Uppsala Underdogs
Sony AIBO ERS-210
Sony AIBO ERS-210

ERS-2xx Legs

<table>
<thead>
<tr>
<th>Joint</th>
<th>Δx</th>
<th>Δy</th>
<th>Δz</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. - shoulder</td>
<td>59.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2. - elevator</td>
<td>0</td>
<td>0</td>
<td>59.2</td>
</tr>
<tr>
<td>3. - knee</td>
<td>64</td>
<td>0</td>
<td>12.8</td>
</tr>
<tr>
<td>4. - ball</td>
<td>55.748</td>
<td>-11.708</td>
<td>0.5</td>
</tr>
<tr>
<td>5. - ball</td>
<td>60.627</td>
<td>-22.171</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Paw button is intersection of a 24.00% diameter cylinder and a 27.92% diameter sphere

Each link offset is relative to previous link
Legs are shown with knees bent at 35°
Sony AIBO ERS-210

- AperiOS
- OPEN-R
The field

- 4 robots in each team
- Blue team defends blue goal
- Red team defends yellow goal
- One goalie for each team
- Four beacons
Robot soccer rules

The game:

- 10 min first half
- 10 min half-time break (change team color)
- 10 min second half
RoboCup Fouls

Obstructing player

- Ball holding
- Illegal defender
- Pushing

- Penalty: 30 sec
  - if goalie fouls 0 sec time penalty
Tekkotsu

- Application framework for robotic platforms.
- Builds on the basic functionality provided by the AIBOs OPEN-R API.
- The applications are called Behaviors since they are running on a robot.
Tekkotsu - system processes

- **MainObj** - Does most of the bulk processing
- **MotoObj** - Handles movement of the robot's joints
- **SoundPlay** - Handles sound playback

These processes run in parallel, but multiple behaviors that run in the same process can only execute one at a time.
Tekkotsu – overview
Tekkotsu

- In MainObj we run our own behaviors, this is how we implement most of our modules.
- A behavior can communicate with another behavior via events.
- All events are handled by the EventRouter that schedules behaviors to execute if they have a waiting event.
TekkotsuMon

- A set of tools for monitoring, controlling and calibrating the AIBO
Design
Vision
Vision

- Implements image analysis
- Identifies visible objects and calculates their positions relative to the AIBO
- Provides Positioning with object information
Vision – color segmentation
Vision – color segmentation
Vision – class Vision

- Listens for CMVision events
- Converts image data to identifiable field objects by using the ObjectLocalization class
- Calls Positioning module with list of field objects
Vision – class ObjectLocalization

Identifies and calculates robot-relative positions of:

- Balls
- Beacons
- Goals
- Other AIBOs
Vision – field object identification

Identification parameters

- Color
- Area
- Width/height proportion
- Center of gravity
- Height above field
- Neighboring regions
- Previously identified field objects
Vision – distance calculations

Ball
• Maximum width of color region

Beacons
• Average of color regions' median widths

Goals
• Heights near left and right region edges

AIBOs
• Area of color region cluster
Positioning

- Calculates AIBO and ball positions
- Inertia added with Monte Carlo Localization (MCL)
- Stores calculated position in DB
Positioning – AIBO position

- Two types of input data used:
  - Visible field objects (Vision behavior) for geometrical calculations
  - Odometric data (Movement behavior) for dead reckoning
    - At least two beacons or goal posts needed
Positioning – AIBO position

• Trilateration
  – Uses distance and angle to two landmarks

• Triangulation
  – Bad camera resolution ⇒ bad distance approximations
  – With angles to three landmarks known, we can calculate better distances
Positioning – AIBO position
Positioning – Ball position

- Calculate relative ball position
- Add to current AIBO position
- Does not use MCL
Positioning - Monte Carlo Localization (MCL)

- Makes Positioning more tolerant of input data errors
- Keeps a set of positions with confidences
  - confidence updated with new sensor data
  - position updated with new odometric data
  - "best" current position calculated from this set
Movement
Movement

- Responsible for performing the movements by communicating with the Tekkotsu MotionManager
- Executes the instructions given by the Tactic module
- Calculate and report to the Positioning module how far the AIBO has moved since last time it reported its position
Movement – status

The AIBO is able to:

• walk in any direction by walking forward, backwards, strafing or rotating
• push the ball to the right or left with its head
• kick forward, left or right
• wag the tail
• move the head
• get up if fallen
Movement

Future work:

• Use a faster walking style to move forward

Problems:

• Difficulties with events disappearing in the eventrouter
Tactic
Tactic

The Tactic module shall be able to:

• decide which walking style to use
• decide which kick to use
• choose the best route to a position so the AIBO avoids walking into its team mates
• locate the ball
• tell the Movement module what to do
Tactic

Status:

- The module has all required functionality

Future work:

- The Dijkstra's algorithm will be implemented for obstacle avoidance
Tactic

Problems:

- Since there have been problems with the positioning it has been difficult to test some parts of the module

- The coordinate system for the field is different from any ordinary coordinate system, which has lead to some misunderstandings.
Database
Database

- Provides storage functionality for data
- Replicates data via the network
- All AIBOs have the same replicated data
Database – data distribution
Database – data distribution

Local database
Database – data distribution

Database - middleware
Database – data distribution

Middleware communication
Database – data distribution

Request replication data
Database – data distribution

Reply with replication data
Database – data distribution

Stores and calculates mean values
Strategy
Strategy

- The strategy module coordinates other modules to a playing soccer team.
- Strategic decisions are made locally by each AIBO.
- Local decisions are based on identical data from the database.
- Thanks to the distributed database, a global strategy can be broken down to local decisions made by every AIBO on the field.
- The strategic decisions are then executed by the Tactic module
Strategy

