Wirtschaftsinformatik

RoboCup Soccer Server

Vortrag im Rahmen des Seminars
„Agenten und Robotfussball“

Stephan Hagemann

Was machen wir heute?

- Simulationsliga bisher vielfälltig gesehen:
  - Bei den RoboCup Turnieren
  - Als Trainingswiese für Lernalgorithmen
- Heute:
  - Wie läuft die Simulation ab?
  - Wie funktioniert die Kommunikation mit dem Server?
  - Wie sehen die verwendeten Modelle aus?
  - Was müssen Clients leisten?
  - Bewertung des Servers aus der Sicht der MAS Forschung

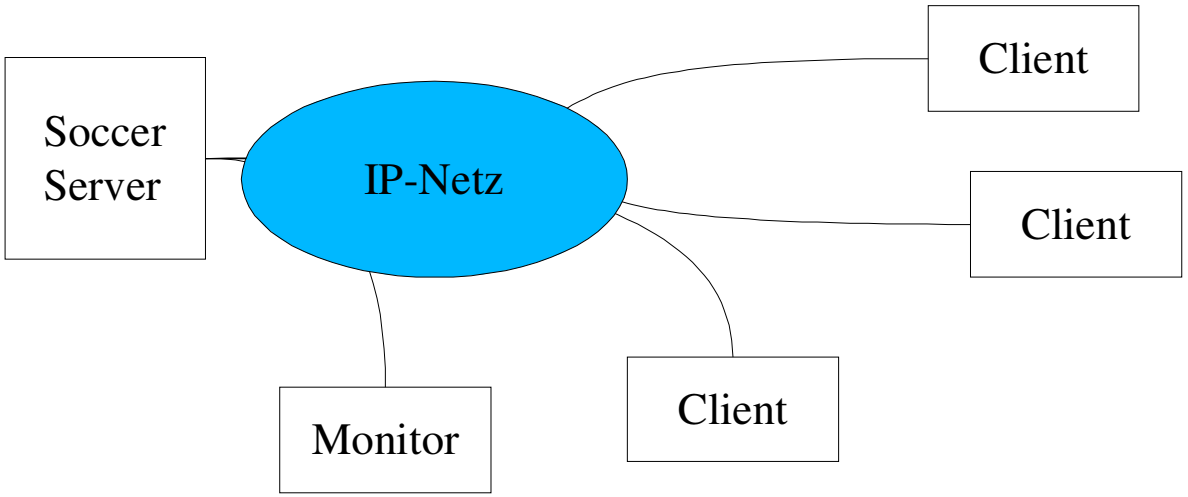
Agenda

- **A. Überblick**
- B. Modelle und Protokolle
  - 1. Wahrnehmung
  - 2. Aktionen
  - 3. Erweiterungen
- C. Clients
- D. Abschluß

Der Soccer Server

- Ein System, dass es autonomen Agenten erlaubt gegeneinander Fußball zu spielen

- Client/Server Architektur:
  - Server verwaltet das Spielfeld und simuliert alle Bewegungen
  - Jeder Client kontrolliert einen Spieler
  - Monitore erlauben die Beobachtung und Analyse eines Spiels
  - Kommunikation erfolgt über UDP/IP



Architektur

- Der Server ist der einzige beteiligte Rechner, der das Weltmodell
  - verändert (natürlich aufgrund von Clientanfragen) und
  - vollständig und genau kennt
- Die Simulation ist zeitlich diskret: Eine Halbzeit 3000 Zyklen, jeder Zyklus standardmäßig 100 ms
- Das Weltmodell ist 2D

S-Ausdrücke

- Kommunikation zwischen Server und Clients/Monitoren basiert auf S-Ausdrücken
- S-Ausdrücke: Basis von LISP und Scheme
- Struktur:
  Nachricht := Sexp

  ```
  SExp → (S)
  S    → char* | S S | (S)
  ```

- Beispiele:
  - 1. Nachricht eines Clients an den Server:
    `(init UNI-Muenster (version 9))`
  - Antwort des Servers auf dieses init:
    ...

