Ostfalia
Hochschule für angewandte Wissenschaften

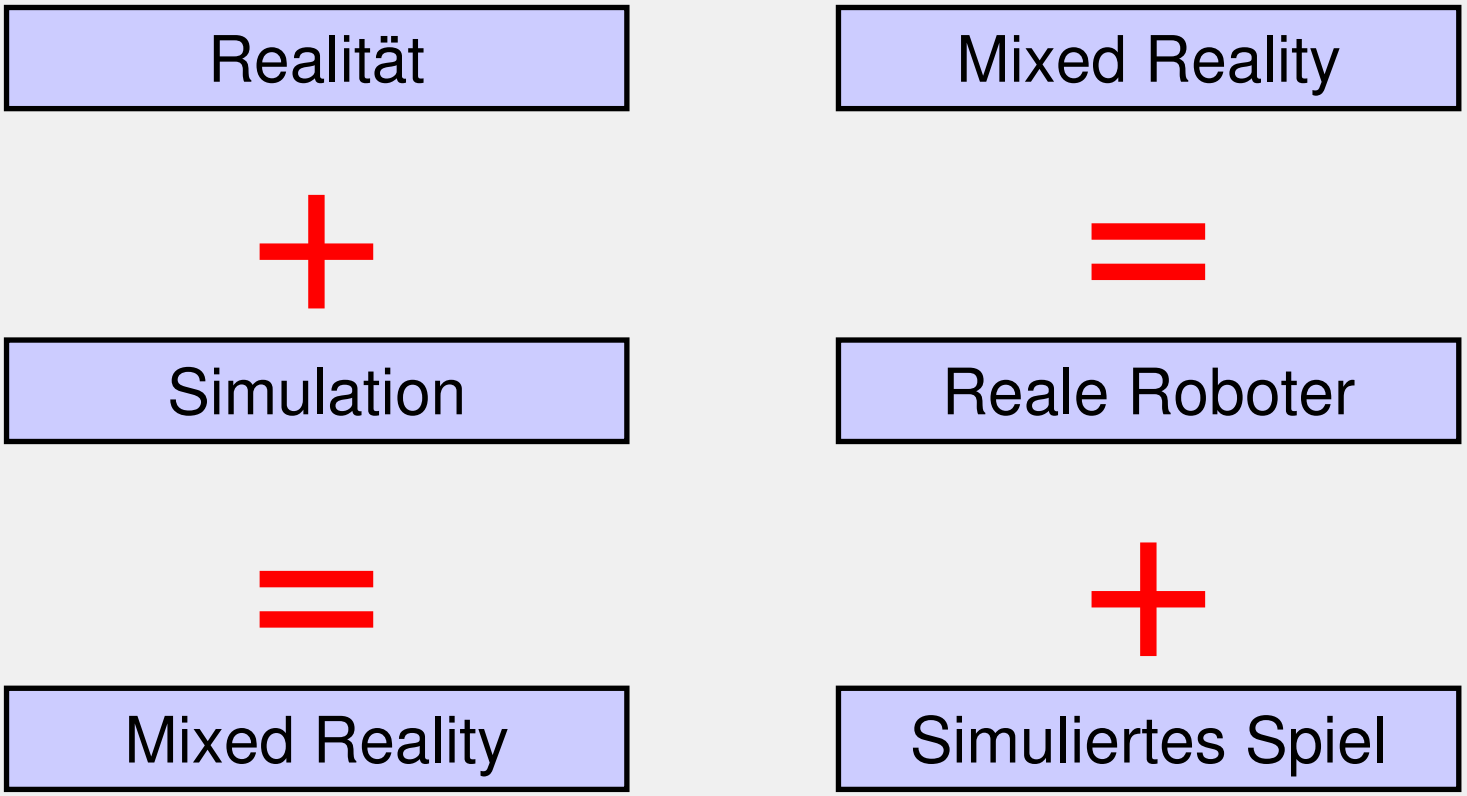
Einführung in das Mixed Reality System

Stefan Krupop

Inhalt

1. Mixed Reality – Was ist das?
2. Hardware
    1. Komponenten
    2. Die Roboter
    3. Differentialantrieb
3. Systemsoftware
    1. Komponenten
    2. Agent-Software
4. Die eigene KI
    1. Beispielagent
    2. Beispiel erweitern
    3. Vektoren berechnen
    4. Spiel einrichten
    5. Auf dem Weg zur KI
    6. Vorsicht Falle(n)!

