CiberRato Competition and Tools

Robótica Móvel e Inteligente
IEETA/IRIS Lab – DETI – Universidade de Aveiro

Outline

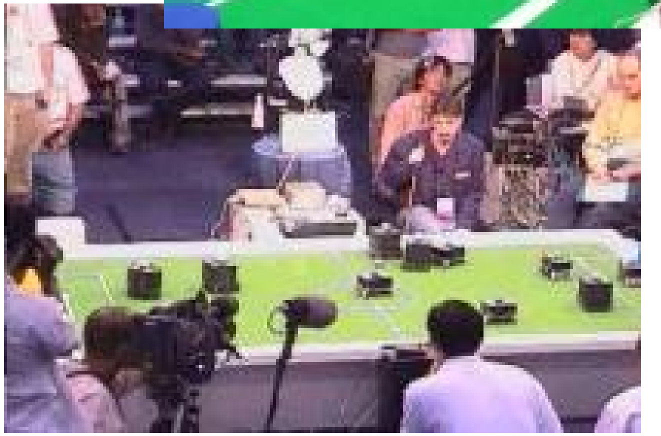
- Robotic Competitions
- Micro-Rato
- Ciber-Rato
- Demos
- Installing the tools

Motivation for robotic competitions

- Use robotic competitions to foster the development of scientific/engineering skills
- Mobile Robotics allows for the use of many technical concepts
- Learn by doing…

Robotic Competitions

- RoboCup
  - Soccer
  - Rescue
  - Dance (Junior)
- AAAI
  - Robot attending conference
  - Rescue
  - Navigation and interaction
- ...



Robotic Competitions in Portugal

- Micro-Rato Competition
  - Solve a maze
- Festival Nacional de Robótica
  - Autonomous Driving
  - Robotic Soccer
  - Other Challenges
- Fire Fighting Robot
- RoboParty
- Micro Mouse

Micro-Rato Contest

- First Robotic Competition in Portugal
  - First edition in 1995, 16th edition in 2011
- Organized by the University of Aveiro
- Challenge:
  - Build a autonomous mobile robot that can navigate in an unknown maze
- Targeted at high school and graduation students
- 3 modalities:
  - Micro-Rato Competition (since 1995)
  - Ciber-Rato Competition (since 2001)
  - PathFinder (since 2013)

Micro-Rato environment

Micro-Rato Challenge

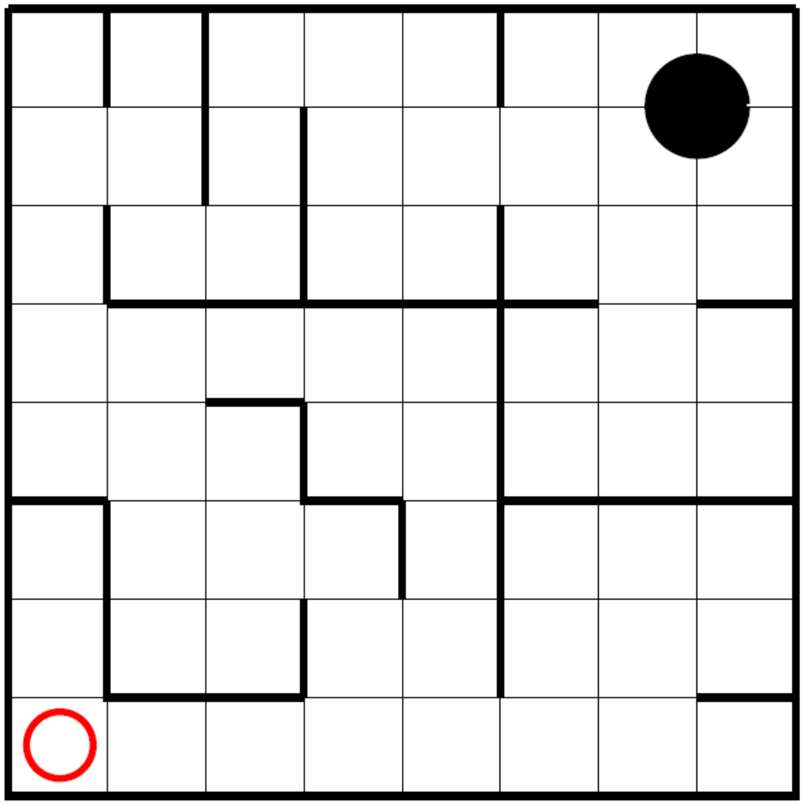
- Go to BEACON Area:

  Go from START to BEACON within a time limit (3 min) (1995-2000)

- Return:

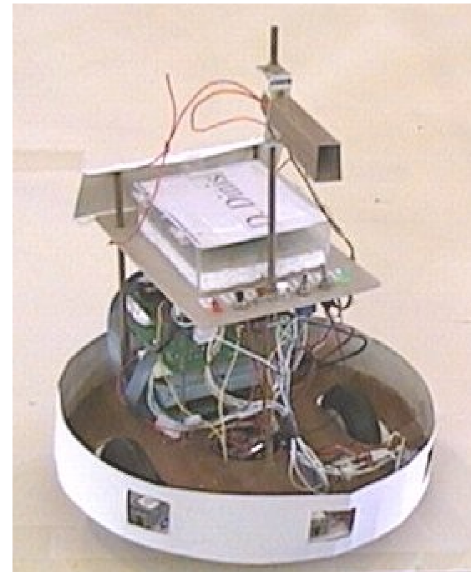
  Go from START to BEACON within a time limit (3 min) (1995-2000)

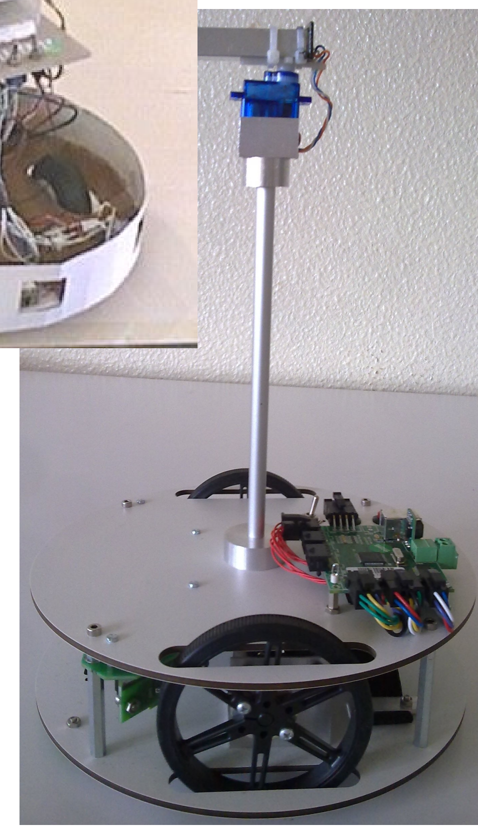
- Score depends on
  - Distance to travel
  - Penalties



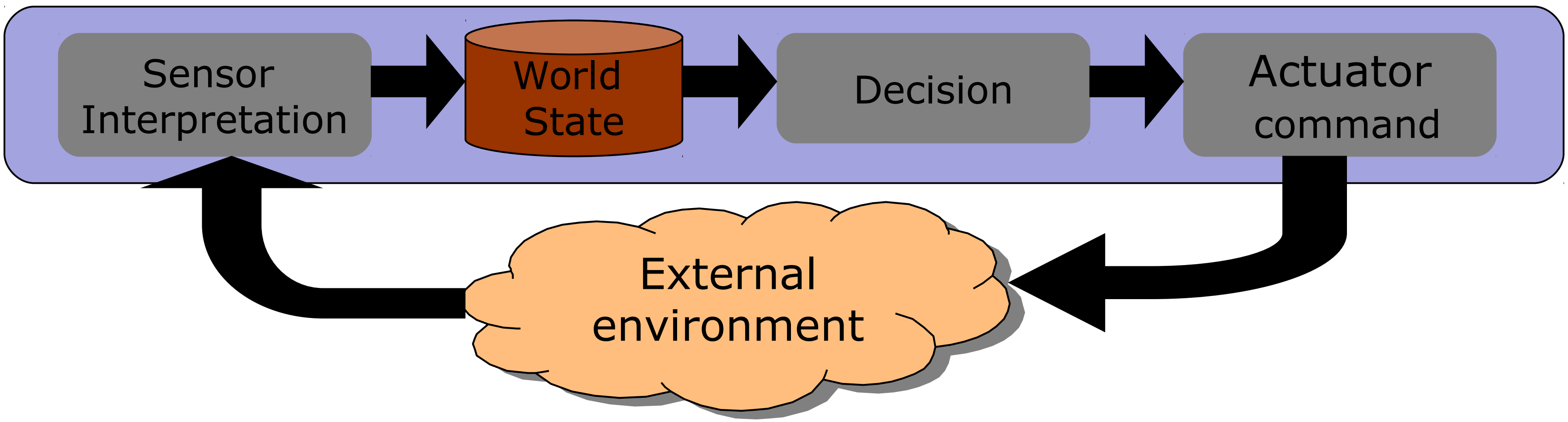
Building a Robot

- Hardware
  - Sensors (obstacles, ground, beacon, encoder, ...)
  - Actuators (locomotion, leds)
  - Processing unit
- Software
  - Sensory data interpretation
  - Decision
  - Actuator command