- The strategies are designed visually as automatons, with a language constructed for this purpose.
- These automatons are then exported to XML and later on read by the strategy-engine and translated to an internal representation.
- This makes it possible to change strategy without recompiling.
A minimum of three automatons are required;

- **lineup** – a description of which roles to use under different circumstances.
- **rolecaster** – decides which AIBO will get which specific role.
- **role** – the local strategy for each AIBO (E.g. goalie or attacker)
Debug Monitor

Need to visualize information about

- Positioning
- Database
- Strategy
Debug Monitor - Positioning
### Debug Monitor - Database

<table>
<thead>
<tr>
<th>LOCAL DATABASE</th>
<th>REPLICATED DATABASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ball</td>
<td>AIBO3_ball</td>
</tr>
<tr>
<td>ballrel</td>
<td>AIBO3_ballrel</td>
</tr>
<tr>
<td>ballvis</td>
<td>AIBO3_ballvis</td>
</tr>
<tr>
<td>pos</td>
<td>AIBO3_pos</td>
</tr>
<tr>
<td>tacticdone</td>
<td>AIBO3_tacticdone</td>
</tr>
<tr>
<td>timeoutreached</td>
<td>AIBO3_timeoutreached</td>
</tr>
<tr>
<td>globalball</td>
<td>AIBO3_globalball</td>
</tr>
<tr>
<td>[210, 97]</td>
<td>[111, 111]</td>
</tr>
<tr>
<td>[72, -106]</td>
<td>[27, 27]</td>
</tr>
<tr>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>[143, -8]</td>
<td>[139, 139]</td>
</tr>
<tr>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>[111, 111]</td>
<td>[210, 210]</td>
</tr>
</tbody>
</table>
Debug Monitor - Strategy

<table>
<thead>
<tr>
<th>State</th>
<th>Target</th>
<th>Guard</th>
<th>Action</th>
<th>Timeout</th>
</tr>
</thead>
<tbody>
<tr>
<td>State::Not seeing...</td>
<td>State::Kicking_ball</td>
<td>closeToBall</td>
<td>kick</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>State::Locating_ball</td>
<td>State::Going_to_ball</td>
<td>timeoutReached</td>
<td>gotoBall</td>
<td>timeout</td>
</tr>
<tr>
<td>State::Locating_ball</td>
<td>State::Seeing_ball</td>
<td>tacticDone</td>
<td>&lt;no action&gt;</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>State::Seeing_ball</td>
<td>State::Going_to_ball</td>
<td>true</td>
<td>gotoBall</td>
<td>timeout</td>
</tr>
<tr>
<td>State::Going_to_b...</td>
<td>State::Kicking_ball</td>
<td>tacticDone</td>
<td>kick</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>State::Going_to_b...</td>
<td>State::Start</td>
<td>timeoutReached</td>
<td>&lt;no action&gt;</td>
<td>&lt;none&gt;</td>
</tr>
</tbody>
</table>

role: attacker state: State::Going_to_ball

History

play State::Going_to_ball
sending states
state State::Going_to_ball State::Kicking_ballAss [tacticDone] kick
state State::Going_to_ball State::Start [timeoutReached]
predicate tacticDone 0
predicate timeoutReached 0
predicate tacticDone 0
predicate timeoutReached 0
predicate tacticDone 0
predicate timeoutReached 0
predicate tacticDone 0
predicate timeoutReached 0
predicate tacticDone 0
predicate timeoutReached 0
predicate tacticDone 0
play State::Start

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Value</th>
<th>Last updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAmGoalie</td>
<td>0</td>
<td>Thu Jan 13 11:28:32 CET 2005</td>
</tr>
<tr>
<td>closestToBall</td>
<td>1</td>
<td>Thu Jan 13 11:28:32 CET 2005</td>
</tr>
<tr>
<td>ballVisible</td>
<td>0</td>
<td>Thu Jan 13 11:28:35 CET 2005</td>
</tr>
<tr>
<td>closeToBall</td>
<td>0 (distance 23.7697)</td>
<td>Thu Jan 13 11:28:36 CET 2005</td>
</tr>
<tr>
<td>timeoutReached</td>
<td>0</td>
<td>Thu Jan 13 11:28:36 CET 2005</td>
</tr>
<tr>
<td>tacticDone</td>
<td>0</td>
<td>Thu Jan 13 11:28:36 CET 2005</td>
</tr>
</tbody>
</table>
Simulator

- Test environment for strategies
  - Simulate Database interface
  - Simulate Tactic interface
- Simplified representation
- Perfect database
Simulator - description

- Simulator
  - Main class
  - Field
  - Ball
  - DB
  - GC
  - Strategy
  - Tactic
  - Player

- SimGUI
  - User interface
Questions ?
Code reusage

- Team Dynamo Pavlov 2002 did not use Tekkotsu
- Team Gifr 2003 used incompatible Tekkotsu release, difficult to continue development
- Old solutions were studied although no actual code was reused
- Several modules that no previous team had attempted
Summary

• Despite not complete functionality, we learned a lot about the following:
  
  – Large projects
  – Working in an unfamiliar environment
  – Teamwork
  – Documentation
  – System design
  – Fancy presentations
  – How to endure a dozen cups of coffee a day :-(
The End