- Handbuch -

SoSi

SoccerSimulator

Inhaltsverzeichnis:

1 Einleitung

Willkommen beim SoccerSimulator **SoSi**.
Dies ist ein Handbuch, welches dazu dient, Ihnen die Benutzung dieses Programms näher zu bringen. Steuerungsmöglichkeiten und -funktionen werden ebenso erklärt, wie die grafische Oberfläche, Bedeutung von Anzeigen und Einstellungsmöglichkeiten für die Simulationen (Spieleranzahl, Farben, Spielfeldhintergründe/-anzeigen).
Für Entwickler von KIs gibt es nochmal ein eigenes Kapitel, das sich intensiv damit auseinandersetzt, welche Ausführungsrechte KIs besitzen und worauf man bei der Erstellung dieser genau achten soll.

2 Systemvoraussetzungen

Prozessor	1600 Mhz
Arbeitsspeicher	1 GB
Festplattenspeicher	90 MB
Eingabegeräte	Tastatur und/oder Maus
Medienlesegeräte	CD-ROM Laufwerk, USB-Anschluss
Betriebssysteme	Windows 8, Windows 7, Windows Vista, Linux (Gentoo, Ubuntu)
Software	JRE 1.7, PDF-Reader

3 Installation

3.1 Spielinstallation

Zu Anfang muss die Datei SoSi.zip mit einem passenden Programm in ein Verzeichnis Ihrer Wahl entpackt werden (7zip, WinRar etc.). In diesem Verzeichnis befindet sich die Datei SoSi.jar. Diese muss nun lediglich mittels eines Doppelklicks geöffnet werden und das Programm wird gestartet.
Sollte auf Ihrem PC der Start von .jar-Dateien nicht konfiguriert sein, so können Sie mittels der Kommandozeileneingabe zu dem Ordner, in welchen Sie die Dateien entpackt haben, wechseln und mittels des Befehls "java -jar SoSi.jar" das Programm starten. Nach dem erfolgreichen Start des Programms befinden Sie sich auf dem Startbildschirm des Programms (siehe Der Startbildschirm).

3.2 Kommandozeilenargumente

Das Verhalten des Programms kann mit zusätzlichen Kommandozeilenargumenten beeinflusst werden. Dies erfolg mit der Kommandozeileneingabe, indem die gewünschten Parameter dem Aufrufbefehl angehängt werden. Um beispielsweise die FPS (Frames per Seconds) Anzeige einzublenden, kann der Aufruf des Programms mittels des Befehls "java -jar SoSi.jar -fps" erfolgen. Ebenfalls ist eine Kombination von Parametern möglich. Um z. B. sowohl die FPS Anzeige einzublenden, als auch die FPS Beschränkung aufzuheben, kann der Programmstart mittels des Befehls "java -jar SoSi.jar -fps -fpsUnlimited" erfolgen.

-fps Es wird die Anzeige der FPS-Werte in den Darstellungsfenster aktiviert.

-debugGraphics Es werden Grafiken angezeigt, die die programmintern Grenzen des Spielfeldes ebenso wie die Pfosten anzeigt.

-fpsUnlimited Die Einschränkung der FPS auf 40 wird aufgehoben. Es werden so viele Frames gezeichnet, wie durch die Leistungsfähigkeit des Computers möglich sind.

-noInterpolation Während der Wiedergabe werden ausschließlich die im Voraus berechneten Ticks der Simulation dargestellt. Eine Interpolation von Zwischenpositionen zweier aufeinanderfolgenden Ticks, welche eine flüssigere Darstellung erlaubt, erfolgt jedoch nicht.

-noAnimation Die Animationen in den perspektivischen Ansichten werden ausgeschaltet und die Hintergrundbilder bleiben statisch.

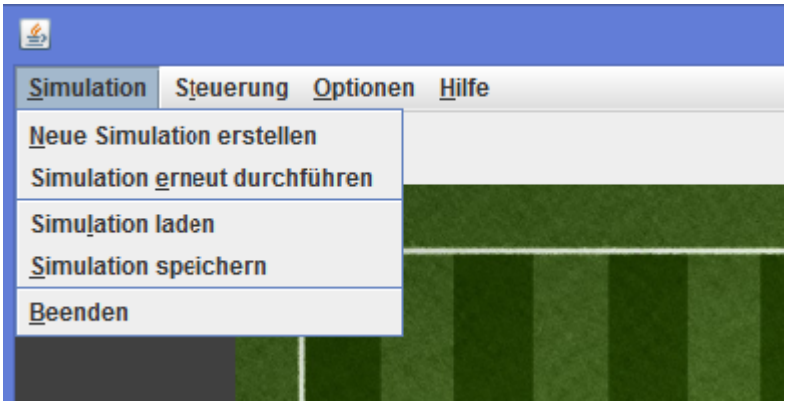
4 Erste Schritte

Dieses Kapitel soll dem Benutzer eine Anleitung zum Starten seiner ersten Simulation geben. Dabei sollen die Standardeinstellungen im "Neue Simulation starten"-Fenster verwendet werden. Die Wiedergabe soll dabei primär mit den Steuerungsbuttons im unteren Bildschirmbereich geregelt werden. Eine Erklärung der Informationsanzeigen am laufenden Programm soll deren Funktionen exemplarisch und anschaulich erläutern.

4.1 Starten einer Simulation

Nach dem erfolgreichen Starten des Programms befinden wir uns auf dem Startbildschirm.

Nun wählen wir in der Menüleiste, die sich am oberen Bildschirmrand befindet, den Menüpunkt Simulation→Neue Simulation erstellen.



Es taucht das Fenster mit dem Titel "Neue Simulation erstellen" auf, das wie folgt aussieht:

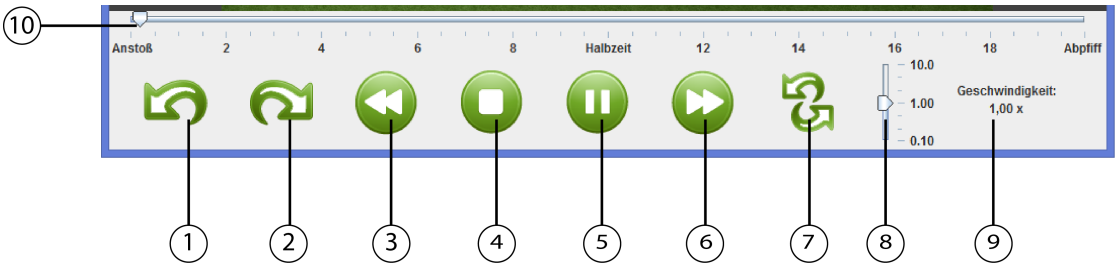
Hierbei sollen keine Einstellungen verändert werden, sondern das Fenster ganz einfach mit dem OK Button bestätigt werden. Es wird nun eine neue Simulation mit folgenden **Spielparametern** erstellt:

- Die beiden antretenden KIs sind die, welche unter "KI Team A" und "KI Team B" angezeigt werden.
- Die Teams bestehen aus jeweils 3 Spielern.
- Team A hat die Farbe blau und Team B hat die Farbe rot.
- Die Spielzeit beträgt 20 Minuten.
- Abseits- und Ausregeln sind deaktiviert.
- Die Soundausgabe ist deaktiviert.
- Die Spielfeldansicht ist Standard (Vogelperspektive) und als Spielfeldart wird ein Rasen verwendet.

Die Simulation und die Wiedergabe starten daraufhin.

4.2 Kontrolle der Wiedergabe

Nun befinden wir uns im Programmzustand, in welchem die Simulation in Normalgeschwindigkeit wiedergegeben wird. Am unteren Bildschirmrand befinden sich Wiedergabesteuerungsmöglichkeiten, die im folgenden Bild markiert sind und nachfolgendend kurz erklärt werden.



① **Simulationsschritt zurück:** Pausiert die Wiedergabe und springt einen Tick zurück. X Klicks sorgen dafür, dass die Wiedergabe X Ticks zurückspringt.

② **Simulationsschritt vor:** Pausiert die Wiedergabe und springt einen Tick vor. X Klicks sorgen dafür, dass die Wiedergabe X Ticks vorspringt.

③ **Rücklauf:** Lässt die Wiedergabe zurücklaufen oder verlangsamt die Vorlaufgeschwindigkeit (bei einer Vorlaufgeschwindigkeit größer 1). Mehrmaliges Klicken beschleunigt den Rücklauf/verringert die Vorlaufgeschwindigkeit immer weiter.

④ **Stop:** Stoppt die aktuelle Wiedergabe und springt an den Anfang der Wiedergabe. Falls die Berechnung der Simulation noch nicht abgeschlossen ist, erscheint ein Dialogfenster, welches fragt, ob die aktuelle Simulation abgebrochen werden soll, womit die bisher berechneten Daten verworfen werden, oder ob die Berechnung weiterhin fortgesetzt werden soll.

⑤ **Pause/Play:** Wenn eine Simulation wiedergegeben wird, dann steht an dieser Stelle der Pause-Button. Wenn dieser betätigt wird, erscheint der Play-Button. Diese Buttons pausieren die Wiedergabe einer Simulation oder nehmen diese wieder auf.

⑥ **Vorlauf:** Lässt die Wiedergabe vorlaufen oder verlangsamt die Rücklaufgeschwindigkeit (bei einer Rücklaufgeschwindigkeit größer 1). Mehrmaliges Klicken beschleunigt den Vorlauf/verringert die Rücklaufgeschwindigkeit immer weiter.

⑦ **Replay:** Springt in der Wiedergabe der Simulation um 5 Sekunden zurück. Der Rücksprung beträgt hierbei nicht zwangsläufig 5 "Spielsekunden", sondern richtet sich nach der Wiedergabegeschwindigkeit. Erfolgt die Wiedergabe mit doppelter Geschwindigkeit, so wird 10 "Spielsekunden" zurückgesprungen. Erfolgt die Wiedergabe mit dreifacher Geschwindigkeit, so wird 15 "Spielsekunden" zurückgesprungen. Dies soll den Effekt haben, dass der Zeitrahmen, welcher zurückgesprungen, beim Anschauen der Simulation für den Benutzer genau 5 Sekunden entspricht.
Während eines Rücklaufs ist die Funktion deaktiviert.

⑧ **Wiedergabegeschwindigkeitsregler:** Hiermit lässt sich die Wiedergabegeschwindigkeit (sei es vor oder zurück) regeln. Je höher der Knopf auf der Leiste ist, desto schneller wird die Geschwindigkeit (beim Vor- und Rücklauf). Es sind nur Werte zwischen 0,10 und 10 (plus oder minus) möglich.

⑨ **Wiedergabegeschwindigkeitsanzeige:** Hier wird die aktuelle Geschwindigkeit angezeigt, die über den Wiedergabegeschwindigkeitsregler eingestellt wurde.

⑩ **Zeitleiste:** Gibt grafisch die gesamte Länge der Simulation wieder. Unter dieser befindet sich eine Zeitleiste, die Zwischenzeiten angibt, ebenso wie die Halbzeit. Die Einfärbung des Hintergrundes gibt an, wie weit die Berechnung der Simulation fortgeschritten ist. Eine Bewegung des Knopfes auf der Zeitleiste ist nur innerhalb des blau eingefärbten Bereiches möglich. Wird der Knopf darüber hinausgezogen, springt er automatisch an den Punkt zurück, der als letztes berechnet worden ist.