Mixed Reality System



Mixed Reality System

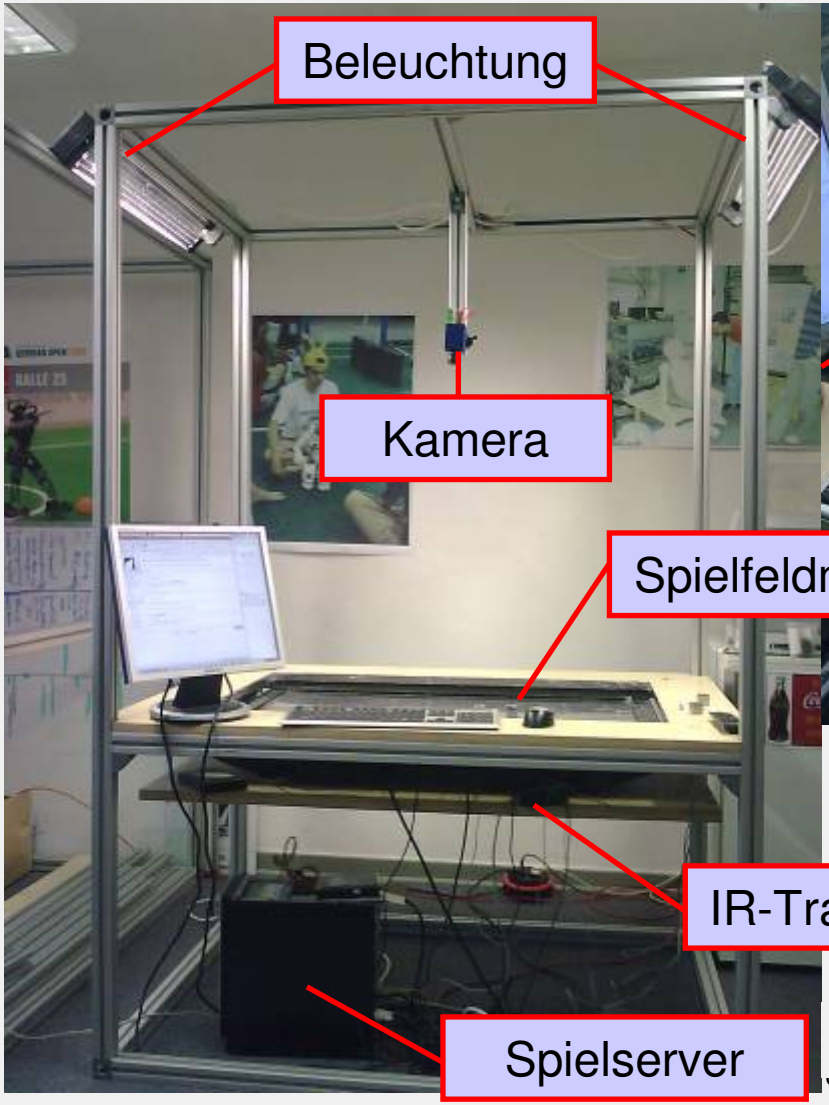
Mixed Reality League

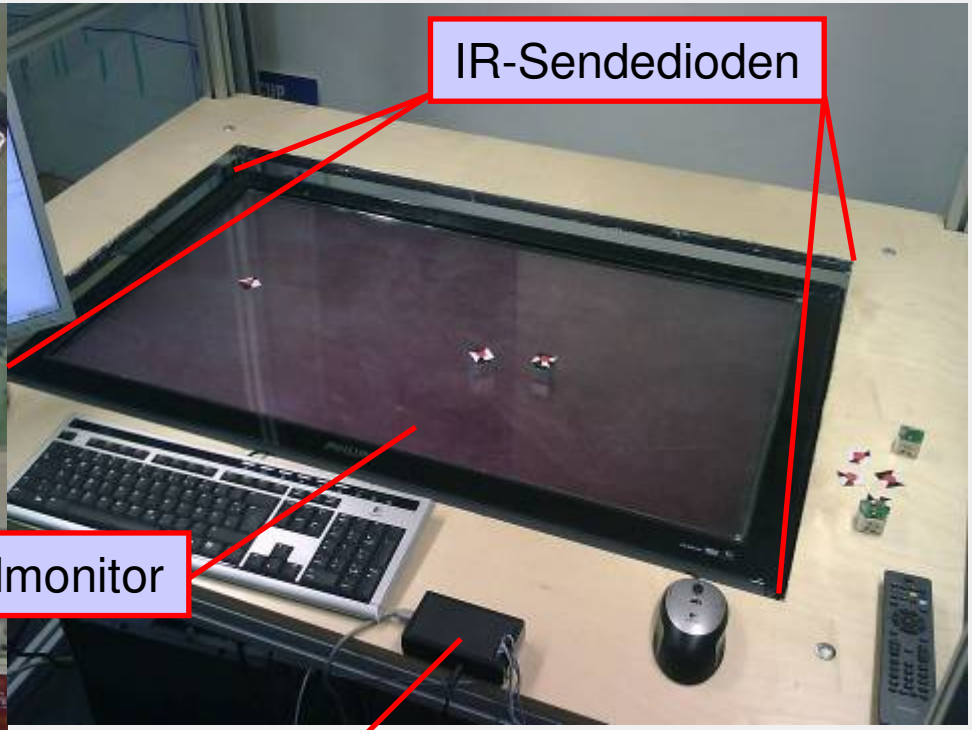
- Seit 2007 neue Liga beim RoboCup
- Entstanden aus Simulation League
- Seit 2008 neues System, erstellt von vielen verschiedenen Teams
- Seit 2009: Spiel 5 gegen 5