Antwort des Servers


```
(init UNI-Muenster (version 9))
(init l 2 before_kick_off)
(server_param (catch_ban_cycle 5)(clang_advice_win 1)
  (clang_define_win 1)(clang_del_win 1)(clang_info_win 1)
  (clang_mess_delay 50)(clang_mess_per_cycle 1)
  (clang_meta_win 1)(clang_rule_win 1)(clang_win_size 300)
  (coach_port 6001)(connect_wait 300)(drop_ball_time 0)
  (freeform_send_period 20)(freeform_wait_period 600)
  (game_log_compression 0)(game_log_version 3)
  (game_over_wait 100)(goalie_max_moves 2)(half_time 10)
  (hear_decay 1)(hear_inc 1)(hear_max 1)(keepaway_start 1)
  (kick_off_wait 100)(max_goal_kicks 3)(olcoach_port 6002)
  (point_to_ban 5)(point_to_duration 20)(port 6000)
  (recv_step 10)(say_coach_cnt_max 128)
  (say_coach_msg_size 128)(say_msg_size 10)
  (send_step 150)(send_vi_step 100)(sense_body_step 100)
  (simulator_step 100)(slow_down_factor 1)(start_goal_l 0)
  (start_goal_r 0)(synch_micro_sleep 1)(synch_offset 60)
  (tackle_cycles 10)(text_log_compression 0)
  (game_log_dir "/home/thoward/data")
  (game_log_fixed_name "rcssserver")keepaway_log_dir "./")
  (keepaway_log_fixed_name "rcssserver")
  (landmark_file "~/.rcssserverlandmark.xml")
  (log_date_format "%Y%m%d%H%M")(team_l_start "")
  (team_r_start "")(text_log_dir "/home/thoward/data")
  (text_log_fixed_name "rcssserver")(coach 0)
  (coach_w_referee 1)(old_coach_hear 0)(wind_none 0)
  (wind_random 0)(auto_mode 0)(back_passes 1)
  (forbid_kick_off_offside 1)(free_kick_faults 1)
  (fullstate_l 0)(fullstate_r 0)(game_log_dated 1)
  (game_log_fixed 1)(game_logging 1)(keepaway 0)
  (keepaway_log_dated 1)(keepaway_log_fixed 0)
  (keepaway_logging 1)(log_times 0)(profile 0)
...
```

Schwierigkeiten

- Menge der zu beachtenden Informationen ist groß
- Zeitintervall ist klein und „unerbittlich“ (*simulator_step*, Standard: 100ms)
  → Realtime Bearbeitung erforderlich
- Informationen sind mit Rauschen belegt
- Einige Hilfen:
  - Library Verzeichnis zur Unterstützung grundlegender Funktionen
  - Alle Teams eines RoboCups als Binaries (teilweise auch als Source) verfügbar
  - Verschiedene Analyse-Tools

Spielregeln

- Im wesentlichen die normalen FIFA-Regeln
- Vom automatischen Schiedsrichter überwacht und kontrolliert:
  - Anstoß
  - Tor
  - Aus
  - Abstand zum ruhenden Ball bei Standardsituationen
  - Abseits (~passives Abseits)
  - Rückpass
  - Freistoß
  - Halbzeit / Spielende (Golden Goal)
  - Spielverzögerung
  - Spielstatus: z.B. `kick_off, free_kick, corner_kick, play_on`
    - werden durch Zeitfortschritt oder Aktionen verändert
    - Schiedsrichter meldet Spielern ständig Änderungen
- Andere Probleme (Ball umzingeln, Tor blockieren, Sperren ohne Ball, sonstiges störendes Verhalten) werden von menschlichem Schiedsrichter behandelt

Gentlemans Agreement

- Die gesamte Simulation läuft nur über den Soccer Server
- Es gibt eingeschränkte Kommunikation zwischen den Agenten, die über den Server läuft
- Es ist ein leichtes die Clients einer Mannschaft ohne den Server Daten austauschen zu lassen (gleicher Prozess oder direkte Kommunikation):
  - Gemeinsames Weltmodell
  - Umgehung der Servernachrichten
- Das tut man nicht!