Building a Robot *Software*



Depending on the complexity of World State and Decision Algorithms

- Reactive architecture
- Deliberative architecture
- Hybrid architecture

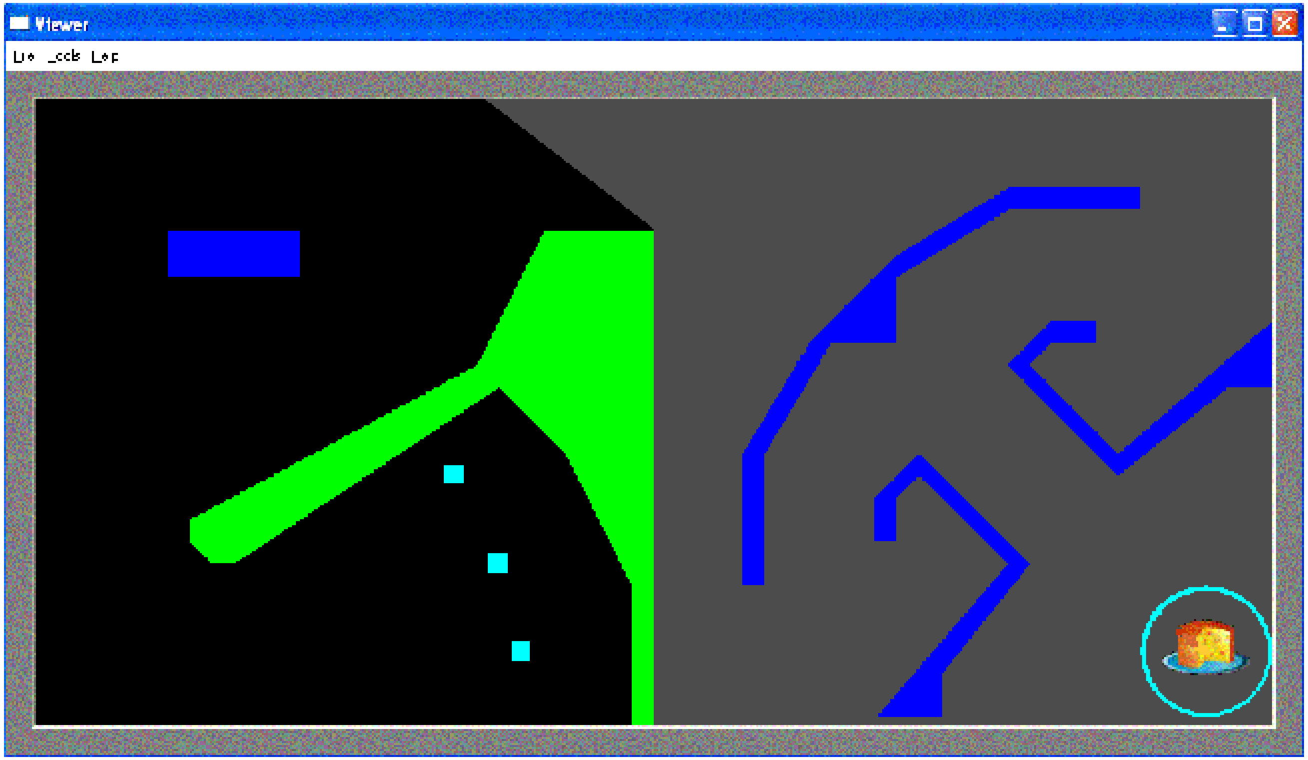
Ciber-Rato

- Micro-Rato in a simulated environment
  - Simulator implements maze and bodies of virtual robots
  - Virtual sensors and motors are **identical for all robots**
- Participants develop the robot brain (control software)
  - Brains take decision **autonomously** (software agents)
- Focus on software components of robotics

http://sourceforge.net/projects/cpss/
http://microrato.ua.pt

Ciber-Rato: the environment

- Maze with high and low walls, a starting grid, one or more beacon-areas and a home-area



CiberRato: Competitive Challenge

- Initial State: Robots are placed at starting grid
- First Objective: Get into BEACON area; if there is more that one Beacon all areas must be visited
- Second Objective: Return to the closest point to departure in the least possible time
- **Score** = Distance to starting point
  + Excess of returning time
  + Penalties
    - Collisions
    - Lack of fulfillment of first objective
    - Not signaling finish

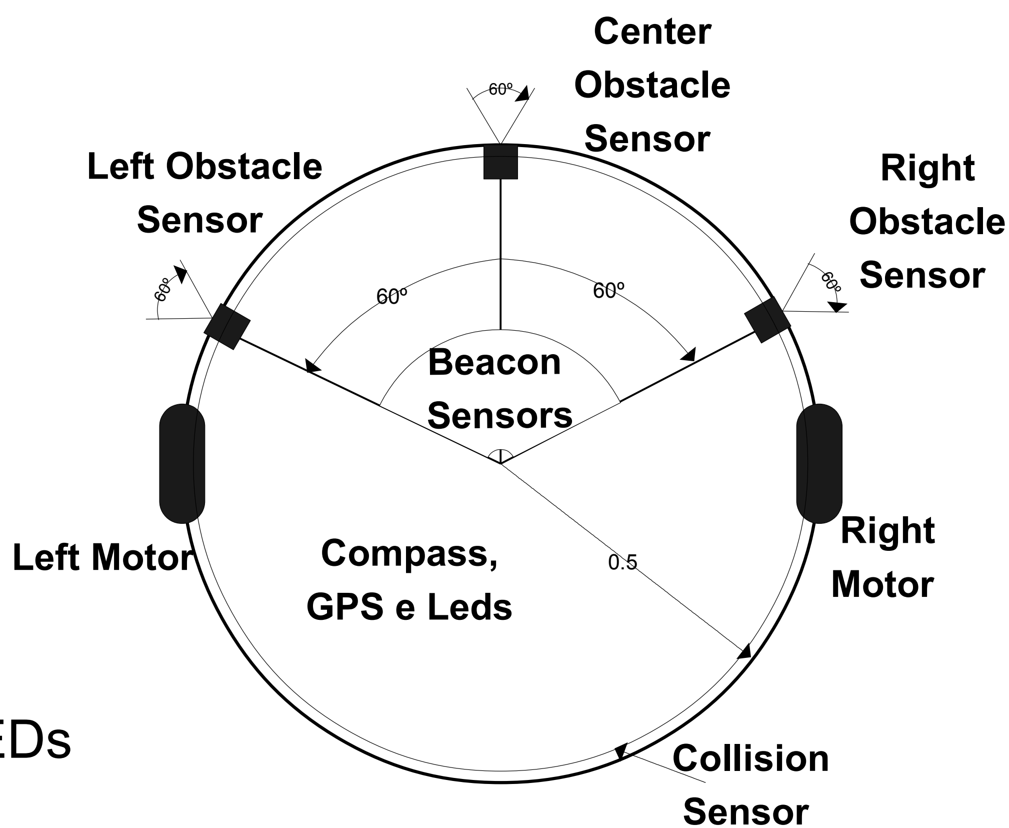
CiberRato Cooperative Challenge

- Rules changed to increase links with sensor networks and cooperative multi-agent systems research
  - More agents: 5
  - Agents can communicate, but communication distance is limited
  - GPS may be used; has noise
  - Target: All agents must reach the beacon area in minimal time
  - Target 2: All agents must return to a home area (since 2011)
- Robots must cooperate to achieve challenge!!!

