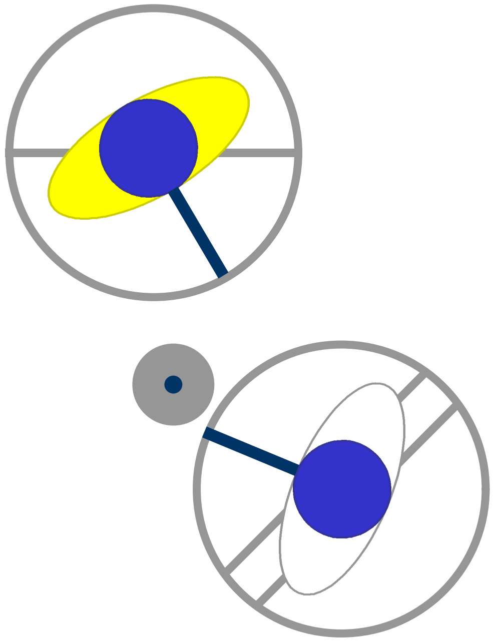
AIsHockey—An Introduction

Jouni Smed

2002

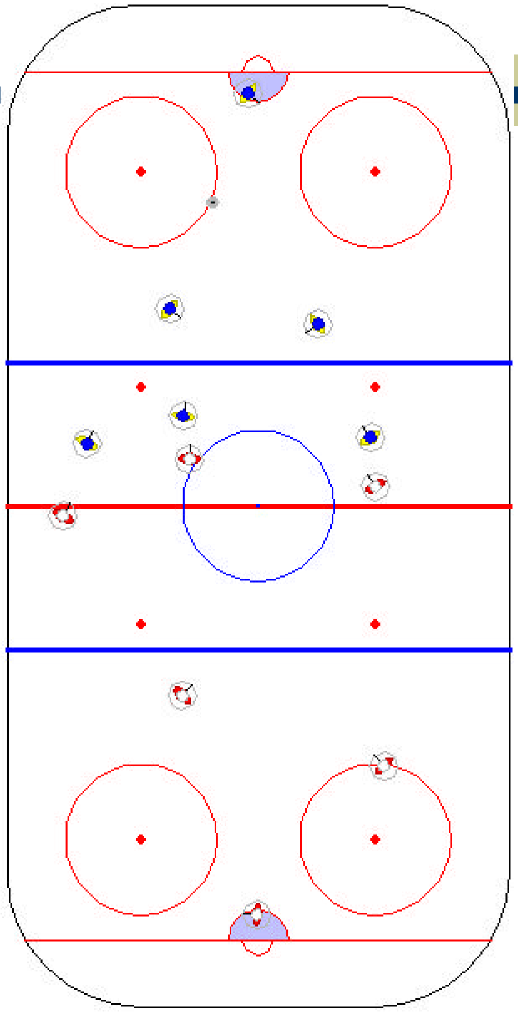
jouni.smed@cs.utu.fi

http://staff.cs.utu.fi/staff/jouni.smed/

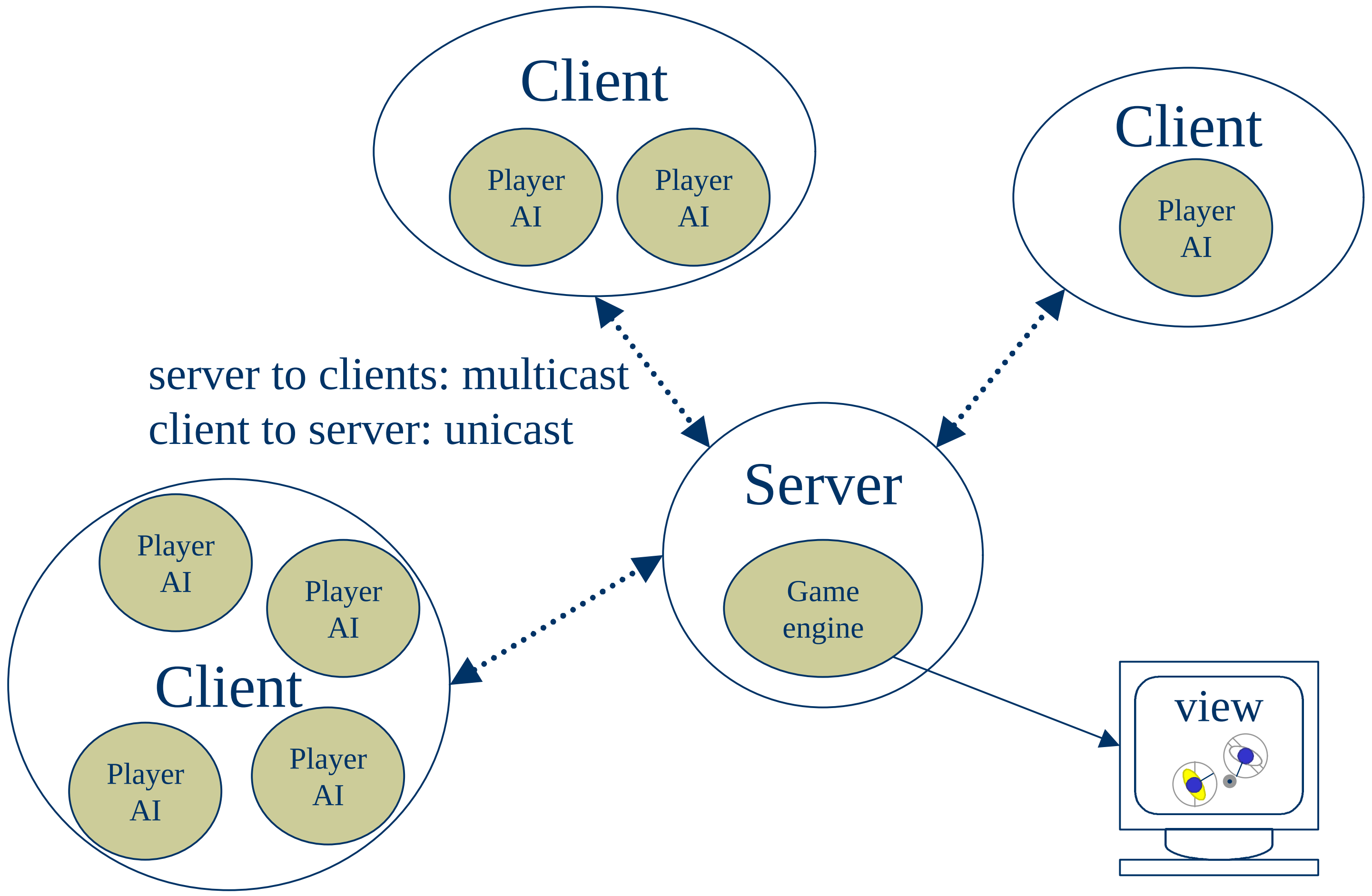


What is AIsHockey?

- simplified ice hockey
  - IIHF rules: `http://www.iihf.com/`
  - game engine checks goals, offsides, icings, interfering the goalie
  - no penalties
- distributed system
  - server = game engine
  - client(s) = player AI(s)
- the challenge: implement a team of autonomous, real-time AIs