➔ 10 Roboter fahren als Platzhalter in sonst simulierter Umgebung auf einem Bildschirm

Hardware

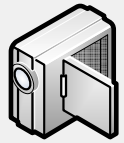
Systemüberblick Hardware



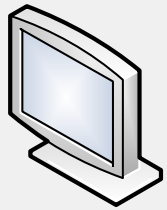


Komponenten 1/2

- Kamera
  - Nimmt Spielfeld auf
  - Liefert Daten für Vision-Tracking

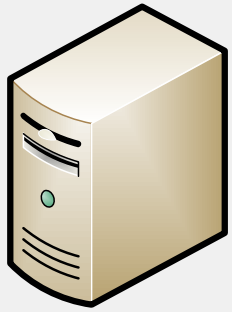


- Spielfeldmonitor
  - Zeigt das Spielfeld an und dient als Spielfläche für die Roboter
  - Kann theoretisch entfallen, macht aber „Mixed Reality“ erst aus



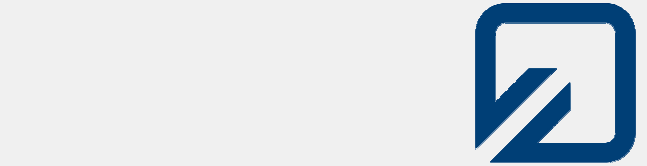
- IR Tranceiver
  - Ermöglicht die Kommunikation vom Spielserver zu den Robotern
  - Verwendet IrDA, aber mit viel höherer Sendeleistung

Komponenten 2/2

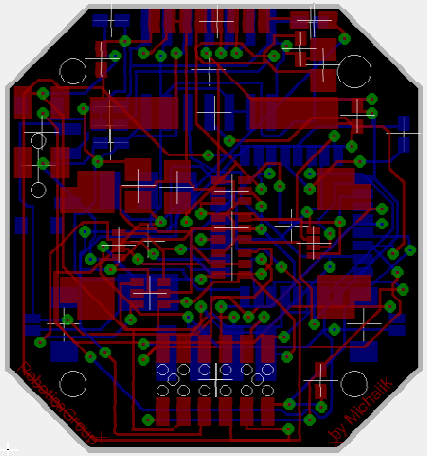


- Spielserver
  - Standard-PC mit Linux
  - Verwaltet das Spiel
  - Software besteht aus vielen Einzelkomponenten
    - SoccerServer
    - Graphics Module
    - Vision Tracking
    - RobotControl
    - Operator
    - (Clients)