- Firstly used in CiberMouse@RTSS2008, Barcelona, Spain
- CyberRescue@RTSS2009 in Washington DC, USA
- CiberRato 2009-2015, Aveiro

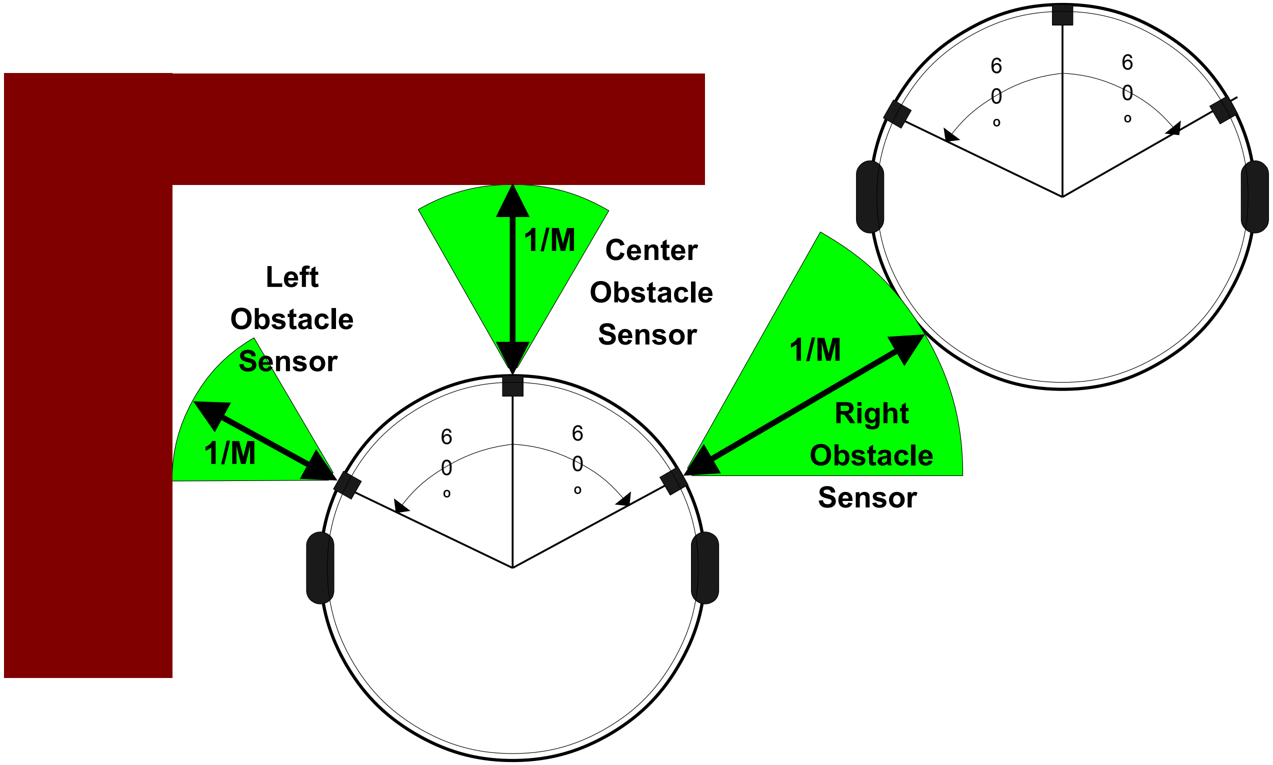
Virtual Robot

- The Virtual Robot is equipped with:
  - Sensors
    - Obstacles
    - Beacon
    - Ground
    - Collisions
    - Compass
    - GPS (debug)
  - Actuators
    - 2 Motors
    - Some signaling LEDs