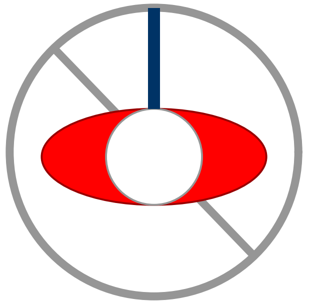


Client/server model

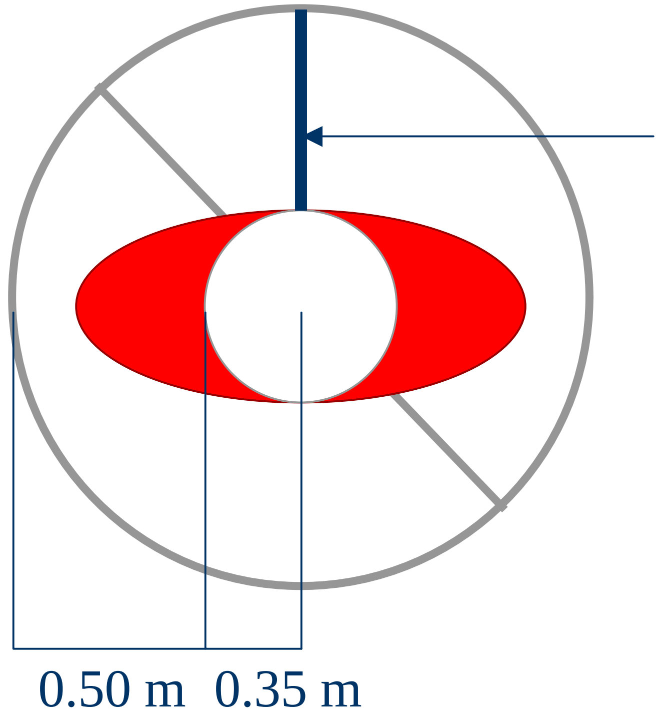


Game engine physics

- player: m = 75 kg, r = 0.35 m
- dash forwards or backwards (i.e., brake)
- skates: friction depends on orientation
- player can change heading to any angle
- shooting from an operating range (0.50 m)
- keeping the puck (only goalies)
- communication (64 bit messages)



The player



heading in radians:

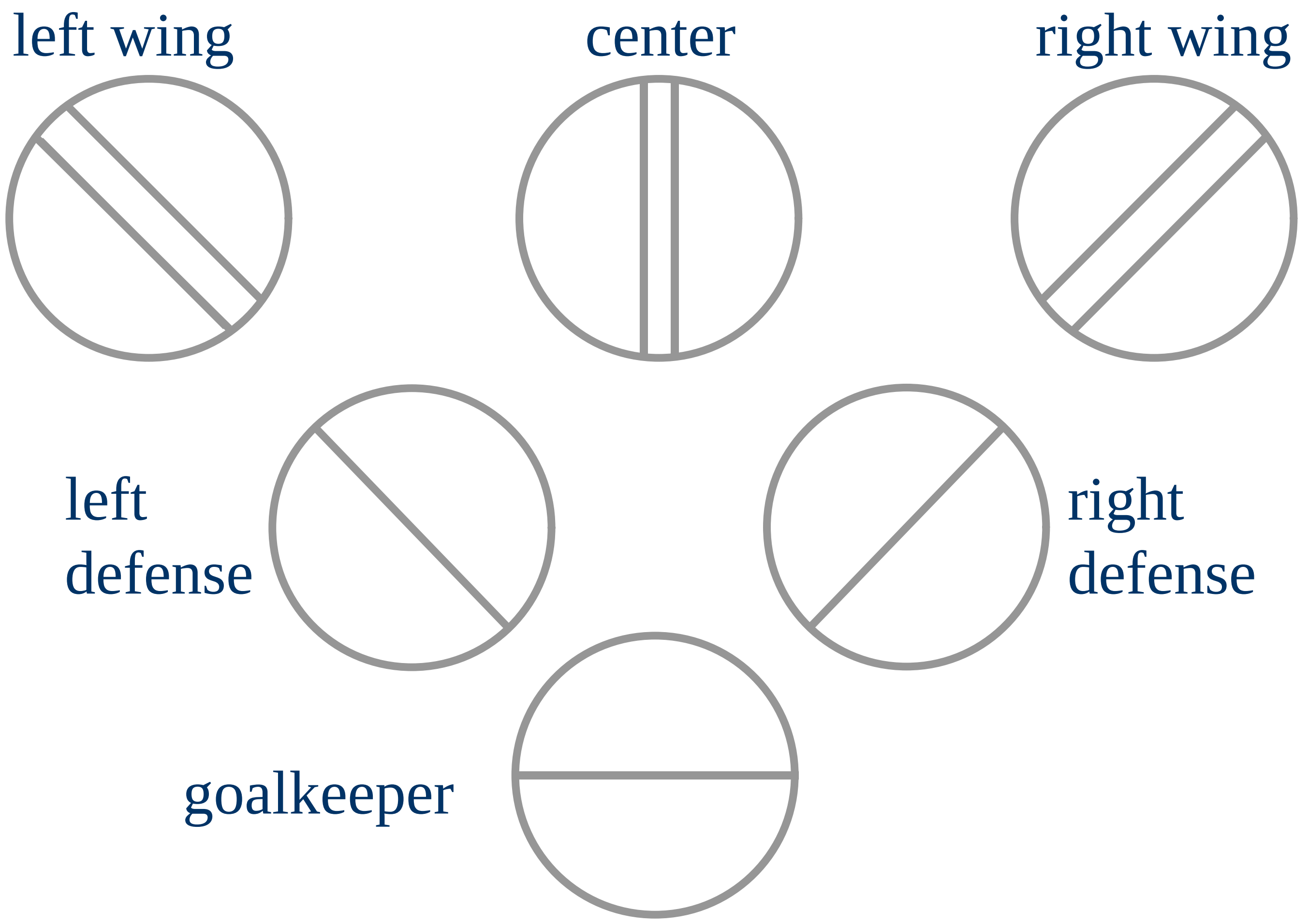
0: towards opponent's end (↑)