Eco!Be-Roboter

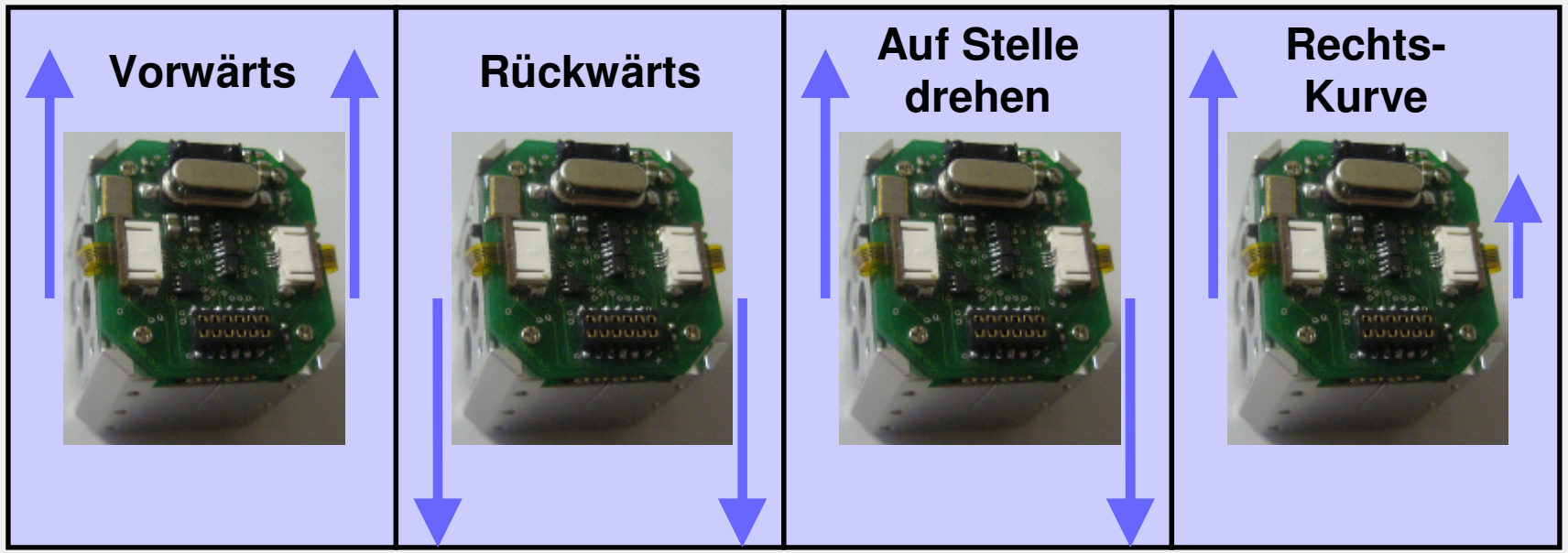


- Sehr kleiner Roboter:
  Nur ca. 25mm Kantenlänge
- Antrieb über zwei Räder
  (Differentialantrieb)
- Steuerung über Infrarot
- 2 LiPo-Akkus integriert
- Keine eigene Sensorik,
  nur „fahrbare Platzhalter“
- Alle Roboter empfangen
  alle Befehle, aber jeder
  Roboter hat eindeutige ID