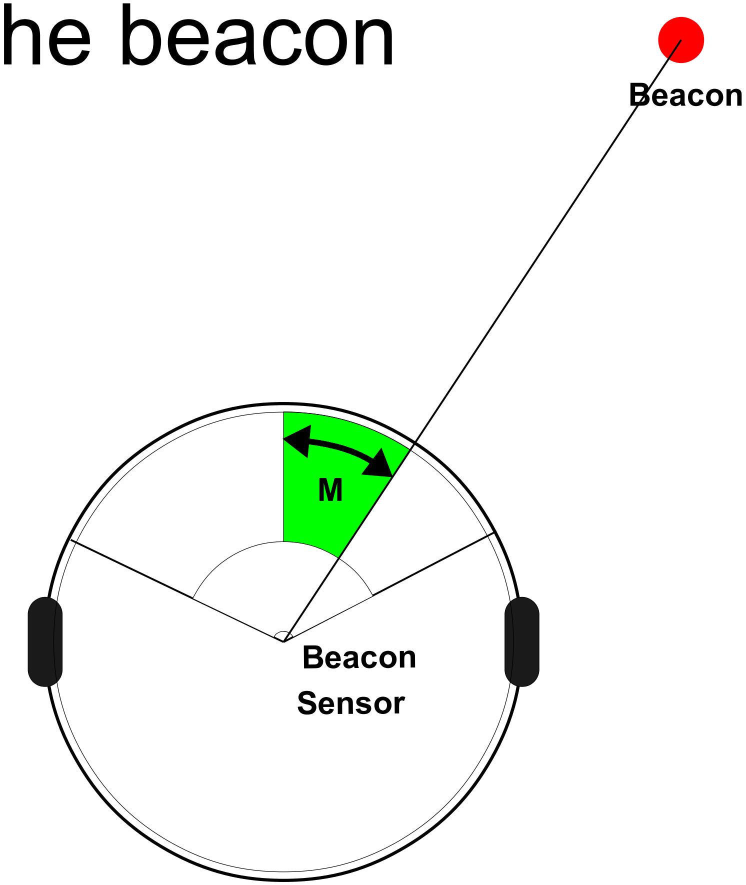
Obstacles sensor

- Measure is inversely proportional to distance to closest obstacle



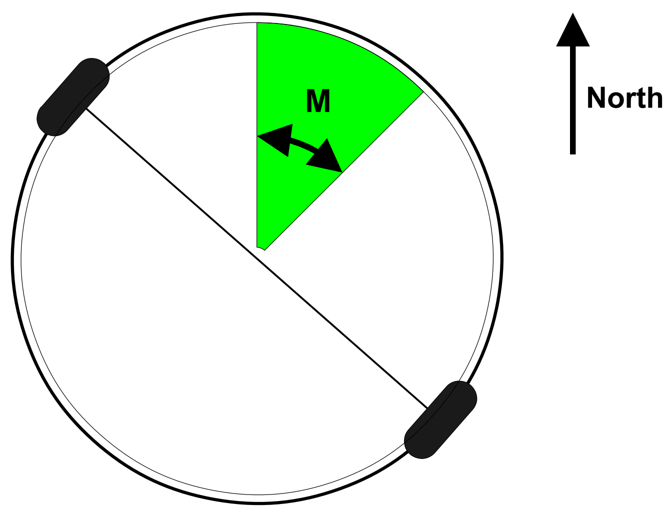
Beacon sensor

- Measure is the angle from the front of the robot to the beacon



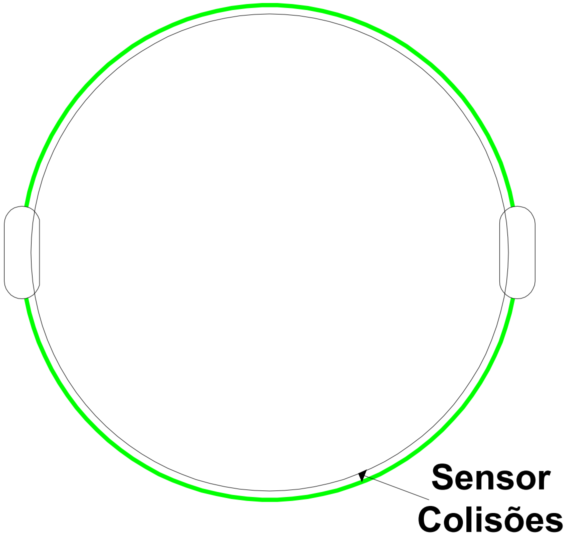
Compass

- Measure is the angle from the Virtual North to the front of the robot



Collision sensor

- Binary Sensor active when the robot collides



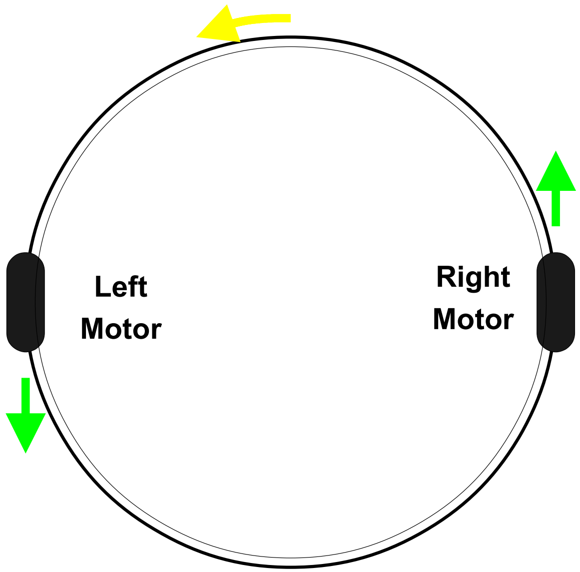
Sensor requests

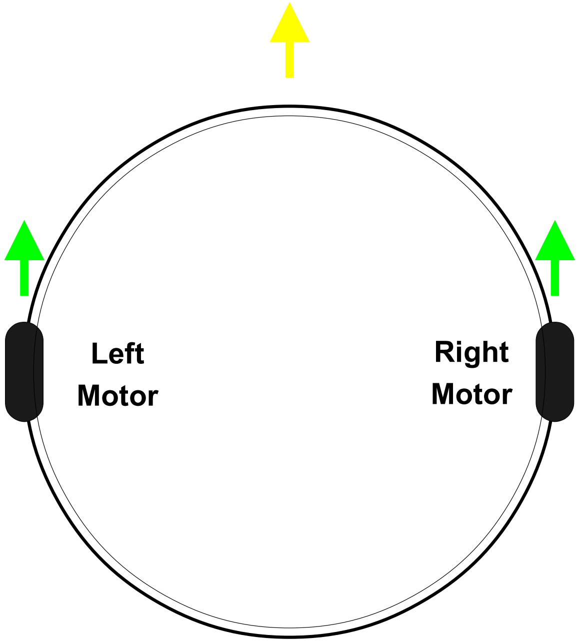
- Starting at CiberMouse@RTSS2006 the number of sensors that is received in each cycle is limited
- Agents must request the sensor measures they want to receive in the next cycle
- Also the latencies of sensors have been increased

Sensor	Range	Resolution	Noise type	Deviation	Latency	On request
Obstacle s.	$[0.0, 100.0]$	0.1	aditive	0.1	0	yes
Beacon s.	$[-180, +180]$	1	aditive	2.0	4	yes
Compass	$[-180, +180]$	1	aditive	2.0	4	yes
Bumper	Yes/No	................N/A................			0	no
Ground s.	Yes/No	................N/A................			0	yes

Motors

- Agents control robot movement defining motor speeds of two independent motors





Actuators

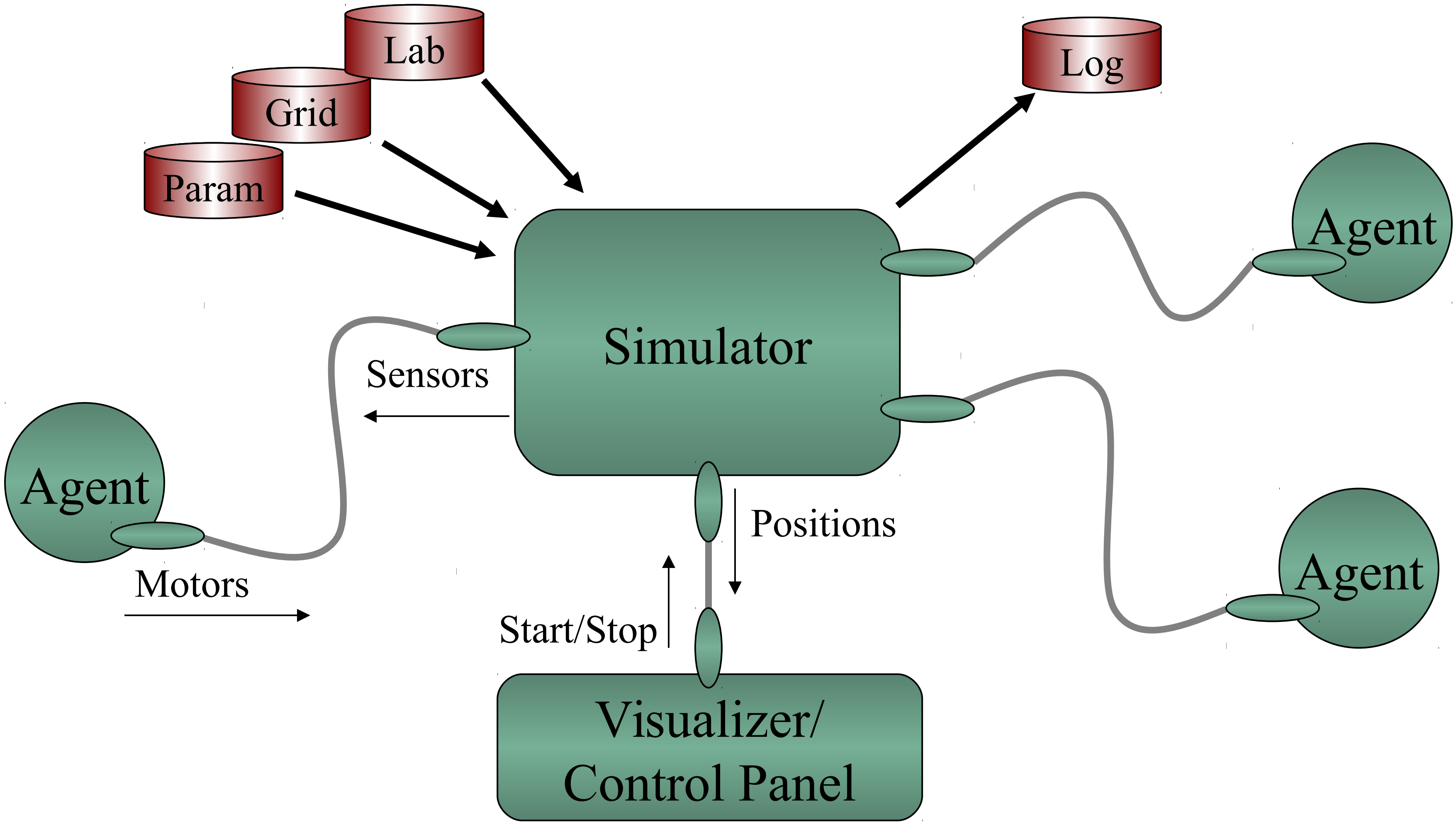
- Motors: to drive the robot
  - Simulator adds noise to actuation commands
  - Motor model uses inertia
- Leds: to signal specific events

Actuator	**Range**	**Resolution**	**Noise type**	**Standard deviation**
Motor	$[-0.15, +0.15]$	0.001	multiplicative	1.5%
End led	On/Off	.... N/A		
Return led	On/Off	.... N/A		
Beacon led	On/Off	.... N/A		