4.3 Erklärung der Informationsanzeigen

Team A Zeigt den Name der KI des Teams A an (erstgewählt). Unmittelbar nach dem Programmstart (ohne gestarteter oder geladener Simulation) steht hierbei der Text "Team SoSi".

Torstand Zwischen den Namen der KIs wird der Torstand angegeben. Dabei entspricht die Zahl, welche näher zum Namen einer KI steht, der Toranzahl des Teams.

Team B Zeigt den Namen der KI des Teams B an (zweitgewählt). Unmittelbar nach dem Programmstart (ohne gestarteter oder geladener Simulation) steht hierbei der Text "Team SoSi".

Zeitanzeige Gibt an, wie viel Zeit in der aktuellen Simulation vergangen ist und wie lange insgesamt zu spielen ist. Die gesamte Spielzeit wird in diesem Fall 20:00 (lies: 20 Minuten) betragen.

Geschwindigkeit Gibt an, wie schnell die Wiedergabe (vor oder zurück) momentan abläuft.

5 Die Benutzeroberfläche

In diesem Kapitel beschäftigen wir uns mit der Benutzeroberfläche. Was wird alles angezeigt und womit kann ich was steuern. Dies soll in diesem Kapitel beantwortet werden.

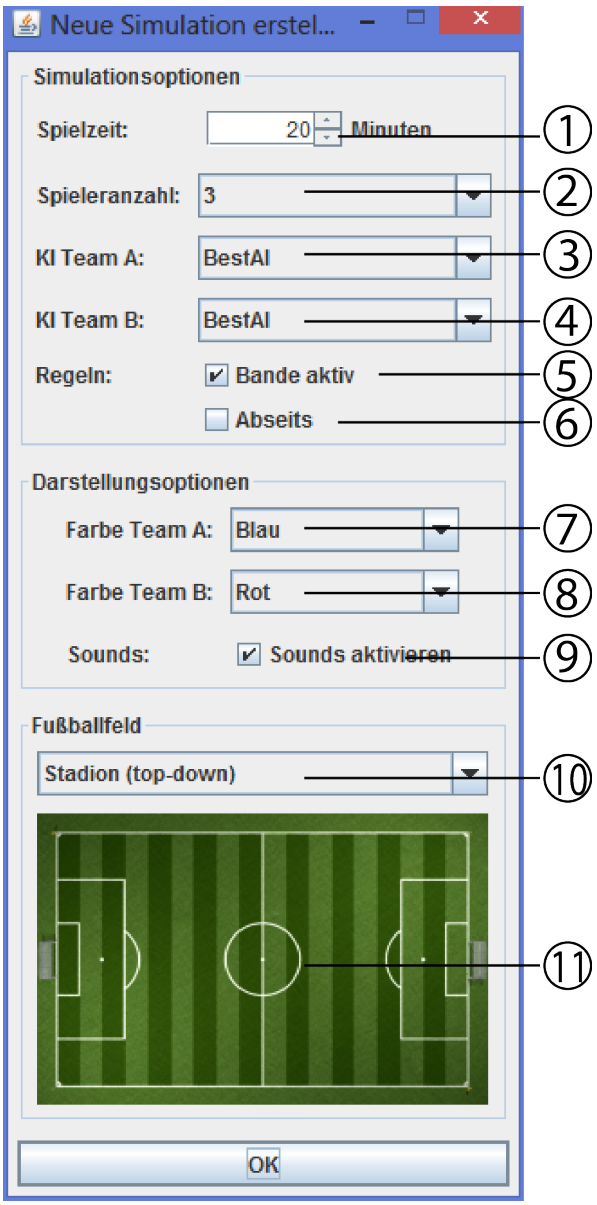
5.1 Der Startbildschirm

Wenn das Programm erfolgreich gestartet worden ist, dann finden Sie sich auf dem Startbildschirm wieder, der wie folgt aussieht.

Am oberen Bildschirmrand finden Sie die Menüleiste, in der sämtliche Einstellungs- und Steuerungsfunktionen für die Simulation bereitgestellt werden.
Direkt unter diesem befindet sich die Informationsanzeige. Ähnlich wie bei Fußballübertragungen im Fernsehen werden hier die antretenden Teams (KIs), der Torstand und die Zeit (vergangene und insgesamte) abgebildet. Im unteren Bildschirmbereich befinden sich Buttons zur bequemen Steuerung der Simulation. Deren genaue Funktion wird an anderer Stelle genau erläutert (siehe Steuerelemente am unteren Bildschirmrand). Zum Programmstart sind sie noch grau, weil noch keine Simulation existiert und man deswegen auch keine Wiedergabe steuern kann. Buttons und Menüeinträge werden ausgegraut, wenn sie zu einem Programmzustand nicht bedienbar sind bzw. deren Funktionen keine logische Konsequenz für das Programm hätten.
Im mittleren Bildschirmbereich befindet sich das Logo der Entwickler. Dieses wird jedoch weichen, sobald eine Simulation gestartet wird. Statt des Logos wird sich hier dann die Anzeige der Simulation mit der Darstellung des Fußballfeldes befinden (siehe Das Hauptfenster mit einer laufenden Wiedergabe).

5.2 Das Fenster "Neue Simulation erstellen"

Wenn nun in der Menüleiste der Menüpunkt "Simulation" ausgewählt wird, öffnet sich ein Untermenü mit verschiedenen Auswahlmöglichkeiten. Der erste Punkt des Untermenüs lautet "Neue Simulation erstellen". Dieser soll nun ausgewählt werden.
Nach der Auswahl schließt sich das Untermenü und es taucht ein neues Fenster mit dem Titel "Neue Simulation erstellen" auf.



- ***Simulationsoptionen***

① **Spielzeit:** Ganz oben in dem Fenster kann man die Simulationsdauer auswählen (wie man der Beschriftung entsprechend erkennen kann). Man kann entweder Zahlen eingeben oder mit Hilfe der Pfeile rechts die gewünschte Zeitdauer angeben. Das Minimum beträgt eine Minute und das Maximum 1440 Minuten (a.k.a. 24 Stunden).

② **Spieleranzahl:** Hier steht ein Auswahlmenü zur Verfügung, in welchem 1 bis 11 Spieler (pro Team) gewählt werden können.

③ **KI Team A:** Hier wird auch ein Auswahlmenü verwendet, das verfügbare KIs aus dem Ordner "AIs" lädt. Für genauere Informationen siehe Kapitel Für KI-Entwickler.
Es sind Standardmäßig bereits mehrere KIs vorhanden, die gewählt werden können. Diese liegen bereits in dem besagten Ordner. Es ist ebenfalls möglich, zwei gleiche KIs gegeneinander antreten zu lassen.

④ **KI Team B:** siehe oben

Regeln: Hier finden sich zwei Kontrollkästchen, die zum einen die Aus- und Abwahl der Abseitsregel und der Ausregel ermöglichen.

⑤ **Ausregel:** Wenn das Kontrollkästchen für die Ausregel aktiviert ist, kann der Ball wie beim normalen Fußballspiel außerhalb des Spielfeldes geraten. Statt eines Einwurfs wird ein Freistoß an der Stelle durchgeführt, an der der Ball das Spielfeld verlassen hat.
Wird das Kontrollkästchen nicht aktiviert, dann prallt der Ball einfach von der Bande ab und bleibt immer innerhalb des Spielfeldes.

⑥ **Abseitsregel:** Wenn das Kontrollkästchen für die Abseitsregel aktiviert wird, dann erfolgt die Simulation mit aktivierter Abseitsregel.
In diesem Fall wird eine vereinfachte Abseitsregeln ohne passives Abseits verwendet. Wenn sich zum Zeitpunkt eines Passes ein Spieler der passenden Mannschaft näher am gegnerischen Tor befindet als alle anderen gegnerischen Spieler und dieser Spieler den Ball, ohne vorherigen weiteren Spielerkontakt, berührt, so wird dies als Foul geahndet. Es wird ein Freistoß für das gegnerische Team an der Stelle durchgeführt, an der sich der foulbegehende Spieler zum Zeitpunkt des Passes befand.

- ***Darstellungsoptionen***

  ⑦ **Farbe Team A** Hier kann mit Hilfe eines Auswahlmenüs die Farbe gewählt werden, mit der die Spieler (hier: Kreise/Ellipsen, je nach Ansicht) angezeigt werden. Dementsprechend ändert sich auch der Hintergrund der Informationsanzeige (KI Name) im oberen Bildschirmbereich.

  ⑧ **Farbe Team B** Analog wie Punkt ⑦.

  ⑨ **Sounds** Mit Hilfe eines Kontrollkästchens kann die Soundausgabe aktiviert oder deaktiviert werden.
  Die Soundausgabe hebt signifikante Simulationszustände während der Wiedergabe hervor, wie z.B. Tor, Foul, Halbzeit, Simulationsende etc.

- ***Fußballfeld***

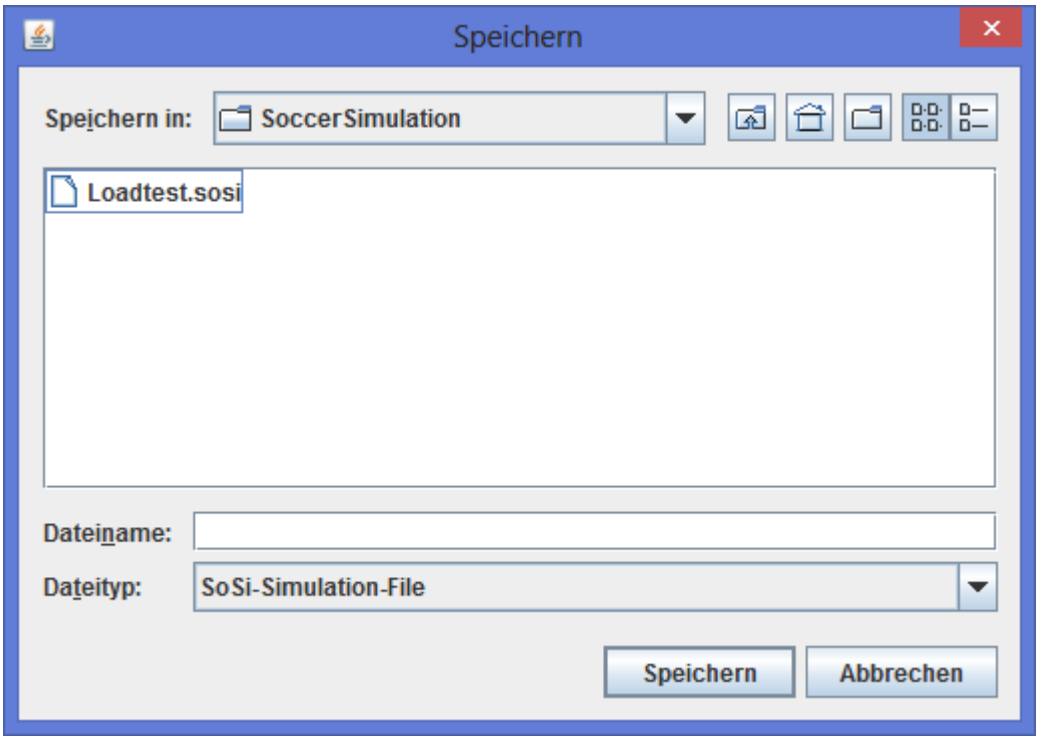
  ⑩ **Auswahlmenü** Über das Auswahlmenü kann man sowohl verschiedene Spielfelder (Stadion, Strand etc.) als auch verschiedene Ansichten (Standard, perspektivisch) wählen.