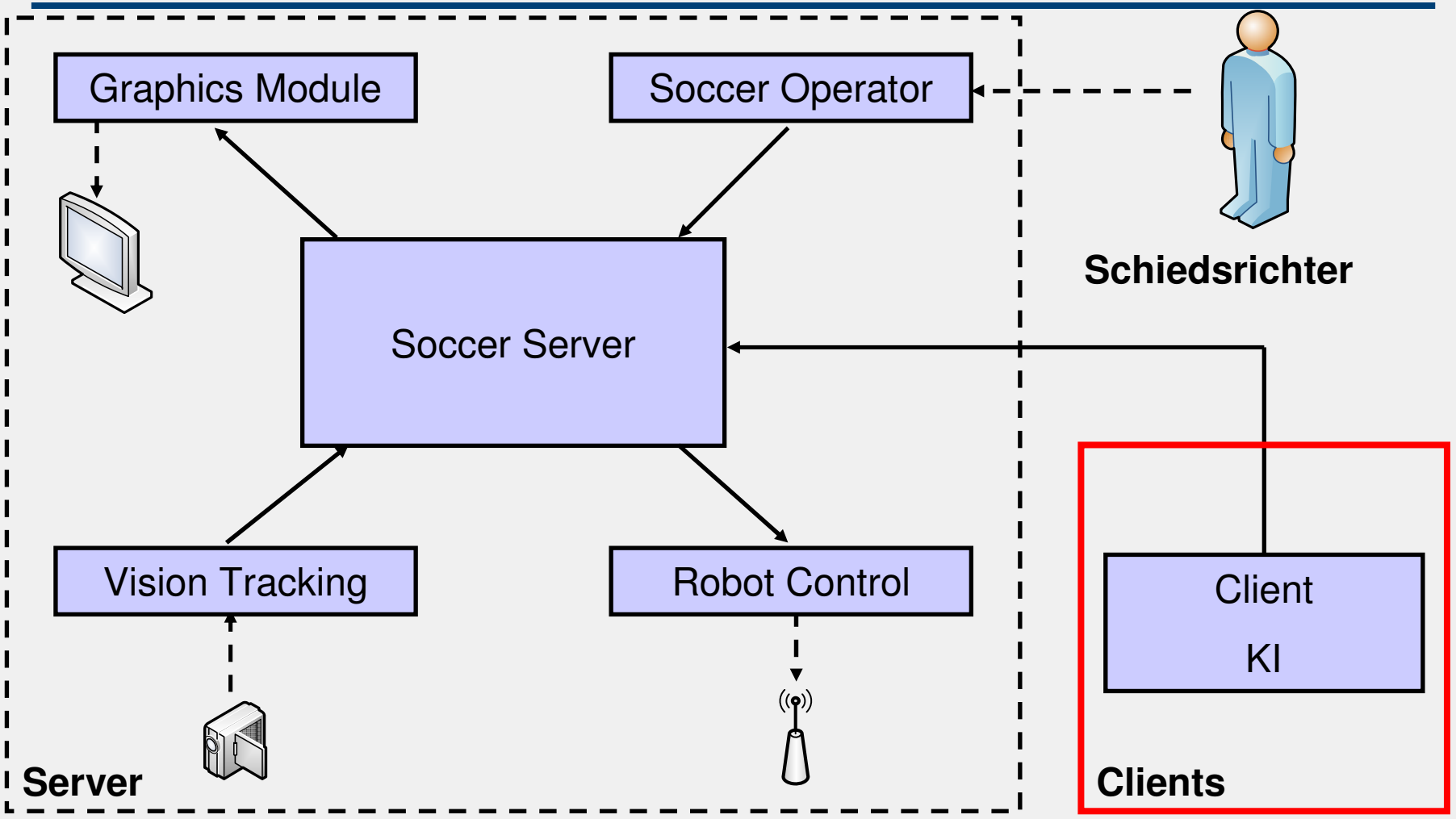
Differentialantrieb

- Bewegungsrichtung abhängig von Geschwindigkeit/Richtung der beiden Räder



Systemsoftware

Systemüberblick Software



Systemüberblick Software: Graphics Module

- Universelles Grafikausgabe-Tool
- Sorgt für Darstellung der virtuellen Welt auf dem Spielfeldmonitor
- Verwendet Themepacks
- Bildinhalt wird per XML vom Soccer Server vorgegeben

Systemüberblick Software: Vision Tracking

- Holt Bilddaten von Kamera
- Erkennt Position und Ausrichtung der Roboter anhand spezieller Marker
- Jeder Marker hat eindeutige ID
- Sendet Daten in XML verpackt an Soccer Server

Systemüberblick Software: Soccer Server, Robot Control und Operator

- Soccer Server
  - Zentrales Element
  - Verwaltet Spiel
  - Koordiniert Module
- Robot Control
  - Verpackt Steuerbefehle des Soccer Server in Roboterprotokoll
  - Steuert IR-Transceiverhardware an
- Soccer Operator
  - Ermöglicht Steuerung des Spiels durch den Schiedsrichter

Agent-Software

Systemüberblick Software: Clients 1/2

- Prinzip: Jeder Agent (=Roboter) hat eigene „Ego-View“ auf das Spiel
  → Agent ist für sich Zentrum der Welt
  → Polarkoordinatensystem (Winkel und Länge)
- Server liefert Daten über
  - Flaggen (z.B. myGoalPole1 & 2)
  - Ballposition
  - Mitspieler
  - Gegner
  - …