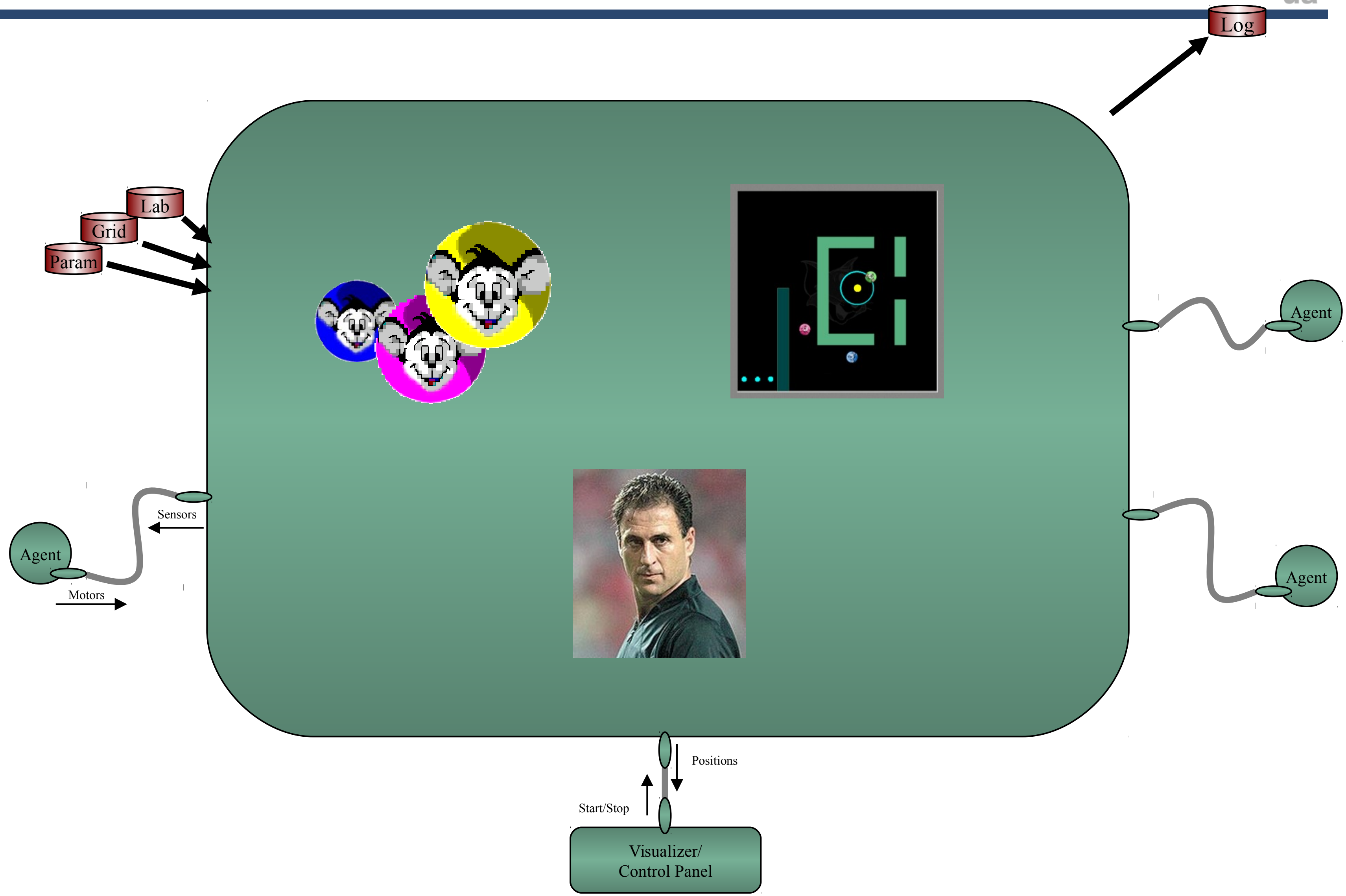
General Architecture

- Client-server distributed system
  - Server: Simulator
  - Clients: Agents and Visualization Tools
- Communication using UDP Sockets
  - XML Messages
- Configuration and Log Files in XML
  - Maze, starting grid, simulation parameters

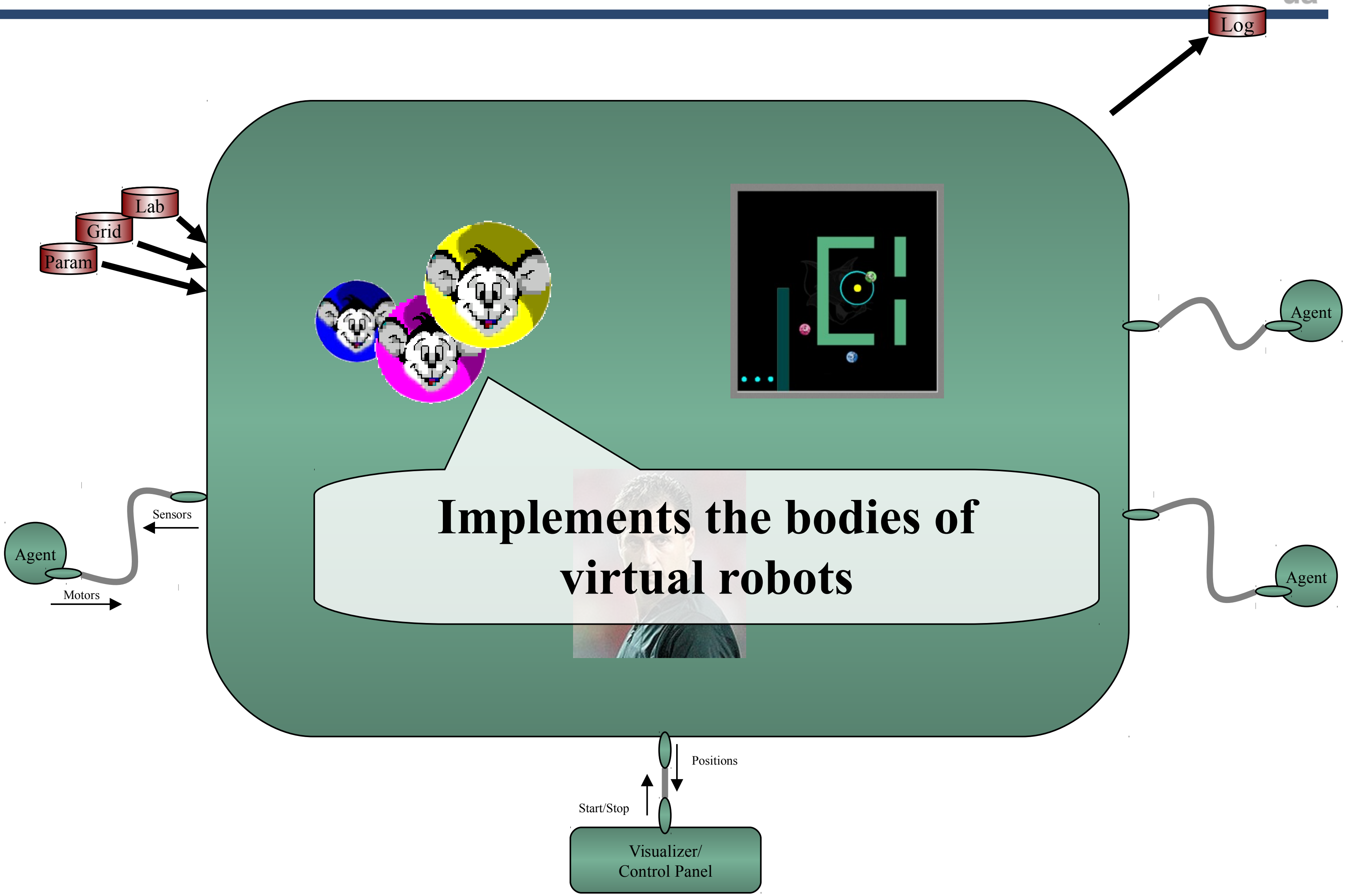
General Architecture



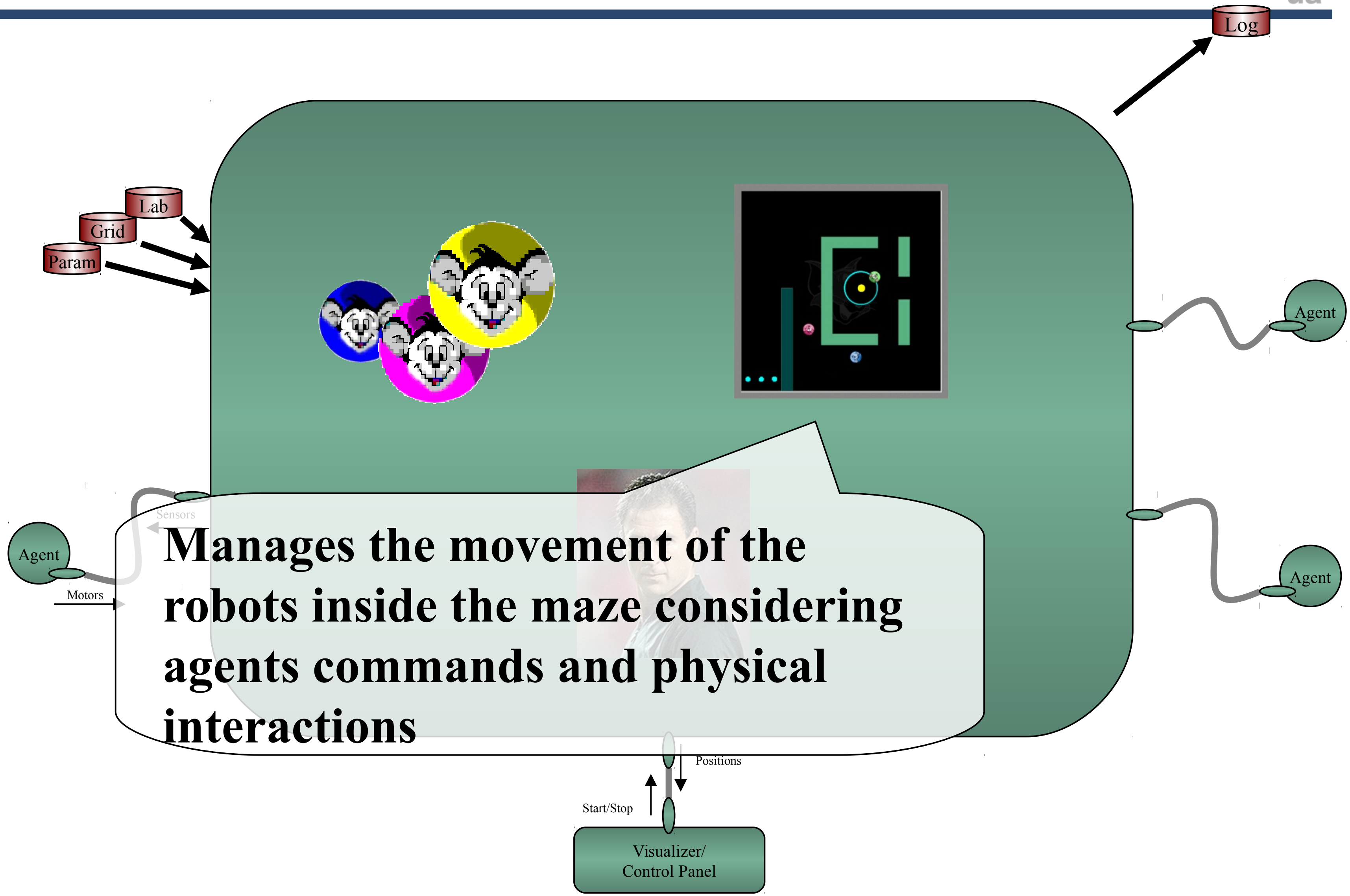
Simulator


Log
Lab
Grid
Param
Agent
Agent
Sensors
Agent
Motors
Agent
Positions
Start/Stop
Visualizer/
Control Panel