  Ansichten:

  - *Die Standardansicht bezeichnet die Betrachtung des Spielfeldes aus der Vogelperspektive, direkt über dem Mittelpunkt gelegen.*
  - *Die perspektivische Ansicht, die auch als "perspektivische 2D Sicht" bezeichnet wird, lässt den Betrachter das Spiel aus einer leicht angewinkelten Vogelperspektive betrachten. Dabei werden die Spieler nicht mehr als Kreise dargestellt, sondern entsprechend dem Sichtwinkel als Ellipsen.*

  ⑪ **Vorschau** Abhängig von dem Element, das im Auswahlmenü oben angezeigt wird, ändert sich das Vorschaufenster, um anzuzeigen, was für eine Ansicht gewählt worden ist. So kann die gewünschte Spielfeldanzeige bequemer und schneller identifiziert werden.

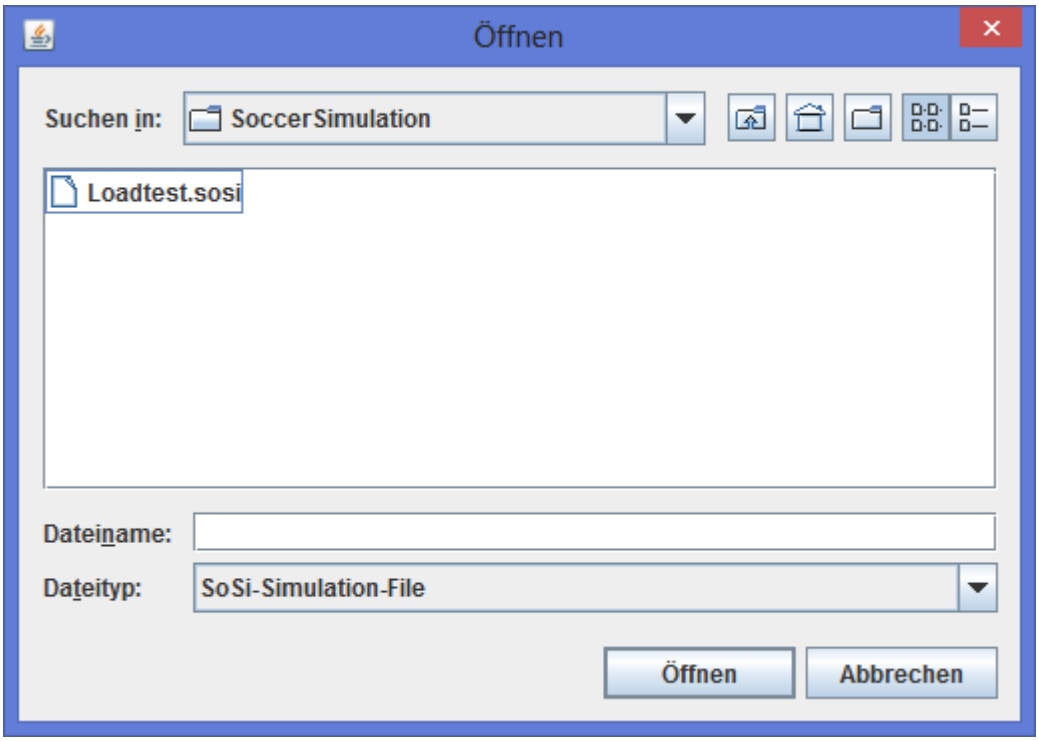
5.3 Das Fenster "Speichern"



Wenn man in der Menüleiste den Punkt "Simulation"→"Simulation speichern" auswählt, erscheint das oben angezeigte Fenster.
Dies ist nur möglich, falls eine neue Simulation gestartet worden ist und deren Berechnung bereits abgeschlossen ist. Sichtbar wird das an der Zeitleiste, deren Hintergrund komplett blau eingefärbt ist. Vorher bleibt der Menüpunkt ausgegraut und kann nicht ausgewählt werden.
Die Auswahl des Speicherortes, in der die Datei abgelegt werden soll, ist an die klassische grafische Navigation und Anzeige angelehnt.
Im Textfeld darf ein beliebiger Name für die Datei gewählt werden, solang dieser vom verwendeten Dateisystem unterstützt wird. (Sollten Ihnen nicht bekannt sein, welche Zeichen Ihr verwendetes Dateisystem unterstützt, so verwenden Sie bitte ausschließlich Buchstaben des Alphabets ohne Umlaute und Ziffern.) Dieser wird automatisch um den Zusatz ".sosi" ergänzt. Dies ist ein programmeigenes Speicherformat, das eigens für das Programm "SoSi" definiert worden ist. Dies wird durch den Dateityp "SoSi-Simulation-File" unten im Fenster nochmals angezeigt.
Die Bestätigung des Speicherdialogs (vorausgesetzt, es wurde eine Dateiname eingetragen) erfolgt mit Hilfe des Buttons "Speichern" im unteren Fensterbereich.
Es können Simulationen mit einer Länge von bis zu 24h gespeichert werden. Falls 24h simuliert werden, sollte vom Benutzer unbedingt dafür gesorgt werden, dass

dies auf auf einem Rechner geschieht, der genug Leistung besitzt (mind. 2,5 Ghz Prozessorleistung, 6 GB Ram, 150 MB Festplattenspeicher). Zusätzlich sollte darauf geachtet werden, dass die JVM mind. 3,5 GB Speicher verwenden kann.
Ein anderer Dateityp kann nicht gewählt werden, weil nur Dateien vom Dateityp "SoSi-Simulation-File" bzw. mit der Endung .sosi vom Programm geladen werden können, was uns zum nächsten Fenster bringt.

5.4 Das Fenster "Öffnen"



Die Auswahl von "Simulation"→"Simulation laden" in der Menüleiste lässt das oben angezeigte Fenster erscheinen. Die Fenster für das Speichern und Laden von Simulationen unterscheiden sich nur durch den Fenstertitel und die Benennung eines Buttons. Der Button im unteren Fensterbereich heißt beim Ladendialog "Öffnen", im Gegensatz zu "Speichern" im Speicherndialog.
Dateien, die nicht das richtige Namensformat (.sosi) haben, werden in diesem Fenster nicht angezeigt, weil ein Laden anderer Dateiformate nicht unterstützt wird.

5.5 Das Fenster "Darstellungsoptionen"

Wenn nun in der Menüleiste Optionen→Darstellungsoptionen gewählt wird, erscheint dieses Fenster.
Das Fenster ist ähnlich aufgebaut wie das "Neue Simulation erstellen"-Fenster. Es fehlt der Unterpunkt "Simulationsoptionen". Die anderen beiden Unterpunkte "Darstellungsoptionen" und "Fußballfeld" sind hier aufgeführt. Mit Hilfe des Fensters sollen Darstellungsoptionen (Farbe der Teams, Ein- und Ausschalten der Sounds und die Auswahl der Fußballfelddarstellungen) der laufenden Wiedergabe angepasst werden können, ohne dafür extra eine neue Simulation erstellen zu müssen.

5.6 Das Fenster "Neues Darstellungsfenster"

Wenn nun im in der Menüleiste Optionen→"Neues Darstellungsfenster" gewählt wird, erscheint dieses Fenster.
Hier wird nur der Unterpunkt "Fußballfeld" angezeigt (im Bezug auf das Fenster "Neue Simulation erstellen"). Zur Funktionsweise siehe Fußballfeld. Eine Bestätigung des Fensters (Klicken des OK-Buttons) hat zur Folge, dass ein neues Fenster erzeugt wird, welches fensterfüllend die Darstellung der Simulation enthält, d.h. es gibt keine Menüleiste im oberen und auch keine Steuerungsbuttons (Steuerelemente am unteren Bildschirmrand) im unteren Bildschirmbereich. Lediglich die Toranzeige wird mittig im oberen Bildschirmbereich angezeigt.
Die Darstellung ist fest an das Hauptfenster gebunden und eine Kontrolle der Wiedergabe erfolgt auch nur in diesem.

5.7 Das Fenster "Über"

Hier werden Informationen zum Programm angezeigt. Der Name des Programm, das Jahr der Erstellung und die Personen, die bei der Implementierung des Programms beteiligt waren.

5.8 Das Fenster "Entwicklerkonsole"

Die Entwicklerkonsole erlaubt Programmierern Ausgaben auf der Konsole zu erzeugen. Durch die implementierten Methoden besteht die Möglichkeit, dass ein Entwickler zu jedem Tick eine Konsolenausgabe erstellt. Somit ist ein vereinfachtes Debuggen möglich, da die Entscheidungen einer KI leichter nachvollzogen werden können. Durch das Durchschalten der Vor- und Zurückbuttons können die einzelnen Ticks betrachtet werden.
ACHTUNG KI-ENTWICKLER: Falls die Simulation beschleunigt wiedergegeben wird, werden einzelne Ticks übersprungen (10-fache Vorlaufgeschwindigkeit: nur jeder 10te Tick wird angezeigt). Dies hat zur Folge, dass die Anzeige "Zuletzt angezeigte Nachricht" immer nur die Nachricht, die die Wiedergabe auch wirklich dargestellt hat, anzeigt.
Es werden auch die derzeitigen Events angezeigt wie z.B. ein Foul.

① **Tick Zurück** Lässt die Simulation einen Tick zurückspringen.

② **Anzeige aktueller Tick/ maximaler Tick** Diese Anzeige gibt zum einen an, in welchem Tick sich die Simulation derzeit befindet und zum anderen zeigt sie die die Anzahl der bereits berechneten Ticks der Simulation an. Das Maximum der Zahlen entspricht daher der maximalen Anzahl an Ticks der Simulation.
Wenn man nun die Tickanzeige anklickt, erscheint dasselbe Fenster wie beim Menüpunkt "Steuerung→Zu Zeitpunkt springen" mit der selben Funktio-

nalität. Für die genaue Erklärung der Funktionsweise siehe Zu Zeitpunkt springen.

③ **Tick Vor** Lässt die Simulation einen Tick vorwärtsspringen.

④ **Konsole von Team A** Hier kann sich der Entwickler z.B. Informationen wie eigens kreierte Debugmessages, die in seiner selbstgeschriebenen KI erstellt wurden, ausgeben. Mit Hilfe derer wird es ihm erleichtert, das Verhalten seiner KI zu untersuchen und anzupassen. Hierfür ist es erforderlich, dass der KI-Entwickler das Interface DebuggingAI implementiert (siehe Debugging mittels DebuggingAI).
Des Weiteren informiert die Konsole über Fehler, welche bei der KI-Ausführung aufgetreten sind. Dies können einfache Warnungen als auch schwerwiegende Exceptions sein.