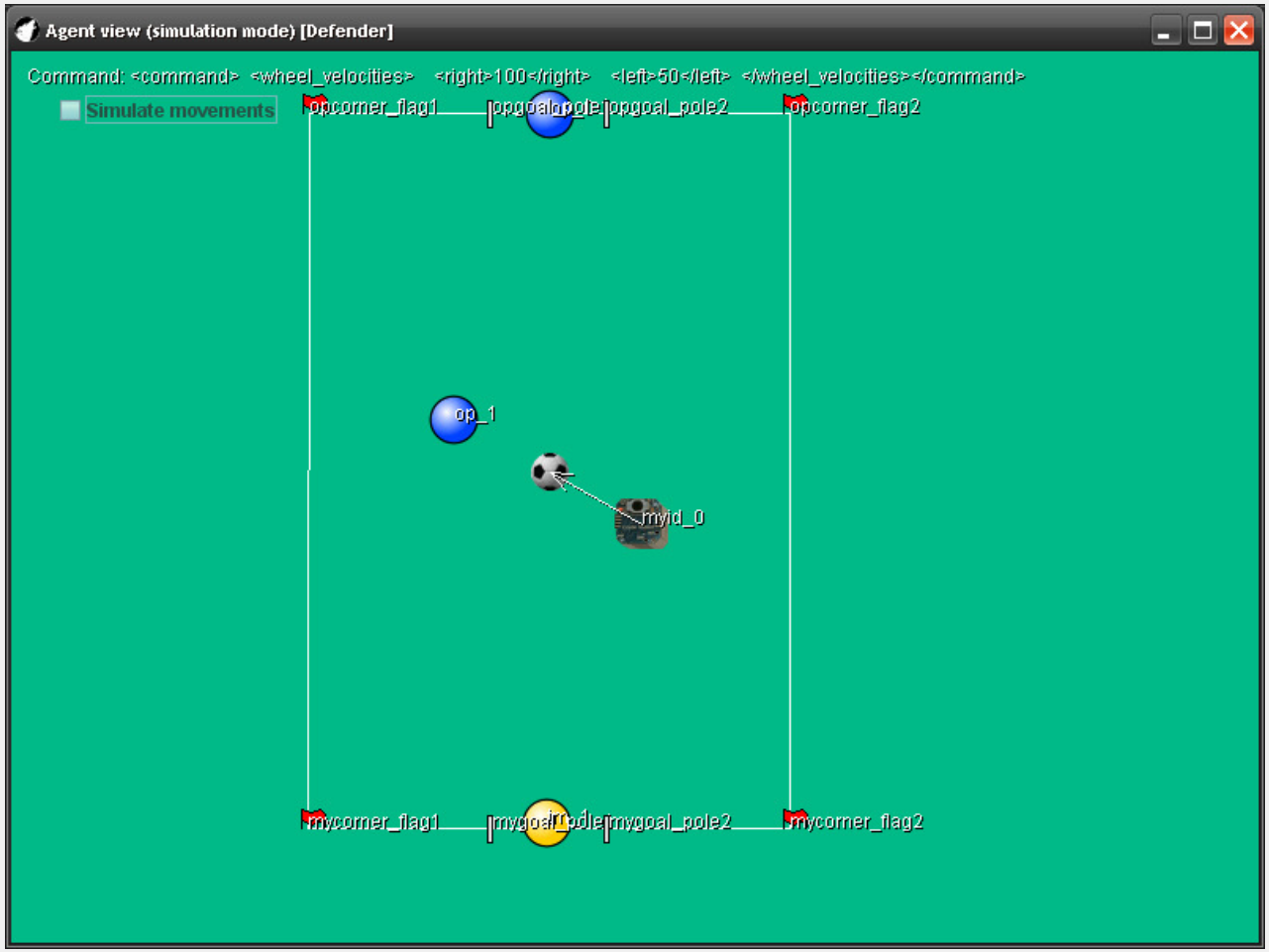
Systemüberblick Software: Clients 2/2

- Clients senden Roboterbefehle an Soccer Server:
  - Radgeschwindigkeiten setzen (links, rechts)
  - Ball schießen (Winkel, Kraft)
- Immer gleicher Zyklus:
  - Daten vom Server holen (update)
  - Daten verarbeiten, zu sendenden Befehl bestimmen
  - Befehl an Server senden (flush)
  - Auf nächsten Zyklus warten (Intervall: 66ms)

Systemüberblick Software: jMRClient (Beispielclient)

- Alle wesentlichen Funktionen bereits abstrahiert
- Spieldarstellung (AgentView) und „Taktik-Board“ (AgentViewEx) inkl. Mini-Simulator enthalten
- Spieldaten und Roboterbefehle über „Client“-Objekt direkt zugänglich

Systemüberblick Software: jMRClient – Screenshot



Die eigene KI

Spielregeln (Grundzüge)

- Zwei Teams mit je 5 Robotern treten gegeneinander an
  - Team Yellow
  - Team Blue
- Einzelne Agenten sind autonom
  - Keine (direkte) Kommunikation untereinander
  - Keine „Master-KI“
- Regelwerk der Liga regelt Verhalten der Teams und des Schiedsrichters

KI ↔ jMRClient

- jMRClient kümmert sich um „technische“ Aspekte
  - Verbindung zu Server
  - Parsen der Daten
  - Einfache Bewegungsfunktionen
- KI braucht nur Verhalten implementieren
- Minimale KI („gehe zum Ball“) benötigt nur 5 Zeilen Code

Systemüberblick Software: jMRClient – Minimaler Agent

```
public class MovingBot extends AbstractBot {

 public MovingBot(String server, ServerType type, Team team, int
   vtID, int rcID, String botName, boolean simulationMode) {
  super(server, type, team, vtID, rcID, botName, simulationMode);
 }

 public boolean process(IClient client, AgentView view) {
  Vector v = client.getPositions().ball();
  if (v != null) {
   view.drawVectorFromVector(client.getPositions().me(), v);
   client.getMovements().goTo(v);
  }
  return true;
 }
}
```

Vektor zum Ball holen

Wenn Ball gefunden…

Vektor einzeichnen

Dorthin fahren

Programm weiterlaufen lassen

Systemüberblick Software: jMRClient – Klassen 1/2

- TeamStarter
  Startet entsprechende Anzahl Agenten und gibt Parameter wie Server-Adresse, Team und Simulationsmodus vor
- AbstractBot
  Enthält Logik zum Start und zur Steuerung des Agenten, kümmert sich um Aktualisierung der AgentView
- <eigene KI>
  Erweitert AbstractBot und braucht nur noch boolean process(IClient, AgentView) { } implementieren