Agenda

- A. Überblick
- **B. Modelle und Protokolle**
  - 1. Wahrnehmung
  - 2. Aktionen
  - 3. Erweiterungen
- C. Clients
- D. Abschluß

Hören – I

- Client/Schiedsrichter: `(say „Message“)`
  Server: `(hear` *`Time Sender „Message“`*`)`
  - *Time:* Aktueller Zyklus der Simulation
  - *Sender:* Trainer, Schiedsrichter, selbst oder Richtung (-180°-180°)
  - *Message:* String (nicht länger als *say_msg_size* (Standard: 512))

- Nachrichten werden sofort an alle hörenden Agenten weitergegeben
- Spieler hören Nachrichten:
  - Von beiden Mannschaften (inklusive Coach)
  - Vom Schiedsrichter
- Aber ein Spieler hört nicht jede Nachricht...

Hören – II

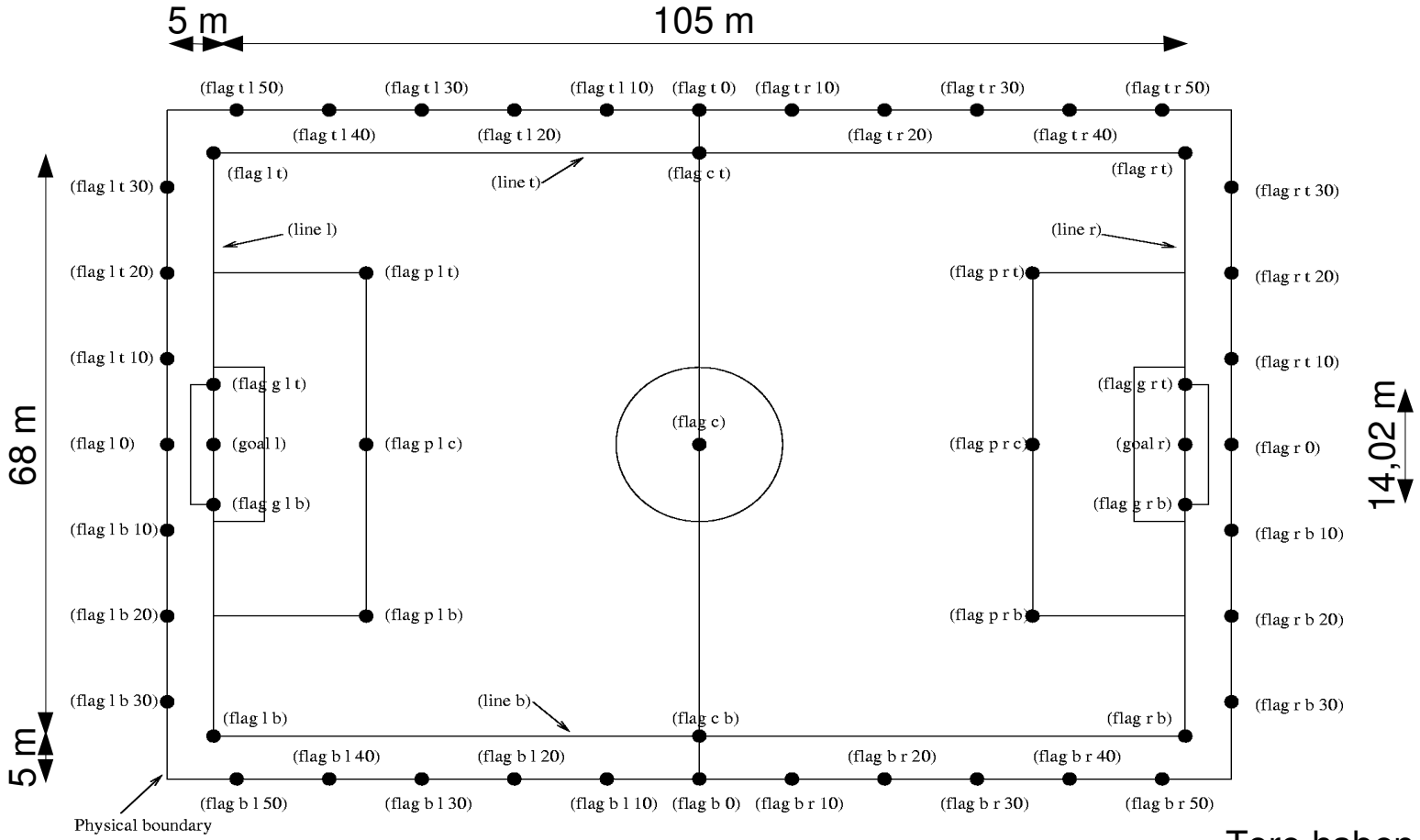
- Kapazität
  - Spieler haben eine begrenzte Aufnahmefähigkeit pro Team (implementiert über ein Punktesystem):
    - Nachricht kann aufgenommen werden, wenn aktuelle Kapazität größer ist als *hear_decay*
    - In jedem Zyklus steigt die aktuelle Kapazität um *hear_inc* bis *hear_max*
    - Standard: maximal 1 Nachricht pro Team in jedem 2. Zyklus
    - Nachrichten bei nicht ausreichender Kapazität → Inhalt undefiniert
  - Nachrichten des Schiedsrichters kommen immer an
- Reichweite
  - Nachrichten sind nur in der Umgebung des Spielers bis zu einer Entfernung von *audio_cut_dist* (Standard: 50 m) zu hören
  - Nachrichten des Schiedsrichters sind auf dem ganzen Feld zu hören