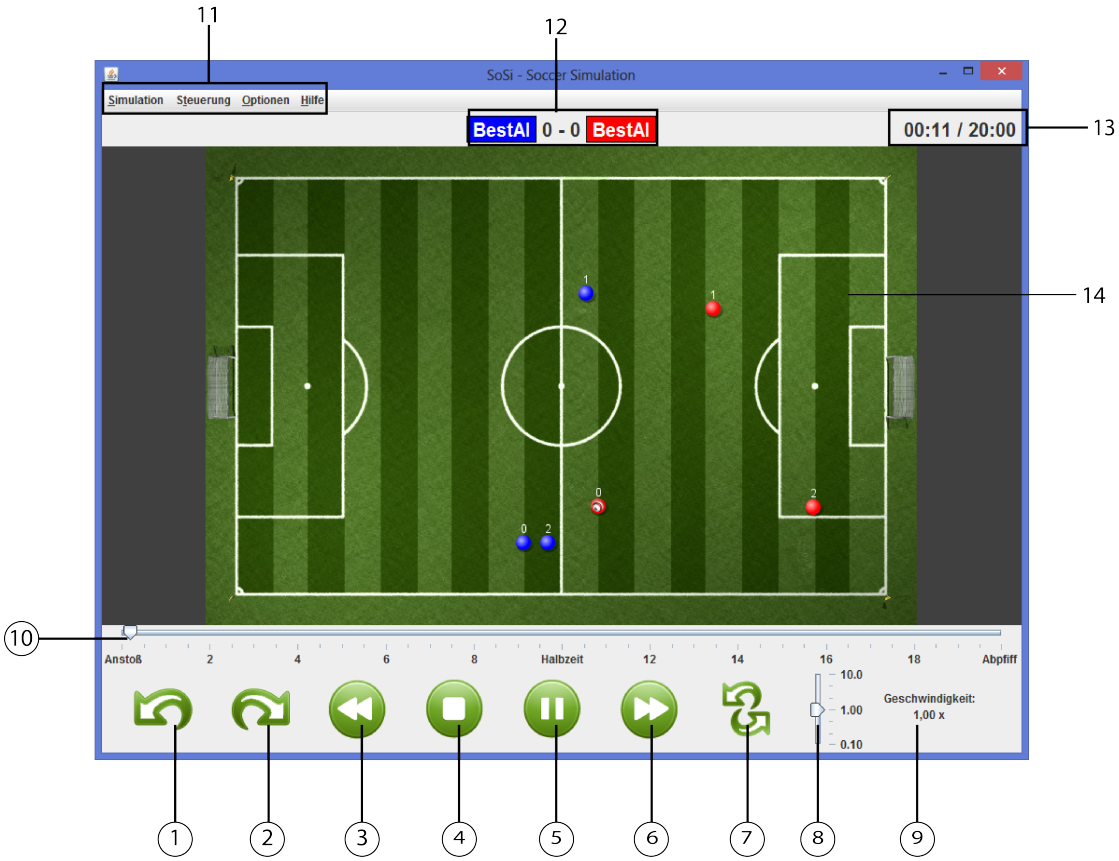
⑤ **Konsole von Team B** Analog Punkt ④.

⑥ **Ereignisanzeige** Zeigt aktuelle TickEvents an. Wenn z.B. ein Tor geschossen wird, es einen Freistoß gibt oder ein Foul begangen worden ist, wird dies hier für jeden einzelnen Tick ausgegeben. Es kann auch möglich sein, dass kein besonderes Tick Event eintritt, dann steht hier lediglich "Das Spiel läuft normal".

6 Die Steuerung

Falls Sie das Kapitel "Erste Schritte" bereits gelesen haben, wird Ihnen vieles bekannt vorkommen. In diesem Kapitel wird auf das ein oder andere Steuerelement genauer eingegangen (insbesondere Vor- und Rücklauf), während auf bereits vollständig beschriebene Funktionen nicht erneut detailiert eingegangen wird.

6.1 Das Hauptfenster mit einer laufenden Wiedergabe



① bis ⑩: Siehe Steuerelemente am unteren Bildschirmrand

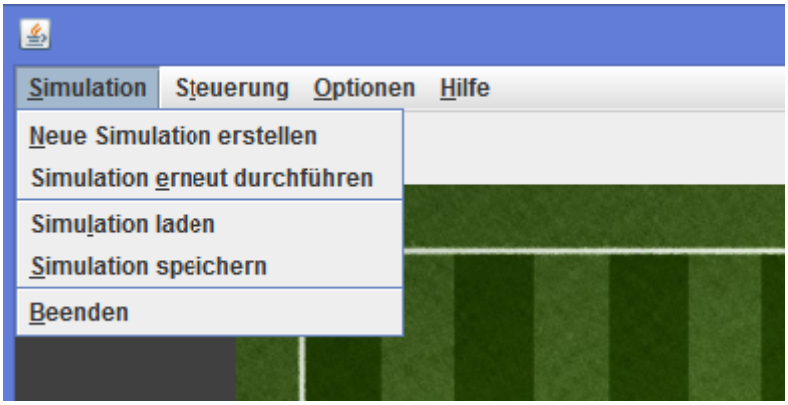
⑪: Siehe Menüleiste

⑫ und ⑬: Siehe Erklärung der Informationsanzeigen

⑭: Die Darstellung der Wiedergabe

6.2 Menüleiste

- **Simulation**



Neue Simulation erstellen Die genaue Beschreibung der Funktionen befindet sich im Kapitel Benutzeroberfläche - Das Fenster "Neue Simulation erstellen".

Simulation erneut durchführen Die aktuelle Simulation wird abgebrochen und eine neue Simulation mit den gleichen Spielparametern startet.

Simulation laden Die genaue Beschreibung der Funktion befindet sich im Kapitel Benutzeroberfläche - Das Fenster "Öffnen".

Simulation speichern Die genaue Beschreibung der Funktion befindet sich im Kapitel Benutzeroberfläche - Das Fenster "Speichern".

Beenden Beendet das Programm ohne weitere Hinweise.

- **Steuerung**

Play Beginnt mit der Wiedergabe oder setzt die pausierte Wiedergabe einer aktuellen Simulation wieder fort.

Pause Pausiert die Wiedergabe der aktuellen Simulation.

Stop Stoppt die Wiedergabe der aktuellen Simulation (springt an den Anfang der Wiedergabe). Falls die Berechnung der Simulation abgeschlossen ist, springt die Wiedergabeposition an den Anfang der Simulation. Ist die Berechnung noch nicht abgeschlossen, erscheinen folgende Optionen: "Zum Anfang springen und Simulationsberechnung fortsetzen" oder "Simulation verwerfen". Sollte die Berechnung, während dieses Dialogfensters offen ist, abgeschlossen werden, so erscheint ein weiteres mit dem Hinweis, dass die Berechnung mittlerweile abgeschlossen ist und deswegen nicht mehr verworfen wird.

Rücklauf Lässt die Wiedergabe zurücklaufen. Zur Auswahl einer Rücklaufgeschwindigkeit erscheint ein Untermenü von Rücklauf, welches die Werte 0.25, 0.50, 0.75, 1.00, 2.00, 5.00 und 10.00 enthält. Dabei bedeuten die Werte, dass mit dem x-fachen der normalen Wiedergabegeschwindigkeit zurückgelaufen wird, falls der entsprechende Menüunterpunkt ausgewählt wird.

Vorlauf Lässt die Wiedergabe beschleunigt vorlaufen. Zur Auswahl der Vorlaufgeschwindigkeit erscheint ein Untermenü von Vorlauf, welches die Werte 0.25, 0.50, 0.75, 1.00, 2.00, 5.00 und 10.00 enthält. Dabei bedeuten die Werte, dass mit dem x-fachen der normalen Wiedergabegeschwindigkeit vorgelaufen wird, falls der entsprechende Menüunterpunkt ausgewählt wird.

Simulationsschritt zurück Die Wiedergabe wird unabhängig von ihrem Zustand pausiert und die Anzeige springt einen Tick zurück. Diese Funktion ermöglicht es, einzelne Simulationsschritte zurückzuspringen, z.B. zur genaueren Betrachtung eines Ereignisses. Zu Beginn einer Simulation ist diese Funktion deaktiviert.

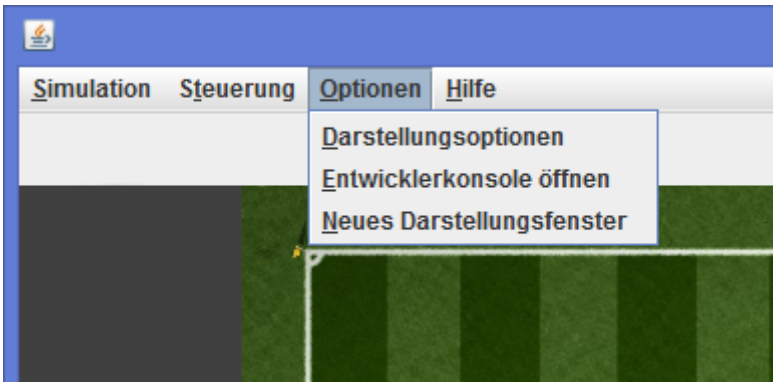
Simulationsschritt vor Die Wiedergabe wird unabhängig von ihrem Zustand pausiert und die Anzeige springt einen Tick vor. Diese Funktion ermöglicht es, einzelne Simulationsschritte vorzuspringen, z.B. zur genaueren Betrachtung eines Ereignisses. Zum Ende einer Simulation ist diese Funktion deaktiviert.

Replay Ermöglicht ein Replay des letzten Geschehens. Dabei springt die Wiedergabeposition eine festgelegte Sekundenzahl vor die aktuelle Wiedergabeposition und führt die Wiedergabe fort. Diese Funktion ist nicht beim Rücklauf verfügbar und beim Vorlauf der Wiedergabe nur bis zu einer bestimmten Geschwindigkeit.