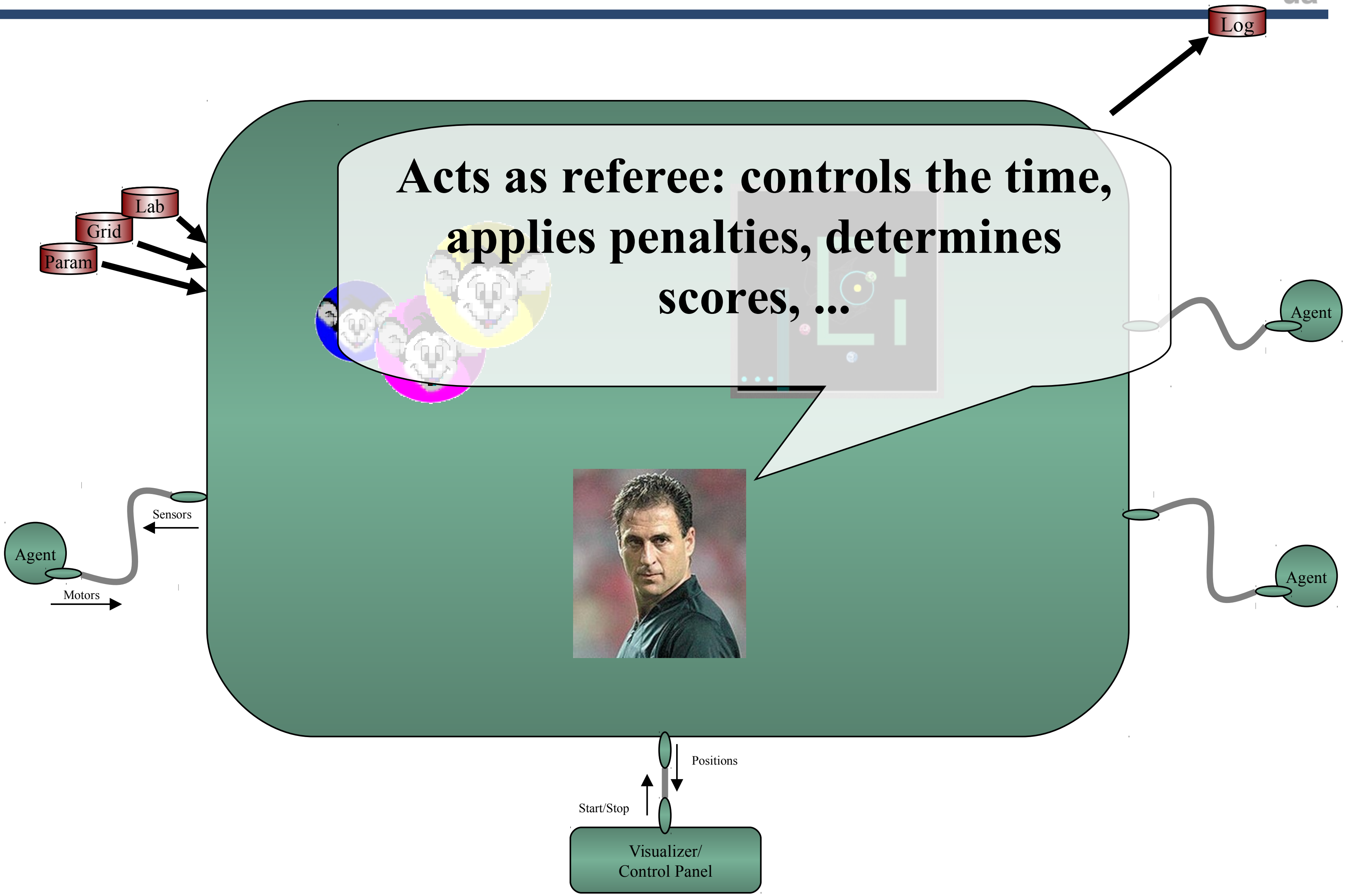
Simulator


Log
Lab
Grid
Param
Agent
Sensors
Motors
Agent
Agent
Implements the bodies of virtual robots
Positions
Start/Stop
Visualizer/
Control Panel

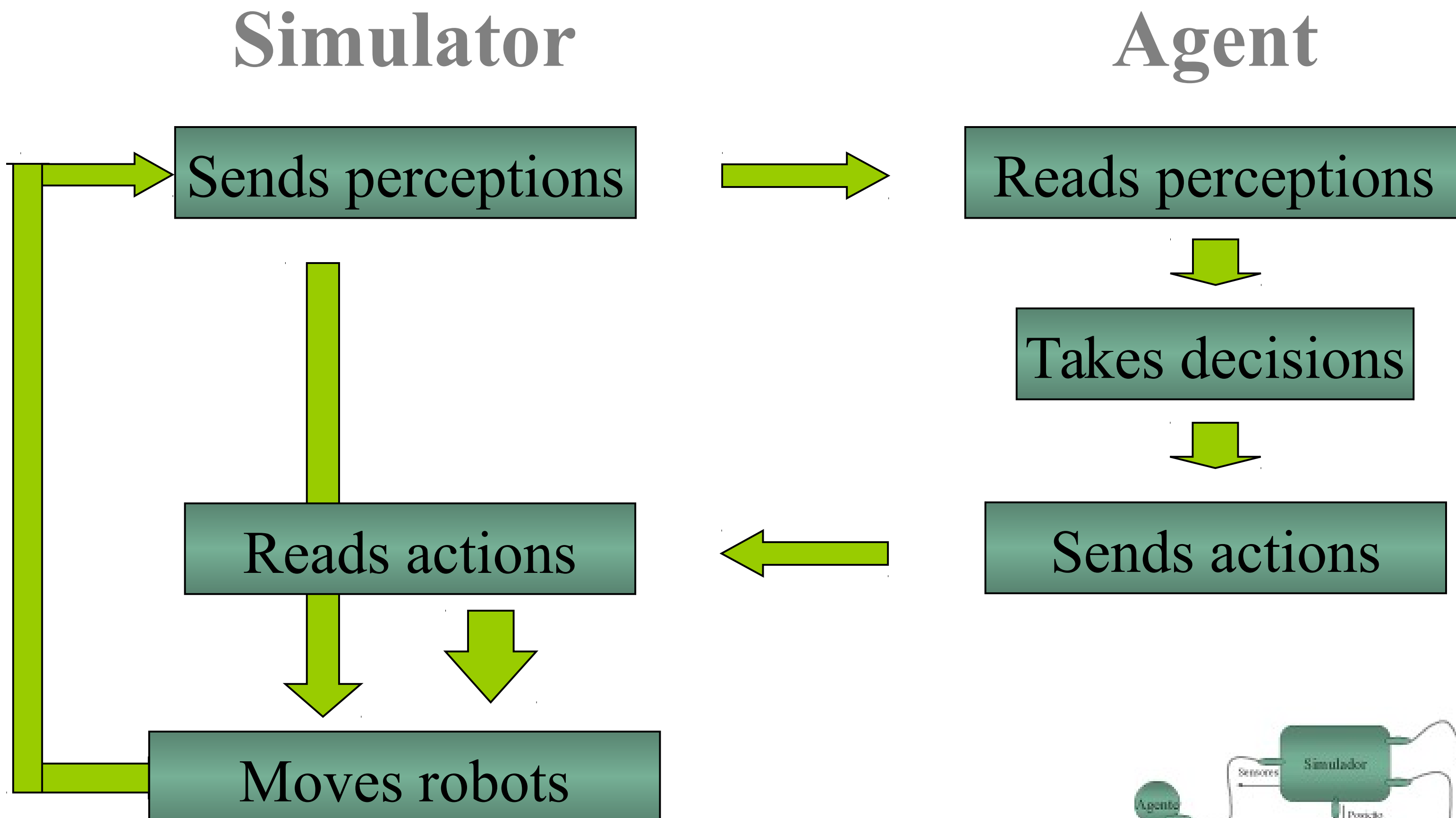
Simulator

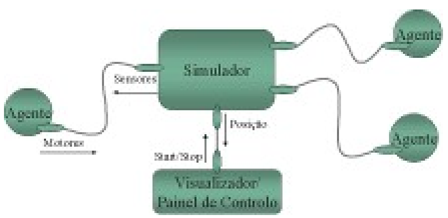

Log
Lab
Grid
Param
Agent
Agent
Agent
Sensors
Motors
Manages the movement of the robots inside the maze considering agents commands and physical interactions
Positions
Start/Stop
Visualizer/
Control Panel