Sehen

- Server: (see (*ObjName Distance Direction [DistChng DirChng [BodyDir HeadDir]]*)$^+$)
  - *ObjName*: Name des Objekts
  - *Distance*: euklidische Distanz
  - *Direction*: Richtung des Objekts relativ zur Blickrichtung
  - *DistChng*: Veränderung der Distanz
  - *DirChng*: Veränderung der Richtung
  - *BodyDir*: Richtung des Körpers realtiv zum eigenen Körper
  - *HeadDir*: Richtung des Kopf realtiv zur eigenen Kopfrichtung

- Objekte:
  - Ball
  - Spieler
  - Linien
  - Spielfeldmarker

Sehen – Das Spielfeld

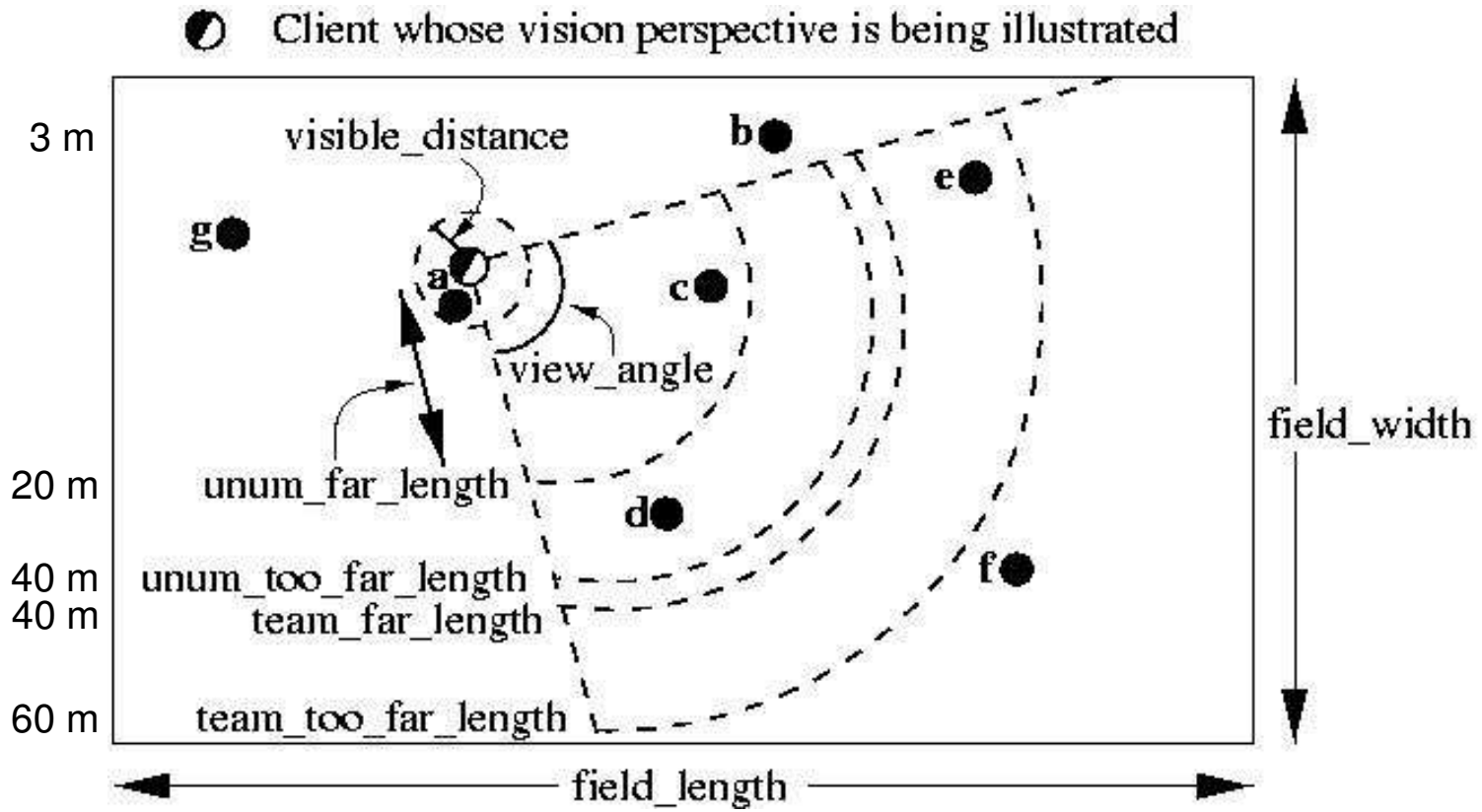


Tore haben keine Höhe!

Eigenschaften der Sicht

- Welche Informationen über einen Spieler mitgeteilt werden hängt von der Entfernung ab:
  - Nummer und Team können verdeckt sein:
    - Bis *X_far_length*: sichtbar
    - *X_far_length* – *X_too_far_length*: Wahrscheinlichkeit für Sichtbarkeit nimmt linear auf 0 ab.
    - Ab *X_too_far_length*: nicht sichtbar
- Visueller Sensor dient auch als „Nahsensor“:
  - Von Objekten in der direkten Umgebung des Agenten wird der Typ mitgeteilt
- Sichtfeld und Wahrnehmungsfrequenz können beeinflusst werden
  - `(change_view` *`Width Quality`*`)`
  - *Width*: narrow | normal | wide
  - *Quality*: high | low
  - Breite des Sichtfeldes ist 45°, 90° oder 180°
  - Frequenz steigt mit kleinerem Sichtfeld und schlechterer Qualität (Standard: 150 ms)

Sicht



- Wahrnehmung nicht fehlerfrei:
  - Entfernungen werden tendenziell überschätzt
  - Standard: bei 100 m bis zu 10 m Fehler

Körperstatus

- (sense_body *Time*
  (view_mode ViewQuality ViewWidth)
  (stamina Stamina Effort)
  (speed AmountOfSpeed DirectionOfSpeed)
  (head_angle HeadDirection)
  (kick KickCount)
  (dash DashCount)
  (turn TurnCount)
  (say SayCount)
  (turn_neck TurnNeckCount)
  (catch CatchCount)
  (move MoveCount)
  (change_view ChangeViewCount))

- Wahrnehmung automatisch alle *sense_body_step* ms (Standard: 100 ms)
- Gibt dem Spieler die Serversicht (interessant z.B. falls UDP Datagramme untergegangen)