Zu Zeitpunkt springen Es erscheint ein Dialogfenster, das es ermöglicht, zu einer bestimmten Zeit (innerhalb der Wiedergabegrenzen) zu springen, um von diesem Zeitpunkt an die Wiedergabe fortzusetzen. Wird ein Zeitpunkt angegeben, der sich außerhalb der Wiedergabegrenzen befindet, wird die Wiedergabeposition entweder an den Anfang oder an das Ende der Wiedergabe gesetzt.
Es ist möglich, den gewünschten Simulationszeitpunkt auf drei verschiedene Arten und Weisen einzugeben (dabei können jedoch stets ausschließlich ganzzahlige Werte verwendet werden):

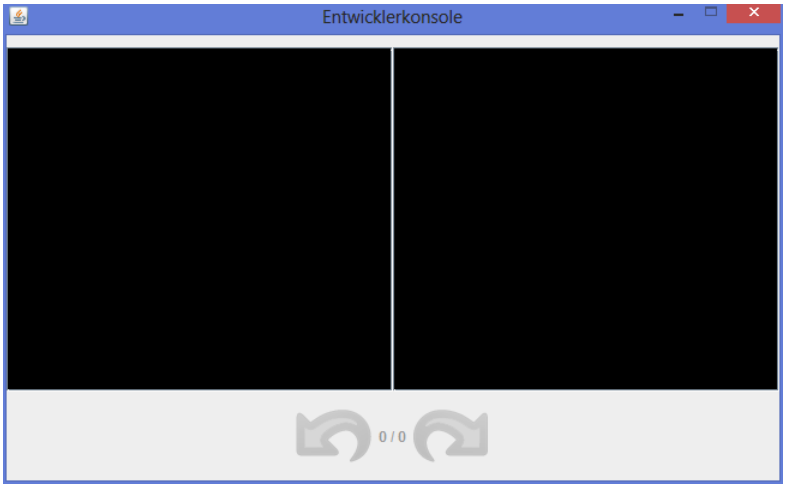
- **Einfache Angabe einer Zahl** Die Zahl wird als Sekunden interpretiert. Wenn mehr als 60 Sekunden angegeben werden, dann wird die Zahl in Minuten und Sekunden umgerechnet und springt auf die entsprechende Stelle in der Wiedergabe.
- **Trennen von zwei Zahlen mit einem Doppelpunkt** X:Y Die Zahl hinter dem Doppelpunkt wird als Sekunden interpretiert und die Zahl vor dem Doppelpunkt wird als Minuten interpretiert. Wenn die Zahl hinter dem Doppelpunkt größer als 60 Sekunden ist, wird sie automatisch in Minuten und Sekunden umgerechnet und zur Zahl vor dem Doppelpunkt addiert. Die Simulation springt nun zu diesem neuen Zeitpunkt.
- **Voranstellung der Raute vor einer einfachen Zahl** #X Die Zahl nach der Raute wird als Tickposition interpretiert und die Simulation springt zur entsprechenden Tickposition in der Wiedergabe.

- **Optionen**



Darstellungsoptionen Es öffnet sich Das Fenster "Darstellungsoptionen", das in einem Menü die Möglichkeit dazu gibt, die aktuelle Wiedergabe anzupassen. Die Veränderung der Teamfarben, Ein- und Ausschalten der Sounds und die Veränderung der Spielfelddarstellungen sind hier möglich.

Entwicklerkonsole öffnen Es öffnet sich ein Fenster mit dem Titel "Entwicklerkonsole".



Genaue Beschreibung siehe Das Fenster "Entwicklerkonsole"

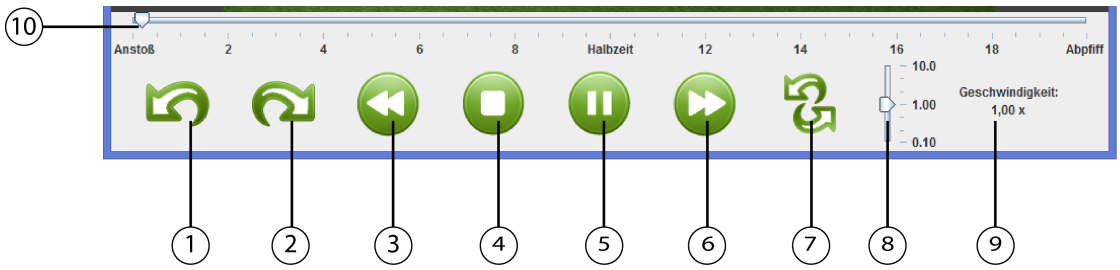
Neues Darstellungsfenster Ermöglicht es ein weiteres Fenster zu öffnen, indem die gleiche Wiedergabe erfolgt, wie im Hauptfenster (Fenster mit Menüleiste). Nach der Auswahl einer passenden Spielfeldansicht schließt sich das Fenster "Neues Darstellungsfenster" und es öffnet sich ein Fenster mit der Wiedergabe der aktuellen Simulation. Dieses Fenster enthält keinerlei Einstellungs- oder Steuerungsmöglichkeiten (Menüleiste, Steuerleiste), sondern lediglich den Torstand und die Darstellung der Simulation.

- **Hilfe**

Benutzerhandbuch Es öffnet sich die PDF-Datei, die Sie gerade lesen.

Über Es öffnet sich ein neues Fenster, welches Auskunft über den Namen des Programms und deren Autoren gibt.

6.3 Steuerelemente am unteren Bildschirmrand



① **Simulationsschritt zurück** Ermöglicht es, einzelne Simulationsschritte zurückzuspringen. Die Wiedergabe der Simulation wird bzw. bleibt dabei pausiert.

② **Simulationsschritt vor** Ermöglicht es, einzelne Simulationsschritte vorzuspringen. Die Wiedergabe der Simulation wird bzw. bleibt dabei pausiert.

③ **Rücklauf** Wiedergabezustände:
Einleitende Erklärung zu den Vorzeichen von "Aktuelle Geschwindigkeit":

- Vorzeichen Minus: Die Wiedergabe läuft zurück, d.h. die Wiedergabe bewegt sich auf den Anfang zu.
- Vorzeichen Plus: Die Wiedergabe läuft vor, d.h. die Wiedergabe bewegt sich auf das Ende zu.

Normale Wiedergabegeschwindigkeit (Aktuelle Geschwindigkeit: +1.00) *Verändert die aktuelle Geschwindigkeit auf -1.00, d.h. die Wiedergabe läuft mit normaler Wiedergabegeschwindigkeit zurück. Durch zusätzliches Klicken des Buttons, kann die Rücklaufgeschwindigkeit in Einserschritten (von -1.00 auf -2.00 auf -3.00 usw.) erhöht werden bis hin zu einem Maximum von -10. Bei der Verwendung des Buttons wird automatisch der Geschwindigkeitsregler (8) mit verschoben.*

Beschleunigter Vorlauf (Aktuelle Geschwindigkeit: größer 1) *Verringert pro Betätigung des Rücklaufbuttons die Vorlaufgeschwindigkeit in Einserschritten auf 1.00. Eine weitere Betätigung führt dazu, das die aktuelle Geschwindigkeit auf -1.00 gesetzt wird. Das weitere Verhalten entspricht dem oben genannten Punkt "Normale Wiedergabegeschwindkeit (Aktuelle Geschwindigkeit: +1.00)"*

Verlangsamter Vorlauf (Aktuelle Geschwindigkeit: größer 0 und kleiner +1) *Verringert bei der ersten Betätigung die aktuelle Wiedergabegeschwindigkeit auf -1.00, setzt den Rücklauf der Wiedergabe also auf die normale*

Wiedergabegeschwindigkeit. Das Verhalten entspricht von diesem Punkt weg dem Verhalten bei "Normale Wiedergabegeschwindigkeit (Aktuelle Geschwindigkeit: +1.00)".

Beschleunigter Rücklauf (Aktuelle Geschwindigkeit: kleiner -1)
Erhöht die Rücklaufgeschwindigkeit mit jeder Betätigung des Rücklaufbuttons um 1 bis das Maximum 10 erreicht ist.

Verlangsamter Rücklauf (Aktuelle Geschwindigkeit: größer -1 und kleiner 0)
Verringert bei der ersten Betätigung die aktuelle Wiedergabegeschwindigkeit auf -1.00, setzt den Rücklauf der Wiedergabe also auf die normale Wiedergabegeschwindigkeit. Das Verhalten entspricht von diesem Punkt weg dem Verhalten bei "Normale Wiedergabegeschwindigkeit (Aktuelle Geschwindigkeit: +1.00)".

④ **Stop** Stoppt die Wiedergabe der aktuellen Simulation. Falls die Berechnung der Simulation abgeschlossen ist, springt die Wiedergabeposition an den Anfang der Simulation. Ist die Berechnung noch nicht abgeschlossen, erscheinen folgende Optionen: "Zum Anfang springen und Simulationsberechnung fortsetzen" oder "Simulation verwerfen". Falls während der Beantwortung der Frage die Simulation bereits fertig berechnet worden ist, wird die Simulation nicht verworfen, sondern lediglich zum Anfang der Simulation gesprungen.

⑤ **Start / Pause** Beginnt mit der Wiedergabe, oder setzt eine Wiedergabe der aktuellen Simulation fort. Falls eine Wiedergabe erfolgt, verschwindet der Play-Button und wird durch den Pause-Button ersetzt. Wenn die Wiedergabe pausiert wird, wird der Pause-Button durch den Play-Button ersetzt.

⑥ **Vorlauf** Einleitende Erklärung zu den Vorzeichen von "Aktuelle Geschwindigkeit":

- Vorzeichen Minus: Die Wiedergabe läuft zurück, d.h. die Wiedergabe bewegt sich auf den Anfang zu.
- Vorzeichen Plus: Die Wiedergabe läuft vor, d.h. die Wiedergabe bewegt sich auf das Ende zu.

Normale Wiedergabegeschwindigkeit (Aktuelle Geschwindigkeit: +1.00)
Lässt die Wiedergabe der aktuellen Simulation mit Vorlaufgeschwindigkeit laufen. Durch zusätzliches Klicken des Buttons, kann die Vorlaufgeschwindigkeit um ein Vielfaches der Normalwiedergabegeschwindigkeit erhöht werden bis hin zu einem Maximum von 10, was der 10-fachen der normalen Simulationsgeschwindigkeit entspricht. Die Erhöhung findet dabei in Einserschritten statt (z.B. von 1.00 auf 2.00 auf 3.00 usw.). Bei der Verwendung wird automatisch der Geschwindigkeitsregler (8) mit verschoben.

Beschleunigter Vorlauf (Aktuelle Geschwindigkeit: größer 1) *Erhöht die Vorlaufgeschwindigkeit mit jeder Betätigung des Vorlaufbuttons um 1 bis das Maximum 10 erreicht ist.*

Verlangsamter Vorlauf (Aktuelle Geschwindigkeit: größer 0 und kleiner 1) *Erhöht bei der ersten Betätigung die aktuelle Wiedergabegeschwindigkeit auf +1.00, setzt die Wiedergabe also auf die normale Wiedergabegeschwindigkeit. Das Verhalten entspricht von diesem Punkt weg dem Verhalten bei "Normale Wiedergabegeschwindigkeit (Aktuelle Geschwindigkeit: +1)".*

Beschleunigter Rücklauf (Aktuelle Geschwindigkeit: kleiner -1) *Verringert die Rücklaufgeschwindigkeit mit jeder Betätigung des Vorlaufbuttons um 1 bis ein "Aktuelle Geschwindigkeit" von -1.00 erreicht ist. Die nächste Betätigung lässt die aktuelle Geschwindigkeit auf +1.00 springen, was der normalen Wiedergabegeschwindigkeit entspricht. Das weitere Wiedergabeverhalten entspricht "Normale Wiedergabegeschwindigkeit (Aktuelle Geschwindigkeit: +1.00)" dieses Abschnittes.*

Verlangsamter Rücklauf (Aktuelle Geschwindigkeit: größer -1 und kleiner 0) *Erhöht bei der ersten Betätigung die aktuelle Wiedergabegeschwindigkeit auf +1.00, setzt die Wiedergabe also auf die normale Wiedergabegeschwindigkeit. Das Verhalten entspricht von diesem Punkt weg dem Verhalten bei "Normale Wiedergabegeschwindigkeit (Aktuelle Geschwindigkeit: +1.00)".*

⑦ **Replay** Ermöglicht ein Replay der letzen Spielsekunden. Dabei springt die Wiedergabeposition eine festgelegten Sekundenzahl vor die aktuelle Wiedergabeposition und führt die Wiedergabe dort fort. Diese Funktion ist nicht beim Rücklauf verfügbar und beim Vorlauf der Wiedergabe nur bis zur 5-fachen Geschwindigkeit.