Systemüberblick Software: jMRClient – Klassen 2/2

- IClient (Client2007/Client2008)
  - Stellt allgemeine Daten zum Roboter bereit:
    - Zeit, Punktestand, Spielzustand, Eigene ID, …
  - Ermöglicht Zugriff auf PositionsStorage und Movements
- IPositionsStorage
  - Stellt Vektoren zu Spielelementen bereit
  - Enthält Hilfsfunktionen
- IMovements
  - Stellt Befehle zur Robotersteuerung zur Verfügung
  - Besonders wichtig: goTo(Vektor) und kick
- Vector
  - Speichert einen Winkel und eine Länge
  - Stellt Operationen für Vektormathematik zur Verfügung

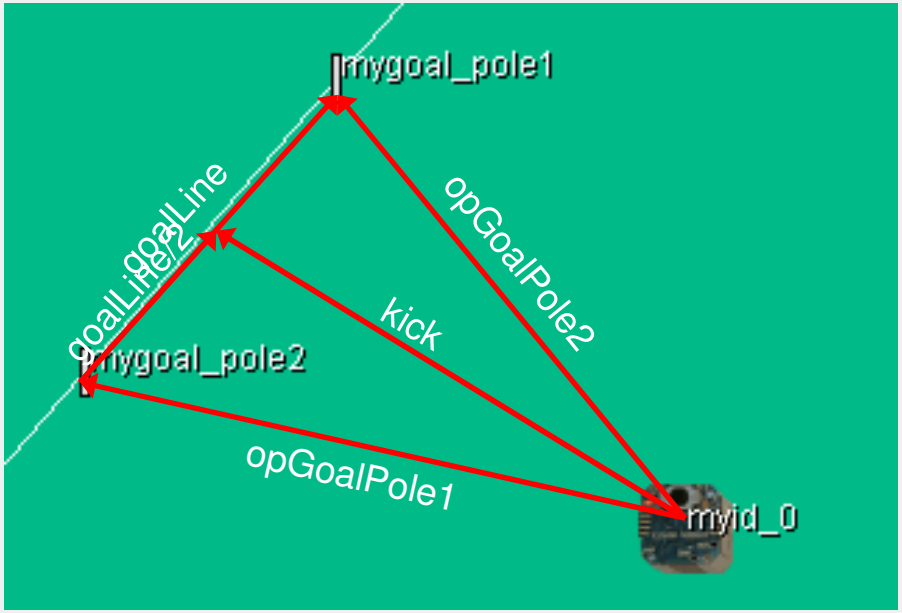
Beispielagenten erweitern

- Beispielclient fährt nur zum Ball
- Einfache Erweiterung: Anschließend auf Mitte des gegnerischen Tores schießen

- Was ist nötig?
  - Herausfinden, ob Ball in Schußreichweite
  - Zielposition des Schusses berechnen
  - Ball „treten“

Vektormathematik

- Darstellung aller Objekte als Vektoren (Winkel und Distanz)
- Berechnung anderer Punkte mittels Vektormathematik möglich



- goalPole1
- goalPole2
- goalLine=goalPole1 - goalPole2
- goalLine=goalLine/2
- kick=goalPole2 + goalLine

Vektormathematik – Als Code

```
public boolean process(IClient client, AgentView view) {
 Vector v = client.getPositions().ball();
 if (v != null) {
  if (client.getMovements().canKick()) {
   Vector opGoalPole1 = client.getPositions().opGoalPole1();
   Vector opGoalPole2 = client.getPositions().opGoalPole2();

   Vector goalLine = opGoalPole1.subtract(opGoalPole2);
   goalLine = goalLine.scale(0.5f);

   Vector kick = opGoalPole2.sum(goalLine);

   client.getMovements().kick(kick.getAngle(), 1.0);
  } else {
   view.drawVectorFromVector(client.getPositions().me(),
   client.getMovements().goTo(v);
  }
 }
 return true;
}
```

Überprüfen, ob Ball in Schussreichweite

Ball in Richtung des Vektors kicken

Der TeamStarter

- Legt globale Parameter fest:
  - Server-IP, Team-Farbe, Simuliertes/Echtes Spiel

```
public static final String     SERVER     = "192.168.0.3";
public static final ServerType SERVERTYPE= ServerType.Server2008;
public static final Team       TEAM       = Team.YELLOW;
public static final boolean    SIMULATION_MODE = true;
```

- Fügt Agents dem Team hinzu
  - Legt Agent-Klasse für jeden Agent fest
  - Legt fest, welche Marker-ID und welche Roboter-ID je Agent verwendet werden

```
bot = new MovingBot(SERVER, SERVERTYPE, TEAM, 0, 0, "Bot " + 0,
                    SIMULATION_MODE);
bot.start();
mBots.add(bot);
```

Der Simulationsmodus

- Hilfreich zum Testen des Agent-Codes
- Benötigt keinen Server
- Roboter kann durch Klicken und Ziehen bewegt werden
  - Linke Maustaste: Verschieben
  - Rechte Maustaste: Drehen
- Bewegungssimulation kann pausiert werden
- Aber Vorsicht: Ersetzt nicht Tests mit echtem System oder zumindest dem Simulator!
  - Simuliert nur den jeweiligen Agent, alles andere statisch
  - Simulation nicht unbedingt realitätsnah!