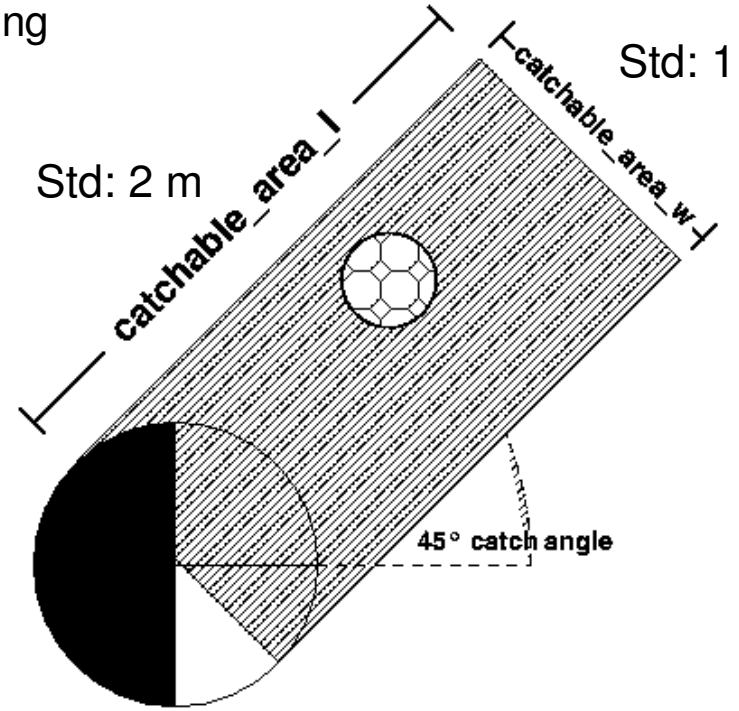
Aktionen

- Catch
- Dash
- Kick
- Move (`move` *X Y*)
- Turn
- TurnNeck

- Jedes Kommando wird vom Server von einem Spieler nur einmal pro Zyklus angenommen

Catch

- `(catch` *`Direction`*`)`
  - *Direction*: Fangrichtung relativ zur Blickrichtung



- nur der Torwart kann catch Kommandos absetzen
- *catch_probability* (Standard: 1), wenn der Ball innerhalb der Reichweite ist
- Nach vergeblichem Fangen nächster Versuch erst nach *catch_ban_cycle* (Standard: 5)
- Nach dem Fangen kann der Torwart bis zu *goalie_max_moves* move-Kommandos ausführen

Das Bewegungsmodell

- Die Bewegung in jedem Zyklus in Schritten:
  - Beschleunigung anwenden
    - Abhängig von der Kraft bei Beschleunigung oder Schuß
    - Beschleunigung kann nicht beliebig groß werden, eventuell Deckelung
  - Bewegung durchführen
  - Geschwindigkeitsverlust
    - Verlust: 6% für den Ball, 60% bei Spielern
  - Beschleunigung = 0 setzen

- Bewegungen sind ungenau:
  - Fehler sind gleichverteilt
  - Bereiche unterschiedlich für verschiedene Objekte
  - Betroffene Berechnungen:
    - Beschleunigung
    - Krafteinsatz beim Schießen und der Beschleunigung
    - Winkel bei Drehungen

- Kollisionen:
  - Überlappen sich zwei Objekte am Ende eines Zyklus, werden sie in kleinen Schritten rückwärts bewegt bis sie sich nicht mehr überlappen

Dash

- Client: `(dash` *`Power`*`)`
  - *Power*: Krafteinsatz bei der Beschleunigung

- Beschleunigen kostet Energie, Rückwärts doppelt

- Ausdauermodell:
  - Einsatz:
    - Wird reduziert, wenn Energie unter 30% des Maximums sinkt
    - Sonst pro Zyklus etwas angehoben
    - Mindesteinsatz: 60%
  - Erholungsrate
    - Wir reduziert, wenn Energie unter 30% des Maximums sinkt
    - Wird nur vor den Halbzeiten auf 100% gesetzt
    - Mindestrate: 50%