⑧ **Geschwindigkeitsregler** Dient zur Anpassung der Vor- und Rücklaufgeschwindigkeit.

⑨ **Aktuelle Geschwindigkeit** Hier wird die aktuelle Abspielgeschwindigkeit angezeigt.

⑩ **Zeitleiste** Die Zeitleiste zeigt die aktuelle Wiedergabeposition und die Simulationsposition an. Die Darstellung der aktuellen Wiedergabeposition wird mit Hilfe eines kleinen grauen Knopfs, der sich auf der Zeitleiste bewegt, angezeigt. Die Simulationsposition wird mit Hilfe der Einfärbung der Zeitleiste angezeigt. Ein Ziehen des grauen Knopfes über den berechneten Bereich des Zeitleiste

ist nicht möglich. Die Wiedergabe springt an den letzen Punkt, der berechnet worden ist zurück und führt dort die Wiedergabe fort.

6.4 Hotkeys

- ***Menüleiste:*** Hier muss zuerst der darüberliegende Menüpunkt angewählt werden (mit einer Tastenkombination), bevor der darunterliegende mit einer entsprechende Tastenkombination aktiviert oder gewählt werden kann usw.

Simulation	ALT + S
Neue Simulation erstellen	ALT + N
Simulation erneut durchführen	ALT + E
Simulation laden	ALT + L
Simulation speichern	ALT + S
Beenden	ALT + B
Steuerung	ALT + T
Play	ALT + P
Pause	ALT + P
Stop	ALT + S
Rücklauf	ALT + C
Vorlauf	ALT + V
Simulationsschritt vor	ALT + U
Simulationsschritt zurück	ALT + O
Replay	ALT + R
Zu Zeitpunkt springen	ALT + Z
Optionen	ALT + O
Darstellungsoptionen	ALT + D
Entwicklerkonsole	ALT + E
Neues Darstellungsfenster	ALT + N
Hilfe	ALT + H

Benutzerhandbuch	ALT + H
Über	ALT + B

- ***Steuerungsbuttons:***

Simulationsschritt zurück	ALT + linke Pfeiltaste
Simulationsschritt vor	ALT + rechte Pfeiltaste
Rücklauf	ALT + B
Stop	ALT + . (Punkt)
Play/Pause	ALT + P
Vorlauf	ALT + F
Replay	ALT + R

7 Für KI-Entwickler

7.1 Schreiben einer KI

Dieses Kapitel beschäftigt sich mit dem Schreiben eigener KIs.
Um eine eigene KI zu schreiben, wird zunächst das Paket mit den vom Programm vorgegeben Schnittstellen benötigt, welches im Ordner 'Interfaces' zu finden ist.

7.1.1 Einführung

Die wohl beste Übung für die Entwicklung einer eigenen KI ist, die wichtigsten Methoden anhand eines Beispiels kennenzulernen. Deshalb wird das Schreiben einer neuen KI auf den nächsten Seiten Schritt für Schritt erläutert.
Als Tutorial KI wird eine einfache KI implementiert, welche seine Spieler, bei jedem Anstoß und Freistoß, am Ball platziert. Während dem freien Spiel wird sie versuchen den Ballbesitz zu erlangen. Sofern die KI in Ballbesitz ist, wird sie sofort versuchen auf das gegnerische Tor zu schießen.
Als verwendete Entwicklungsumgebung in diesem Tutorial wird Eclipse genutzt. Außerdem ist das JDK 1.7 erforderlich.

7.1.2 MyAI Klasse erstellen

Als erstes wird eine neue Klasse **MyAI** im package ***my.ai*** erstellt. Diese Klasse implementiert nun die vorgegebene Schnittstelle **AI** und überschreibt deren Methoden.
Das Ganze sollte in etwa so aussehen:

```
1  package my.ai;
2
3  import sep.football.AI;
4  import sep.football.FreePlayActionHandler;
5  import sep.football.GameInformation;
6  import sep.football.KickActionHandler;
7  import sep.football.TickInformation;
8
9  public class MyAI implements AI {
10
11     @Override
12     public void kickOff(GameInformation game, TickInformation tick,
13             KickActionHandler actionHandler) {
14
15     }
16
17     @Override
18     public void freeKick(GameInformation game, TickInformation tick,
19             KickActionHandler actionHandler) {
20
21     }
22
23     @Override
24     public void freePlay(GameInformation game, TickInformation tick,
25             FreePlayActionHandler actionHandler) {
26
27     }
28
29 }
```

Diese überschriebenen Methoden stellen das Grundgerüst jeder KI dar und werden nur zu bestimmten Spielsituationen aufgerufen.

- *kickOff*: Wird beim Anstoß aufgerufen
- *freeKick*: Wird bei jedem Foul, sowie bei Einwürfen und Ecken aufgerufen
- *freePlay*: Wird während dem freien Spielen aufgerufen

Ein einzelner Berechnungsschritt des Programms ist als ein Tick definiert. Zu jedem solchen Tick werden unter anderem diese Methoden, abhängig von der jeweiligen Spielsituation, aufgerufen. Dies ermöglicht es sofort auf Änderungen reagieren zu können.

7.1.3 Spieler platzieren

Alle eigenen Spieler sollen nun bei jedem An- und Freistoß direkt am Ball platziert werden. Sollte der Gegner gerade in Ballbesitz sein, werden die Spieler automatisch

um einen einzuhaltenden Mindestabstand (von der spielinternen Physik) versetzt.

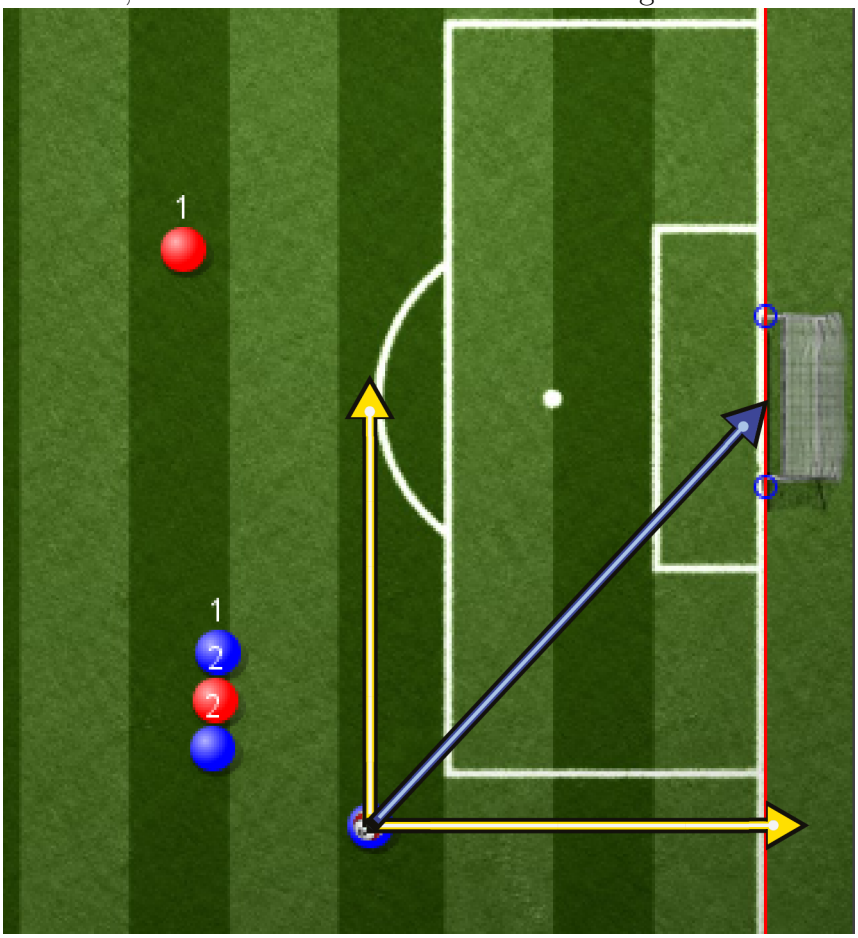
```java
package my.ai;

import sep.football.AI;
import sep.football.FreePlayActionHandler;
import sep.football.GameInformation;
import sep.football.KickActionHandler;
import sep.football.TickInformation;

public class MyAI implements AI {

    @Override
    public void kickOff(GameInformation game, TickInformation tick,
            KickActionHandler actionHandler) {

      placePlayers(tick, actionHandler);

    }

    @Override
    public void freeKick(GameInformation game, TickInformation tick,
            KickActionHandler actionHandler) {

        placePlayers(tick, actionHandler);

    }

    @Override
    public void freePlay(GameInformation game, TickInformation tick,
            FreePlayActionHandler actionHandler) {

    }

    /**
     * Platziert jeden Spieler des eigenen Teams auf der Position des Balls.
     *
     * @param tick
     *            - Eine Referenz auf die aktuelle TickInformation.
     * @param actionHandler
     *            - Eine Referenz auf den aktuellen KickActionHandler.
     */
    private void placePlayers(TickInformation tick,
            KickActionHandler actionHandler) {

        // Alle Spielerpositionen des eigenen Teams
        List<Position> myPlayerPositions = tick.getPlayerPositions();

        // Die Position des Balls
        Position ballPosition = tick.getBallPosition();

        // Die Anzahl der eigen Spieler
        int numberPlayers = myPlayerPositions.size();

        // Platziert jeden eigenen Spieler auf die Position des Balls
        for (int playerID = 0; playerID < numberPlayers; ++playerID) {
            actionHandler.placePlayer(playerID, ballPosition);
        }

    }

}
```

7.1.4 Schuss aufs Tor

Um auf das gegnerische Tor zu schießen, ist ein Aufruf der *kickBall(int playerId, Position direction, double strength)* Methode eines **ActionHandler** nötig. Diese erwartet die folgenden drei Parameter:

- *playerId*: Die ID des schießenden Spielers. Sie ergibt sich aus dem Index in der Positionsliste (Achtung! Listen starten mit Index 0)
- *direction*: Eine relative Positionsangabe, in die der Ball gespielt werden soll.
- *strength*: Die Schussstärke, welche im Bereich]0;1] liegen muss. Je stärker der Schuss ist, desto wahrscheinlicher ist es, dass der Ball sein Ziel verfehlt.

Die Schussrichtung wird abhängig von der Position des schießenden Spielers angegeben, wobei die Position dieses Spielers der Nullpunkt ist. Hierbei ist zu beachten, dass die Y-Koordinate nach oben negativ ist.