Simulator


Log
Lab
Grid
Param
Acts as referee: controls the time, applies penalties, determines scores, ...
Agent
Sensors
Motors
Agent
Agent
Positions
Start/Stop
Visualizer/
Control Panel

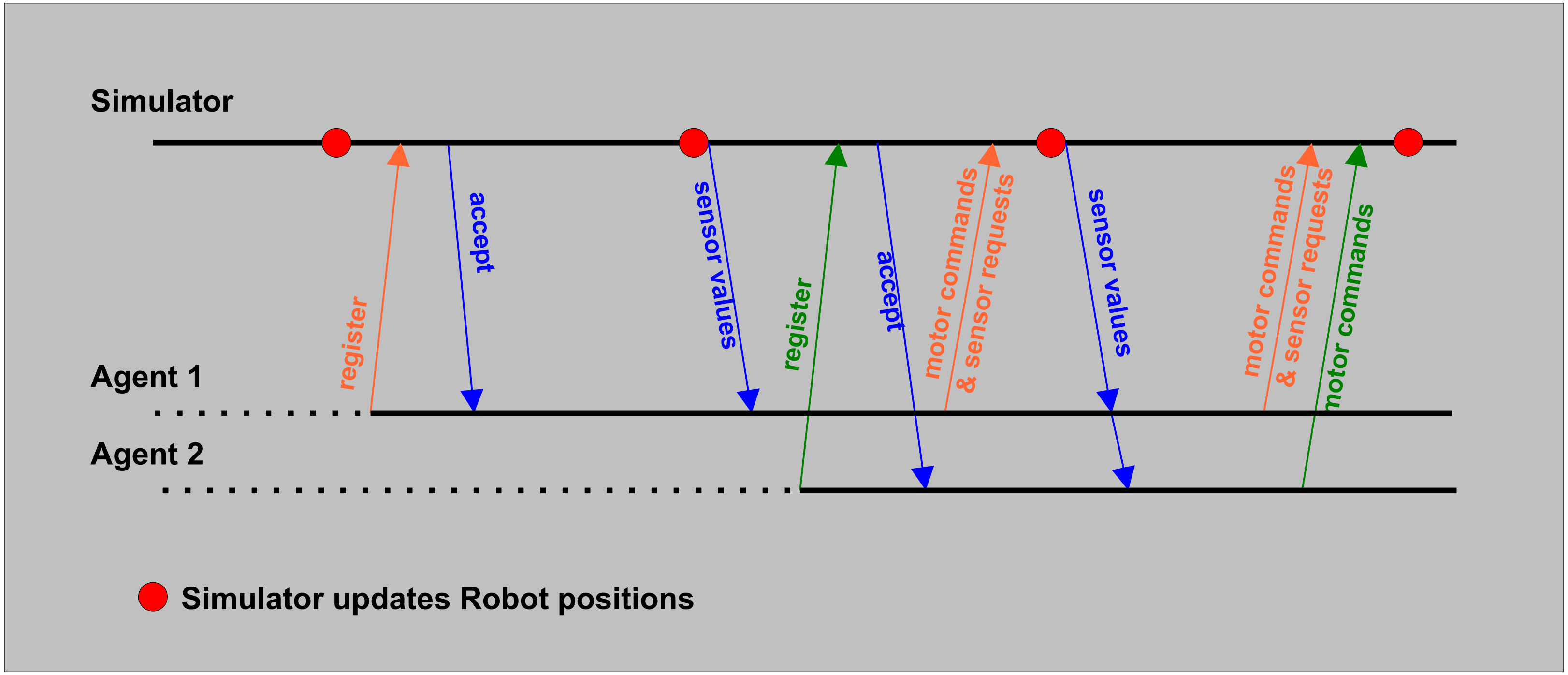
Simulator-Agent interaction



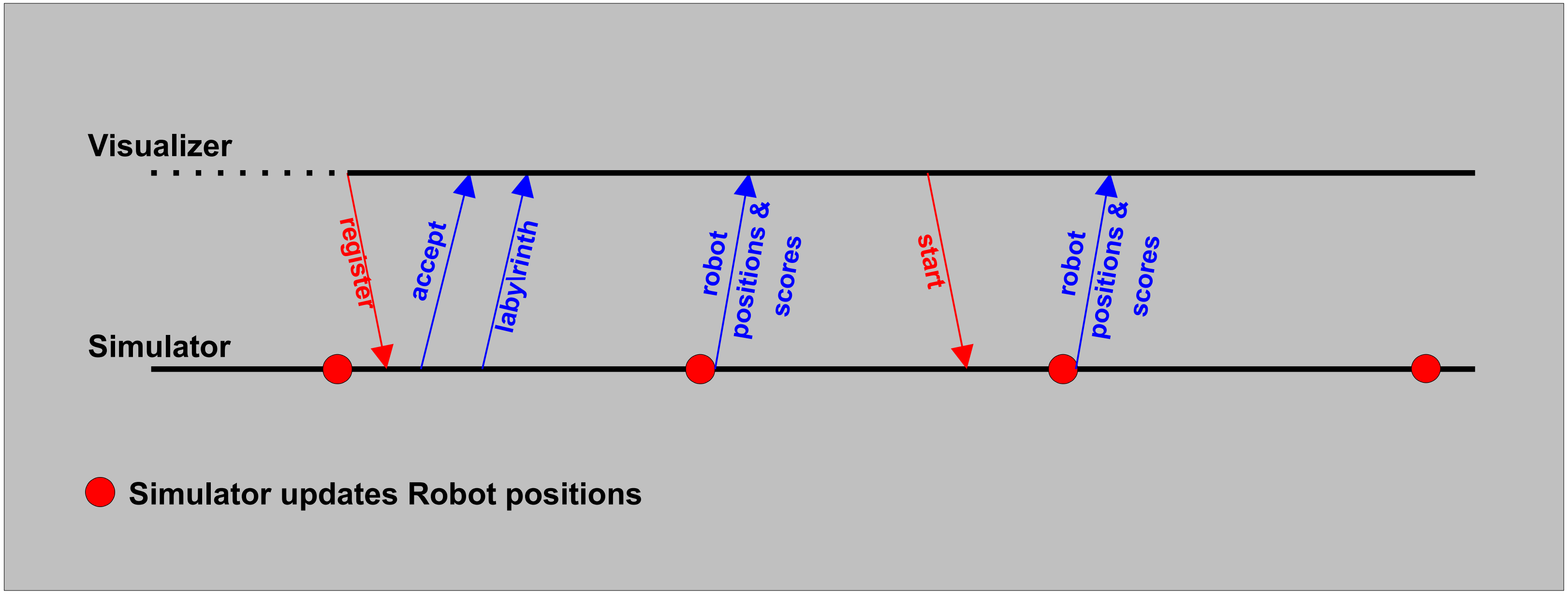


Time is discrete

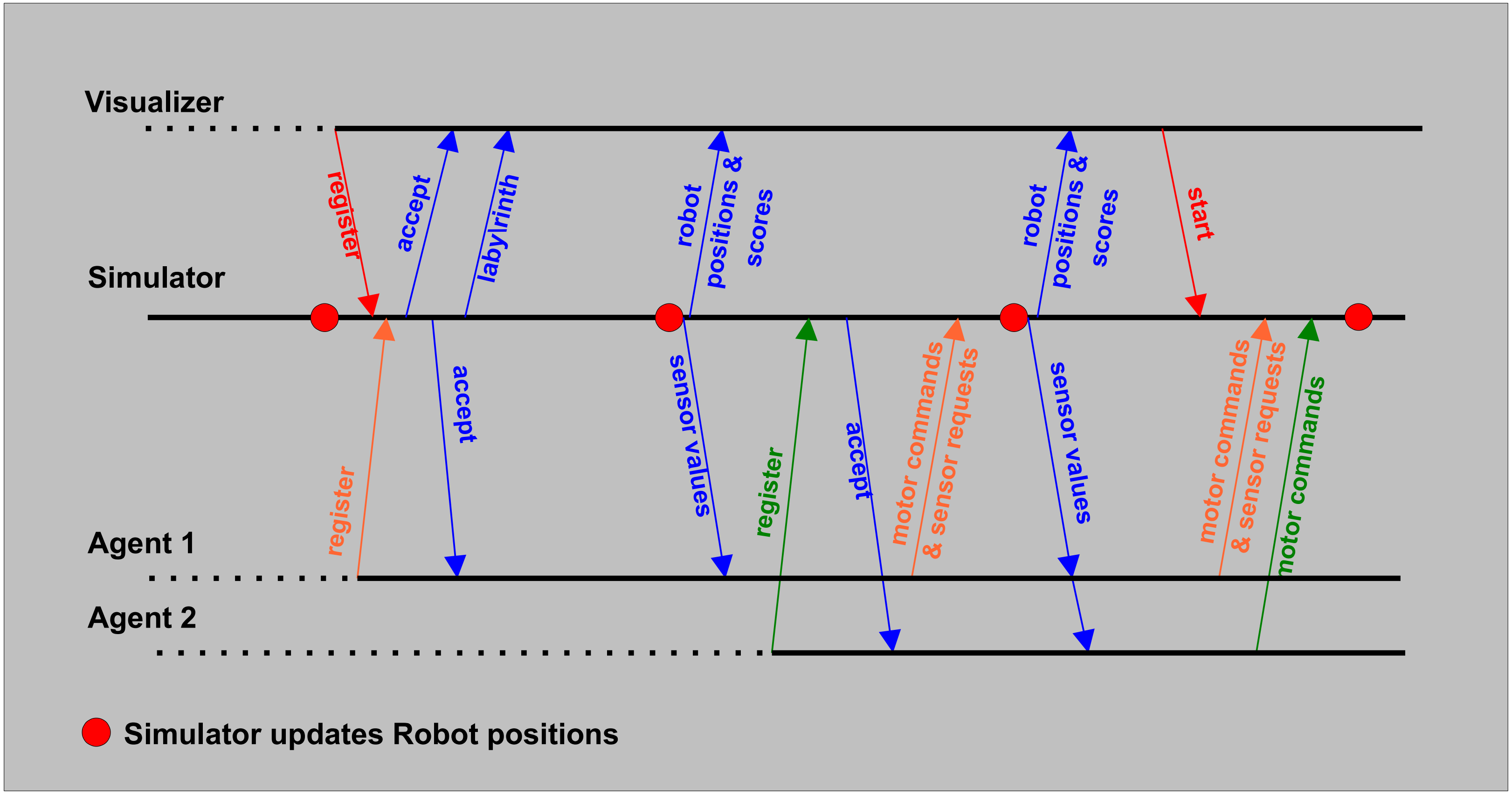
Simulator-Agent interaction



Simulator-Visualizer interaction



Simulator operation



Movement model

- First order approximation determines linear and rotational displacement of robot

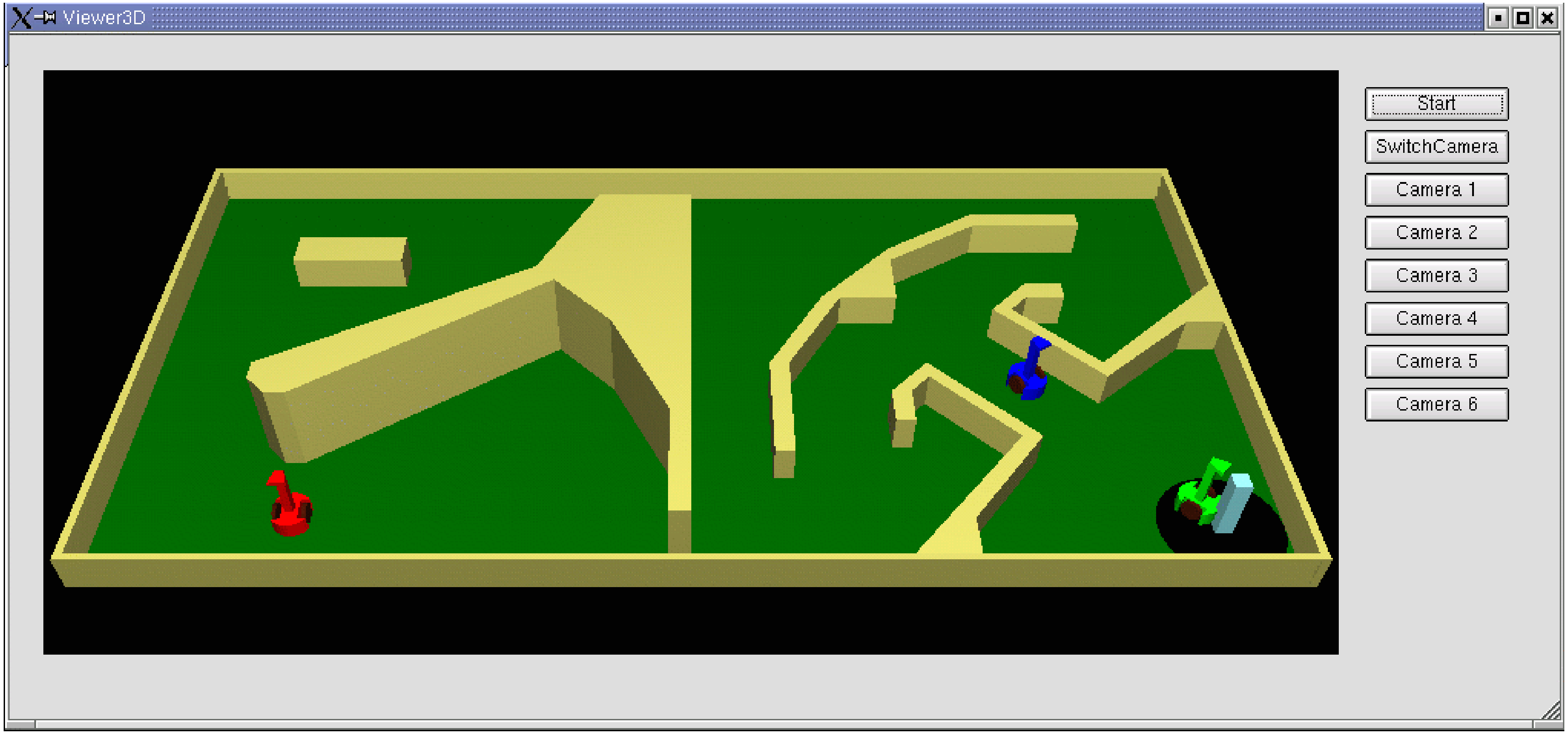
$$lin = \frac{out_{right} + out_{left}}{2} \qquad rot = \frac{out_{right} - out_{left}}{diam}$$

- Inertia is simulated through a IIR filter

$$out_t = (in_t * 0.5 + out_{t-1} * 0.5) * noise$$

Visualization

Visualization



Agents

- Programmed in any language/OS
  - Communication libraries in C, Prolog, Java, and Visual Basic are available
  - Participants: Windows/Linux, C/C++/Delphi/VB
- C reactive agent is part of the published tools
- Some development support tools are also available
- Binaries from previous years are available for testing purposes

Reactive agent

- Avoiding obstacles

```
while(1)
{
    ReadSensors();
    RequestSensors();
    left  = GetObstacleSensor(LEFT);
    right = GetObstacleSensor(RIGHT);
    center= GetObstacleSensor(CENTER);
    if(center>4.5 || right>4.5 || left>4.5 )
        DriveMotors(0.06,-0.06); /* Rotate*/
    else
        DriveMotors(0.1,0.1);    /* GO */
}
```

Challenges in Ciber-Rato

- Agent Architecture
  - How should the agent be structured?
  - Which are the modules of the agent?
- Localization
  - Where is the robot?
- Mapping
  - Build a map of the maze from past experience
  - Find the beacon position
- Path planning
  - Find the shortest path to the next beacon, or to the starting position
- Sensor Fusion and communication
  - Best estimate from several sensors and communicated info

Challenges in Ciber-Rato

- Navigation
  - Avoid obstacles
  - Detect cycles
- Plan execution and monitoring
  - Which is the strategy?
  - Explore or return?
  - Is the planned path possible?
- Agent development/debugging/tuning
  - Tools to foster the development