasBall() | - | Boolean: Gibt an, ob sich der Ball in Reichweite des Spielers befindet |
| MoveForward(x, y) | TODO | - |
| MoveTo(x, y, TODO) | Zielposition x,  Zielposition y | - |
| Rotate(angle) | Winkel, in dessen Richtung der Spieler sich drehen soll. | Bewegung wird abgebrochen (TODO) |
| Kick(x, y, force) | Zielposition x,  Zielposition y,  Stärke des Kicks | - |
| StopBall() | - | Ball wird gestoppt, wenn er in Reichweite ist, oder wenn er innerhalb der nächsten TODO Sekunden in Reichweite kommt. |
| GetOwnLocation() | - | Integer: x, y und z - Wert der Spielerposition (TODO) |
| GetBallPosition() | - | Integer: x, y und z - Wert der Ballposition (TODO) |
| GetVisiblePlayers() | - | Liste der Spieler, die sich innerhalb des Sichtfelds des Spielers befinden  TODO Sichtfeld  TODO Erklärung Array |
| GetGoal1Position() | - | Integer: x, y und z - Wert des Tors von Team 1 |
| GetGoal2Position() | - | Integer: x, y und z - Wert des Tors von Team 2 |
| GetStamina() | - | Integer: die momentane Ausdauer des Spielers in Prozent (0 - 100) |
| Speak(text) | String mit gewünschter Aussage | - |
| Listen() | - | String: der älteste gesprochene Text |
| AllowedToRun() | - | Boolean: Gibt an, ob das Spiel im Gange ist und die Spieler sich bewegen dürfen. |

Es stehen ihnen sämtliche Funktionen der LUA Bibliothek zur Verfügung, dazu gehören beispielsweise die grundlegenden Mathematikbefehle. Weitere Informationen finden sie unter: http://www.lua.org/manual/5.3/

## Aufbau der Scripte

Um einen problemlosen Ablauf zu gewährleisten, sollte der Script einige grundlegende Voraussetzungen erfüllen:

* Der Script muss von einer while-Schleife umgeben sein, die den Status von AllowedToRun() abfragt.
* Es sollte in jedem Schleifendurchlauf überprüft werden, ob ein Anstoß ansteht.
* TODO weitere?

Exemplarische Scripte sind im Archiv enthalten, es bietet sich die Möglichkeit, diese grundlegenden KIs als Grundlage für eigene Entwicklungen zu verwenden.