- Beschleunigung also von Kraft und Einsatz (also der Ausdauer) abhängig

Kick

- Client: (kick *Power Direction*)
  - *Power*: Krafteinsatz
  - *Direction*: Schußrichtung
- Ein Ball kann nur geschossen werden, wenn er im Umkreis des Spielers ist
- Wirkung:
  - Die effektive Schußkraft wird reduziert durch
    - Abstand zwischen Ball und Spieler
    - Winkel des Ball relativ zur Körperrichtung
    - In beiden Fällen Worst-Case: -25% Schußkraft
- Krafteinsatz und Ballgeschwindigkeit sind gedeckelt:
  - 45 m größtmögliche Schußdistanz bei optimalen Schuß
- Wie beim richtigen Fußball:
  - Es kann besser sein den Ball nicht sofort wegzuballern
  - Annehmen (Abstoppen), vorlegen und dann Schießen meistens besser
- Treten mehrere Spieler den Ball wird der Ball durch jeden Schuß beschleunigt

Turn, Turn Neck

- Client: (`turn` *`Angle`*)
- Lässt den Spieler eine Körperdrehung ausführen
- Trägheit:
  - Effektiv möglicher Drehwinkel von der Geschwindigkeit des Spielers abhängig

- Client: (`turn_neck` *`Angle`*)
- Verändert die Blickrichtung des Spielers relativ zur Körperrichtung
- Blickrichtung ändert sich bei Richtungsänderung automatisch mit

Weitere Möglichkeiten im Modell

- Heterogene Spieler
  - Spieler können sich in praktisch allen Eigenschaften unterscheiden
  - Jede Mannschaft erhält die gleichen Spielertypen in gleicher Anzahl

- Wind

Trainer

- Trainieren der Spielereigenschaften erfordert eventuell viele gleiche Spielsituationen
- Trainer-Client bietet Möglichkeiten, dass Spiel vollständig zu kontrollieren
  - Kontrolle des Spielstatus
  - Kommunikation mit den Spielern
  - Veränderen von Objektpositionen und -geschwindigkeiten
  - Vollständige und unverrauschte Sicht des Spielfeldes
- Schiedsrichter kann ausgeschaltet werden
- Darf bei Turnieren nicht verwendet werden, dafür gibt es den Online Coach...

Online Coach

- Darf in Turnieren verwendet werden. Also: 11 Spieler + Online Coach
- Hat eingeschränkte Fähigkeiten gegenüber dem Trainer
  - Kann mit den Spielern kommunizieren
  - Vollständige und unverrauschte Sicht des Spielfeldes
- Kommunikation mit den Spielern ist stark eingeschränkt:
  - Es soll verhindert werden, dass der Coach die Planung des Spiels übernimmt
  - Coaches können nur Nachrichten in einer eingeschränkten Sprache schicken
    - Definitionen: Bedingungen, Aktionen, Regionen, ...
    - Regeln setzen und löschen
  - Senden nur alle 300 Zyklen erlaubt

Agenda

- A. Überblick
- B. Modelle und Protokolle
  - 1. Wahrnehmung
  - 2. Aktionen
  - 3. Erweiterungen
- **C. Clients**
- D. Abschluß

Was muss ein Client tun?

- Paralleles Verarbeiten der Informationen und Vorbereiten der eigenen Aktionen

- Klassen:
  - Player.java
  - CommunicationChannel.java
  - Actor.java

Agenda

- A. Überblick
- B. Modelle und Protokolle
  - 1. Wahrnehmung
  - 2. Aktionen
  - 3. Erweiterungen
- C. Clients
- D. Abschluß

Der SoccerServer als MAS Architektur

- Fußball ist ein Standardproblem für das sich viele begeistern können
  Spiel einfach, Umsetzung mit gutem Teamspiel schwierig

- Der Server ist allgemein verfügbar
  - Auf verschiedenen Plattformen
  - Source-Code
  - Leichtgewichtig; verbraucht keine besonderen Ressourcen
  - UDP macht unabhängig von Programmiersprachen

- Viele Felder für die Forschung interessant
  - Monitore
  - Schiedsrichter
  - Coaches
  - Simulation
  - Kommentatoren

- Weiterentwicklungen:
  - Realistischere Modell: 3D, Bewegung, Ausdauer
  - Schiedsrichter als Person?

Vielen Dank!

Nicht vergessen:

Ab dem 2. Juli: RoboCup 2003 in Padua!