Eine entsprechende Java Methode würde in etwa so aussehen:

```
    /**
     * Erstellt eine relative Positionsangabe.
     */
    private Position getRelativePosition(Position fromPosition,
            Position toPosition) {
        return new MyPosition(toPosition.getX() - fromPosition.getX(),
                toPosition.getY() - fromPosition.getY());
    }
```

Mit allen Neuerungen, inklusive einer Methode, welche den Ball aufs gegnerische Tor schießt:

```
package my.ai;

import java.util.List;

import sep.football.AI;
import sep.football.ActionHandler;
import sep.football.FreePlayActionHandler;
import sep.football.GameInformation;
import sep.football.KickActionHandler;
import sep.football.Position;
import sep.football.TickInformation;

public class MyAI implements AI {

    private static final double DEFAULT_KICK_STRENGTH = 1.0;

    public MyAI() {

    }

    @Override
    public void kickOff(GameInformation game, TickInformation tick,
            KickActionHandler actionHandler) {
        placePlayers(tick, actionHandler);
        shootBall(tick, game, actionHandler);
    }

    @Override
    public void freeKick(GameInformation game, TickInformation tick,
            KickActionHandler actionHandler) {
        placePlayers(tick, actionHandler);
        shootBall(tick, game, actionHandler);
    }

    @Override
    public void freePlay(GameInformation game, TickInformation tick,
            FreePlayActionHandler actionHandler) {
    }

    /**
     * Platziert jeden Spieler des eigenen Teams auf der Position des Balls.
     *
     * @param tick
     *            - Eine Referenz auf die aktuelle TickInformation.
     * @param actionHandler
     *            - Eine Referenz auf den aktuellen KickActionHandler.
     */
```

```
48      private void placePlayers(TickInformation tick,
49              KickActionHandler actionHandler) {
50
51          // Alle Spielerpositionen des eigenen Teams
52          List<Position> myPlayerPositions = tick.getPlayerPositions();
53
54          // Die Position des Balls
55          Position ballPosition = tick.getBallPosition();
56
57          // Die Anzahl der eigen Spieler
58          int numberPlayers = myPlayerPositions.size();
59
60          // Platziert jeden eigen Spieler auf die Position des Balls
61          for (int playerID = 0; playerID < numberPlayers; ++playerID) {
62              actionHandler.placePlayer(playerID, ballPosition);
63          }
64
65      }
66
67      /**
68       * Erstellt eine relative Positionsangabe.
69       */
70      private Position getRelativePosition(Position fromPosition,
71              Position toPosition) {
72          return new MyPosition(toPosition.getX() - fromPosition.getX(),
73                  toPosition.getY() - fromPosition.getY());
74      }
75
76      /**
77       * Schießt den Ball aufs gegnerische Tor, sofern die eigene Mannschaft in
78       * Ballbesitz ist.
79       */
80      private void shootBall(TickInformation tick, GameInformation game,
81              ActionHandler actionHandler) {
82          boolean isInBallPossession = tick.hasTeamBall(true);
83          if (isInBallPossession) {
84              int strikerID = tick.getPlayerWithBall();
85              Position striker = tick.getPlayerPositions().get(strikerID);
86
87              double xPosLeftGoal = 0.0;
88              double xPosRightGoal = game.getFieldLength();
89              double yPosGoal = game.getFieldWidth() / 2;
90
91              double xPosOpponentGoal = tick.isPlayingOnTheLeft() ? xPosRightGoal
92                      : xPosLeftGoal;
93              double yPosOpponentGoal = yPosGoal;
94
95              Position shootingPosition = getRelativePosition(striker,
96                      new MyPosition(xPosOpponentGoal, yPosOpponentGoal));
97              actionHandler.kickBall(strikerID, shootingPosition,
98                      DEFAULT_KICK_STRENGTH);
99          }
100     }
101
102     /**
103      * Eine Position auf dem Spielfeld.
104      */
105     private class MyPosition implements Position {
106
107         private double x;
108         private double y;
109
```

```
110         public MyPosition(double x, double y) {
111             this.x = x;
112             this.y = y;
113         }
114 
115         @Override
116         public double getX() {
117             return this.x;
118         }
119 
120         @Override
121         public double getY() {
122             return this.y;
123         }
124 
125     }
126 
127 }
```

Als letztes wird noch die *freePlay(...)* Methode fertiggestellt.
Hier soll die KI versuchen den Ballbesitz zu erlangen. Insofern sie in Ballbesitz ist, wird sie sofort auf das gegnerische Tor schießen.
Sollte sie nicht in Ballbesitz sein, wird jeder Spieler in Richtung des Balls laufen. Sollte der Ball innerhalb des Aktionsradius (welcher 1.0 Meter beträgt) eines Spielers sein, wird er versuchen den Ball zu gewinnen. Dies geschieht mittels der *acquireBallControl(int playerId, double determination)* Methode. Die 'determination' gibt die Härte des Tacklings (Versuch den Ballbesitz zu erringen) an. Dieser Wert muss im Intervall]0;1] liegen. Ein hoher Wert erhöht zwar die Wahrscheinlichkeit, dass der Spieler den Ball erlangt, erhöht aber gleichzeitig die Wahrscheinlichkeit eines Foulspiels bei einem Misserfolg.
Sollte ein Versuch scheitern, wird der eigene Spieler zudem, abhängig von der 'determination', für einige Ticks blockiert und kann in dieser Zeit keine Aktionen ausführen.
Die Bewegungsänderung erfordert wie beim Schießen eine relative Positionsangabe.
Somit wäre die KI fertig und bereit für ihren Einsatz.

```
1  package my.ai;
2 
3  import java.util.List;
4 
5  import sep.football.AI;
6  import sep.football.ActionHandler;
7  import sep.football.FreePlayActionHandler;
8  import sep.football.GameInformation;
9  import sep.football.KickActionHandler;
10 import sep.football.Position;
11 import sep.football.TickInformation;
12 
13 public class MyAI implements AI {
14 
15     private static final double DEFAULT_KICK_STRENGTH = 1.0;
16     private static final double DEFAULT_PLAYER_SPEED = 1.0;
17     private static final double DEFAULT_AQUIRING_DISTANCE = 1.0;
18     private static final double DEFAULT_AQUIRING_DETERMINATION = 1.0;
```

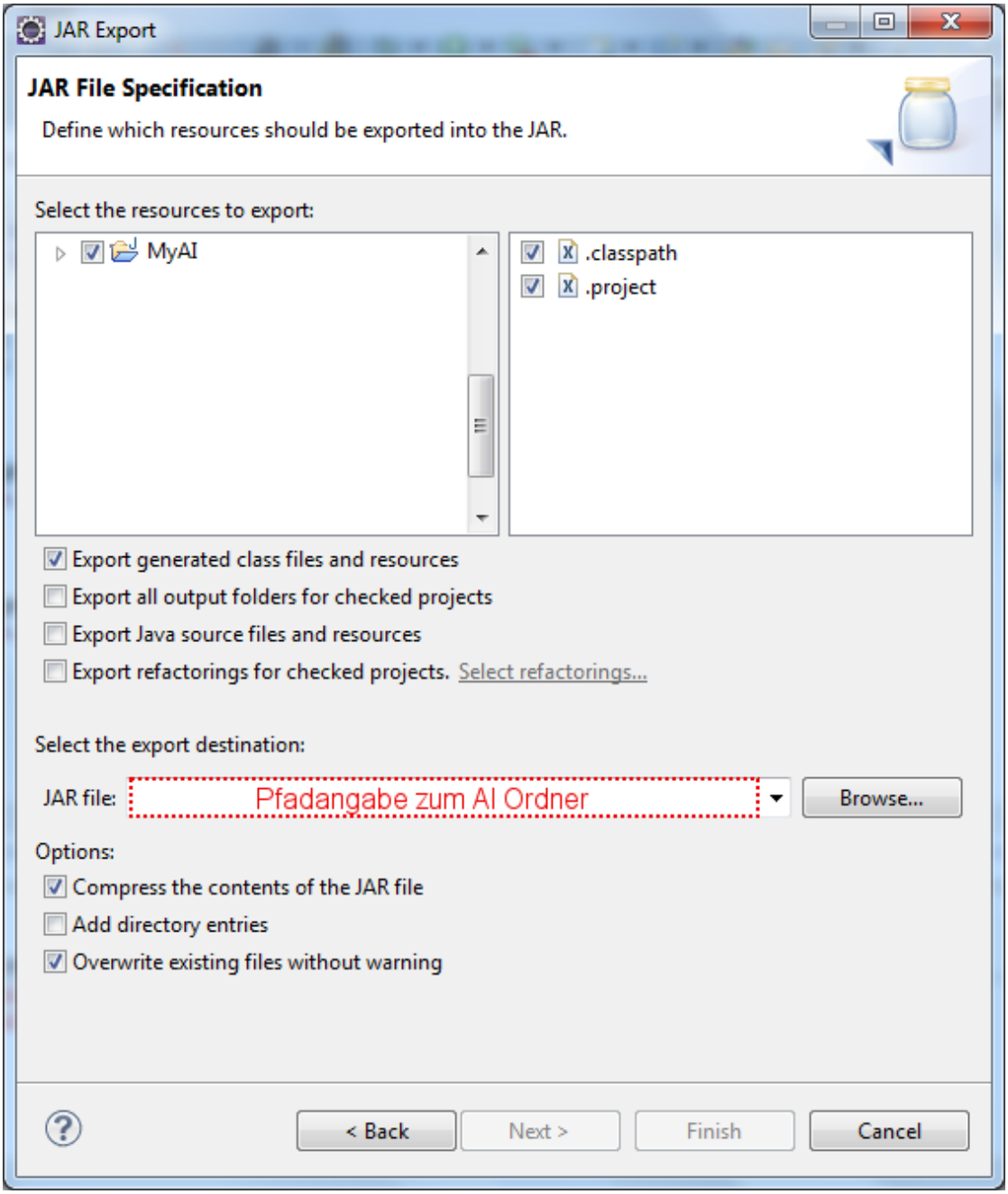
```
19
20      public MyAI() {
21
22      }
23
24      @Override
25      public void kickOff(GameInformation game, TickInformation tick,
26              KickActionHandler actionHandler) {
27          placePlayers(tick, actionHandler);
28          shootBall(tick, game, actionHandler);
29      }
30
31      @Override
32      public void freeKick(GameInformation game, TickInformation tick,
33              KickActionHandler actionHandler) {
34          placePlayers(tick, actionHandler);
35          shootBall(tick, game, actionHandler);
36      }
37
38      @Override
39      public void freePlay(GameInformation game, TickInformation tick,
40              FreePlayActionHandler actionHandler) {
41          boolean hasBallPossession = tick.hasTeamBall(true);
42          if (hasBallPossession) {
43              shootBall(tick, game, actionHandler);
44              hasBallPossession = false;
45          } else {
46
47              // Alle Spielerpositionen des eigenen Teams
48              List<Position> myPlayerPositions = tick.getPlayerPositions();
49
50              // Die Position des Balls
51              Position ballPosition = tick.getBallPosition();
52
53              // Die Anzahl der eigen Spieler
54              int numberPlayers = myPlayerPositions.size();
55
56              for (int playerID = 0; playerID < numberPlayers; ++playerID) {
57                  Position currentPlayerPos = myPlayerPositions.get(playerID);
58                  actionHandler.changePlayerDirection(playerID,
59                          getRelativePosition(currentPlayerPos, ballPosition),
60                          DEFAULT_PLAYER_SPEED);
61                  double distancePlayerBall = getDistance(currentPlayerPos,
62                          ballPosition)
63                          - game.getPlayerDiameter()
64                          / 2
65                          - game.getBallDiameter() / 2;
66                  if (distancePlayerBall < DEFAULT_AQUIRING_DISTANCE) {
67                      actionHandler.acquireBallControl(playerID,
68                              DEFAULT_AQUIRING_DETERMINATION);
69                  }
70                  if (distancePlayerBall <= 0.0) {
71                      shootBall(tick, game, actionHandler);
72                  }
73
74              }
75          }
76      }
77
78      /**
79       * Platziert jeden Spieler des eigenen Teams auf der Position des Balls.
80       *
```

```
81          * @param tick
82          *            - Eine Referenz auf die aktuelle TickInformation.
83          * @param actionHandler
84          *            - Eine Referenz auf den aktuellen KickActionHandler.
85          */
86         private void placePlayers(TickInformation tick,
87                 KickActionHandler actionHandler) {
88
89             // Alle Spielerpositionen des eigenen Teams
90             List<Position> myPlayerPositions = tick.getPlayerPositions();
91
92             // Die Position des Balls
93             Position ballPosition = tick.getBallPosition();
94
95             // Die Anzahl der eigen Spieler
96             int numberPlayers = myPlayerPositions.size();
97
98             // Platziert jeden eigen Spieler auf die Position des Balls
99             for (int playerID = 0; playerID < numberPlayers; ++playerID) {
100                actionHandler.placePlayer(playerID, ballPosition);
101            }
102
103        }
104
105        /**
106         * Berechnet die Distanz zwischen zwei Positionen.
107         *
108         * @param from
109         *            - Die erste Position.
110         * @param to
111         *            - Die zweite Position.
112         * @return Distanz zwischen zwei Positionen
113         */
114        private double getDistance(Position from, Position to) {
115
116            // Satz des Pythagoras
117            return Math.sqrt(Math.pow((to.getX() - from.getX()), 2)
118                    + Math.pow((to.getY() - from.getY()), 2));
119        }
120
121        /**
122         * Erstellt eine relative Positionsangabe.
123         */
124        private Position getRelativePosition(Position fromPosition,
125                Position toPosition) {
126            return new MyPosition(toPosition.getX() - fromPosition.getX(),
127                    toPosition.getY() - fromPosition.getY());
128        }
129
130        /**
131         * Schießt den Ball aufs gegnerische Tor, sofern die eigene Mannschaft in
132         * Ballbesitz ist.
133         */
134        private void shootBall(TickInformation tick, GameInformation game,
135                ActionHandler actionHandler) {
136            boolean isInBallPossession = tick.hasTeamBall(true);
137            if (isInBallPossession) {
138                int strikerID = tick.getPlayerWithBall();
139                Position striker = tick.getPlayerPositions().get(strikerID);
140
141                double xPosLeftGoal = 0.0;
142                double xPosRightGoal = game.getFieldLength();
```

```
143             double yPosGoal = game.getFieldWidth() / 2;
144
145             double xPosOpponentGoal = tick.isPlayingOnTheLeft() ? xPosRightGoal
146                     : xPosLeftGoal;
147             double yPosOpponentGoal = yPosGoal;
148
149             Position shootingPosition = getRelativePosition(striker,
150                     new MyPosition(xPosOpponentGoal, yPosOpponentGoal));
151             actionHandler.kickBall(strikerID, shootingPosition,
152                     DEFAULT_KICK_STRENGTH);
153         }
154     }
155
156     /**
157      * Eine Position auf dem Spielfeld.
158      */
159     private class MyPosition implements Position {
160
161         private double x;
162         private double y;
163
164         public MyPosition(double x, double y) {
165             this.x = x;
166             this.y = y;
167         }
168
169         @Override
170         public double getX() {
171             return this.x;
172         }
173
174         @Override
175         public double getY() {
176             return this.y;
177         }
178
179     }
180
181 }
```