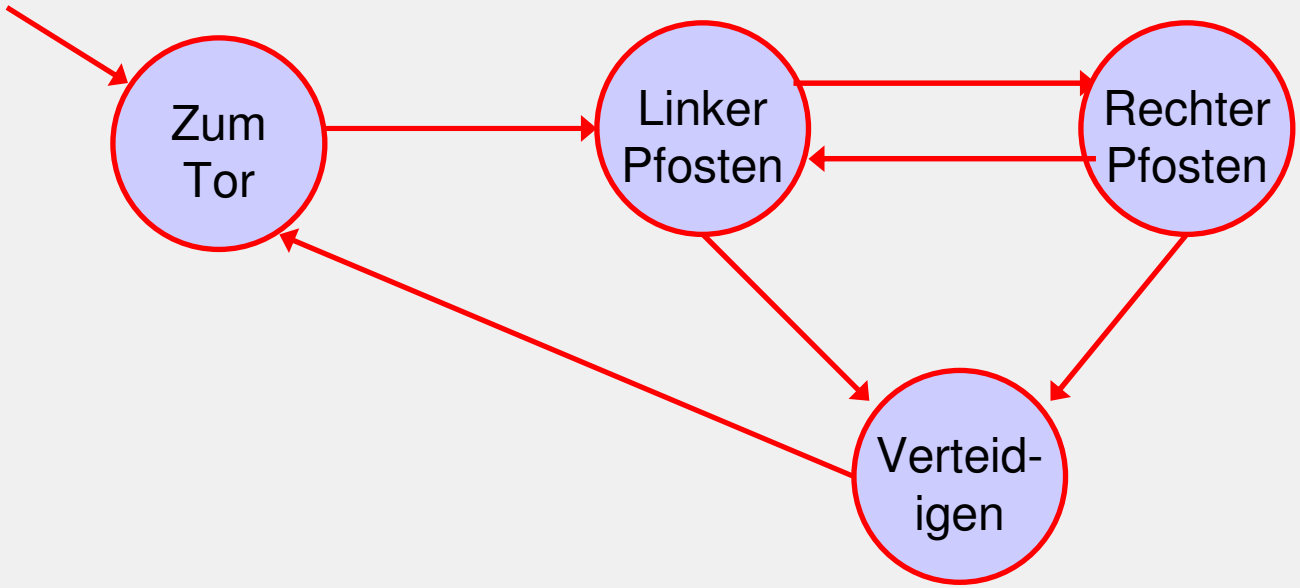
Vom primitiven Agent zur „KI“

- Nur „Gehe zum Ball und schieße“ reicht nicht
  - Tor unverteidigt
  - Großes Gedränge um Ball
- ➔ „Rollenverteilung“ nötig
  - Torwart
  - Verteidiger
  - Angreifer
  - …
- Rollenzuweisung kann fest oder dynamisch sein
  - Bei dynamischer Zuweisung Festlegung von Regeln für z.B. „Ich bin Torwart“ nötig

Mögliche Realisierungen von „KI“

- Verhalten gesteuert durch Ansammlung von „if-then-else“ (Reflex-Agent)
  - Erweiterung des Beispiel-Agenten
  - Einfach Erfolge möglich, aber schnell unübersichtlich
- Zustandsautomat steuert Verhalten
  - Einfache Möglichkeit, „Wissen“ über die Umwelt zu speichern
- Mehrere Verhalten laufen „parallel“ und können sich überlagern
- Steuerung durch externen Regelsatz und Zustandsautomat
- Selbstlernendes Verhalten
  - Sehr komplex (großer Zustandsraum)
    → Kaum für Steuerung des ganzen Agenten geeignet, eher für einzelne Aufgaben

Beispiel für Zustandsautomat



➔ GoalKeeperBot.java

Vorsicht Falle(n)!

- Reales Spiel ist nicht so perfekt wie Simulation!
- Häufig auftretende Probleme:
  - Vision erkennt Marker nicht → Bot „lost“
  - Roboter empfängt eine Zeit lang keine Befehle

  ➔ KI muss mit solchen Situationen umgehen können
- Zeitlicher Abstand zwischen Ansteuerung und Datenaktualisierung sorgt für Probleme bei Regelung
  - z.B. Anfahren eines Winkels: Starkes Übersteuern möglich, muss bedacht werden

Downloads

Download dieser Folien und des Beispielclients unter:

http://stefankrupop.de/