7.1.5 KI exportieren

Um die neue KI nun zu benutzen muss eine JAR Datei erstellt werden und im Ordner AI des Installationsverzeichnisses gespeichert werden.



7.2 Debugging mittels DebuggingAI

Zusätzlich zum Interface "AI" liegt noch das erweiterte Interface "DebuggingAI" vor, welches von "AI" erbt. Dieses Interface ist für die Erstellung einer KI nicht erfordelich, kann Ihnen jedoch bei der Fehlersuche oder bei der Untersuchung der genauen Verhaltensweise Ihrer KI behilflich sein. Dadurch wird es Ihnen ermöglicht, zu jedem Tick Nachrichten in der Entwicklerkonsole anzuzeigen.
Falls Ihre KI das Interface "DebuggingAI" implementiert, wird in jedem Tick, in welchem die KI aufgerufen wird, zuerst eine der Methoden aus dem Interface "AI" (kickOff, freeKick, freePlay) aufgerufen und anschließend die Methode "debug(DebugManager manager)" aus dem Interface DebuggingAI. Mit Hilfe der Methode "print(String message)" von DebugManager kann dabei die auszugebende Nachricht übergeben werden. Dabei kann der Aufruf von "print" auch mehrmals hintereinander erfolgen.
Da die Methode "debug" stets, wie bereits beschrieben, als zweites aufgerufen wird, können innerhalb der Methoden des Interface "AI" Debugnachrichten zwischengespeichert werden, welche anschließend ausgegeben werden können.

Nachfolgend ein einfaches Beispiel einer KI, welche die zuletzt aufgerufene Aktion und die derzeitige Ballposition ausgibt:

```
1  package my.ai;
2
3  import sep.football.AI;
4  import sep.football.FreePlayActionHandler;
5  import sep.football.GameInformation;
6  import sep.football.KickActionHandler;
7  import sep.football.TickInformation;
8  import SoSi.Debugging.DebugManager;
9  import SoSi.Debugging.DebuggingAI;
10
11 public class MyDebugAI implements AI, DebuggingAI {
12
13     /**
14      * Auszugebende Nachricht beim nächsten Aufruf der Methode "debug"
15      */
16     private String debugMessage = "";
17
18     /**
19      * Vorlage für die Generierung der Beispiels-Debugnachricht
20      */
21     private static final String DEBUG_TEMPLATE = "%s: Ballposition (X:%f Y:%f)";
22
23     @Override
24     public void debug(DebugManager manager) {
25         manager.print(debugMessage);
26     }
27
28     @Override
29     public void kickOff(GameInformation game, TickInformation tick,
           KickActionHandler actionHandler) {
30         debugMessage = String.format(DEBUG_TEMPLATE, "kickOff", tick.
               getBallPosition().getX(), tick.getBallPosition().getY());
```

```
31      }
32
33      @Override
34      public void freeKick(GameInformation game, TickInformation tick,
            KickActionHandler actionHandler) {
35          debugMessage = String.format(DEBUG_TEMPLATE, "freeKick", tick.
                getBallPosition().getX(), tick.getBallPosition().getY());
36      }
37
38      @Override
39      public void freePlay(GameInformation game, TickInformation tick,
            FreePlayActionHandler actionHandler) {
40          debugMessage = String.format(DEBUG_TEMPLATE, "freePlay", tick.
                getBallPosition().getX(), tick.getBallPosition().getY());
41      }
42  }
```

8 Probleme/Lösungen

Fehlermeldung beim Speichern Falls Sie eine Fehlermeldung beim Speichern erhalten haben, überprüfen, ob Sie auf das Verzeichnis in das Sie die Simulation speichern wollen, Schreibzugriff besitzen. Zusätzlich sollte überprüft werden, ob genug Festplattenspeicher zur Verfügung steht, falls eine längere Simulation gespeichert wird.

Fehlermeldung beim Laden Die Datei kann fehlerhaft sein oder ungültig (z.B. ein unbekanntes Dateiformat, welches im Nachhinein mit der Dateiendung ".sosi" versehen wurde). Die entsprechende Datei ist nicht ladbar und kann mit dem Programm nicht weiter verwendet werden. Eine Lösung dieses Problems ist leider nicht ohne weiteres möglich.

Fehlermeldung beim Erstellen einer neuen Simulation Die Fehlermeldung tritt auf, wenn für beide Teams die gleiche Farbe gewählt wird. Wählen Sie bitte zwei unterschiedliche Farben.

Fehlermeldung beim Laden von KIs Überprüfen Sie, ob die KI Dateien in dem entsprechenden Ordner noch vorhanden sind.
Für KI-Entwickler: Es sollte zusätzlich überprüft werden, ob das KI Interface eingehalten wird und die exportierte JAR-Datei sämtliche Abhänigkeiten enthält.

9 Haftungsausschluss

Hiermit distanzieren wir uns, namentlich Bastian Birkeneder, Matthias Cetto, Bernhard Doll, Manuel Reischl und Konstantin Riel, von jeglichen Schäden, die im

Zusammenhang mit dieser Software entstehen könnten. Diese Software ist nicht für gewerbliche Zwecke gedacht und wird auch nicht vertrieben. Jegliche Nutzung geschieht auf eigene Gefahr.

10 Glossar

Wiedergabe: Die grafische Darstellung der berechneten Simulation mit einer Wiedergabegeschwindigkeit

Wiedergabegeschwindigkeit: Die Geschwindigkeit, mit der die Wiedergabe abläuft (auch bei Vorlauf und Rücklauf)

Wiedergabeposition: Die aktuelle Position der Wiedergabe

Normalwiedergabegeschwindigkeit: Eine fest definierte Wiedergabegeschwindigkeit

Simulation: Die interne (Vor-)Berechnung des Spielablaufs (in Simulationsschritten)

Simulationsschritt: Das Zeitintervall, in der die Simulation eine neue Spiel-Aktion berechnet hat

Simulationsposition: Die aktuelle Position der bereits simulierten Simulationsschritte

Simulationsdauer: Die Dauer der Wiedergabe eines Spiels insgesamt

Replay: Die erneute Wiedergabe des letzten Zeitintervalls mit definierter Länge vor der aktuellen Wiedergabeposition

Rücklaufgeschwindigkeit: Die Geschwindigkeit, mit der die Wiedergabe zurückläuft. Diese ergibt sich aus einem Vielfachen der Normalwiedergabegeschwindigkeit

Vorlaufgeschwindigkeit: Die Geschwindigkeit, mit der die Wiedergabe vorläuft. Diese ergibt sich aus einem Vielfachen der Normalwiedergabegeschwindigkeit

Tick: Ein Simulationsschritt, in welchem die KI Entscheidungen trifft und die Auswertung der Regeln und Anwendung der Spielphysik stattfindet.

Zeitleiste: Die Steuerungsleiste unter dem Spielfeld, die die Wiedergabeposition mit Hilfe eines Reglerknopfes darstellt. Zusätzlich gibt sie durch die Umfärbung der Steuerungsleiste Auskunft darüber, welche Simulationsposition die Simulation bereits erreicht hat

Vorwärtsrichtung: Die Wiedergabe erfolgt zeitlich aufsteigend der Berechnungsschritte. Die Anzeige der Spielzustände erfolgt somit in chronologischer Reihenfolge

Rückwärtsrichtung: Die Wiedergabe erfolgt zeitlich absteigend der Berechnungsschritte. Die Anzeige der Spielzustände erfolgt somit in umgekehrter chronologischer Reihenfolge