π/2: to the right (→)

-π/2: to the left (←)

π: towards own end (↓)

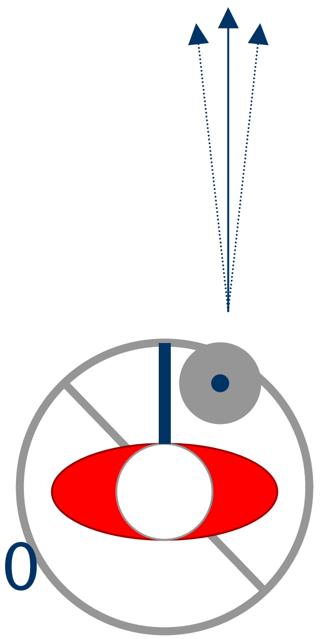
Player symbols



Implementing an AI

- package `fi.utu.cs.hockey.ai`
  - inherit the class `AI` and implement the abstract method `react()`
  - useful constant values in the class `Constants`
- package `fi.utu.cs.hockey.net`
  - interface `Communication` defines the inputs and outputs

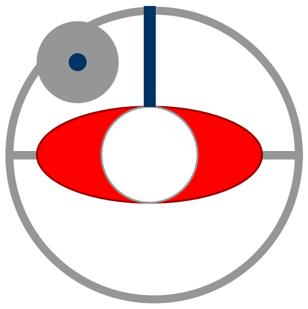
Methods 1 (3)



shoot(p)

0.0 <= p <= 1.0

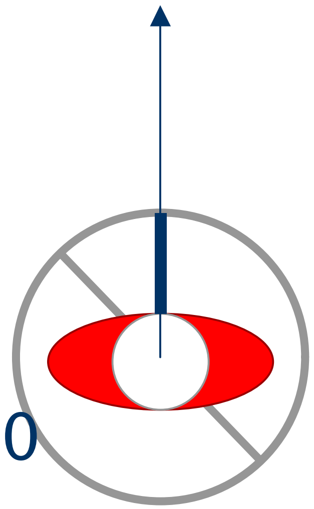
keepPuck()



Methods 2 (3)

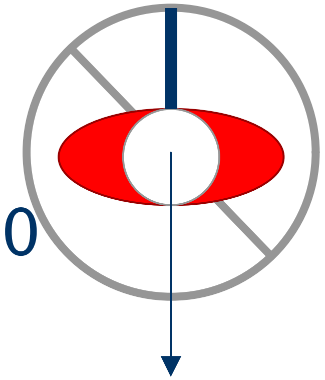
dash(p)

0.0 <= p <= 1.0

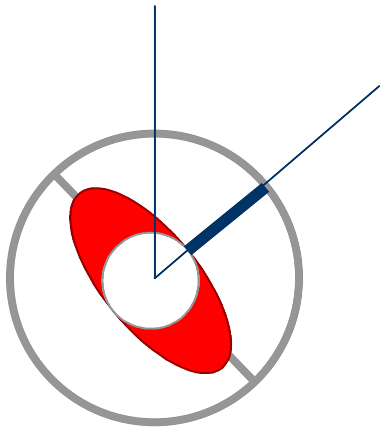


brake(p)

0.0 <= p <= 1.0

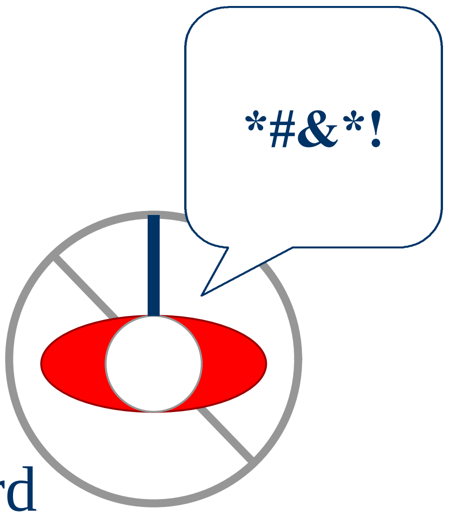


Methods 3 (3)



`head(a)`

`a` = angle in radians



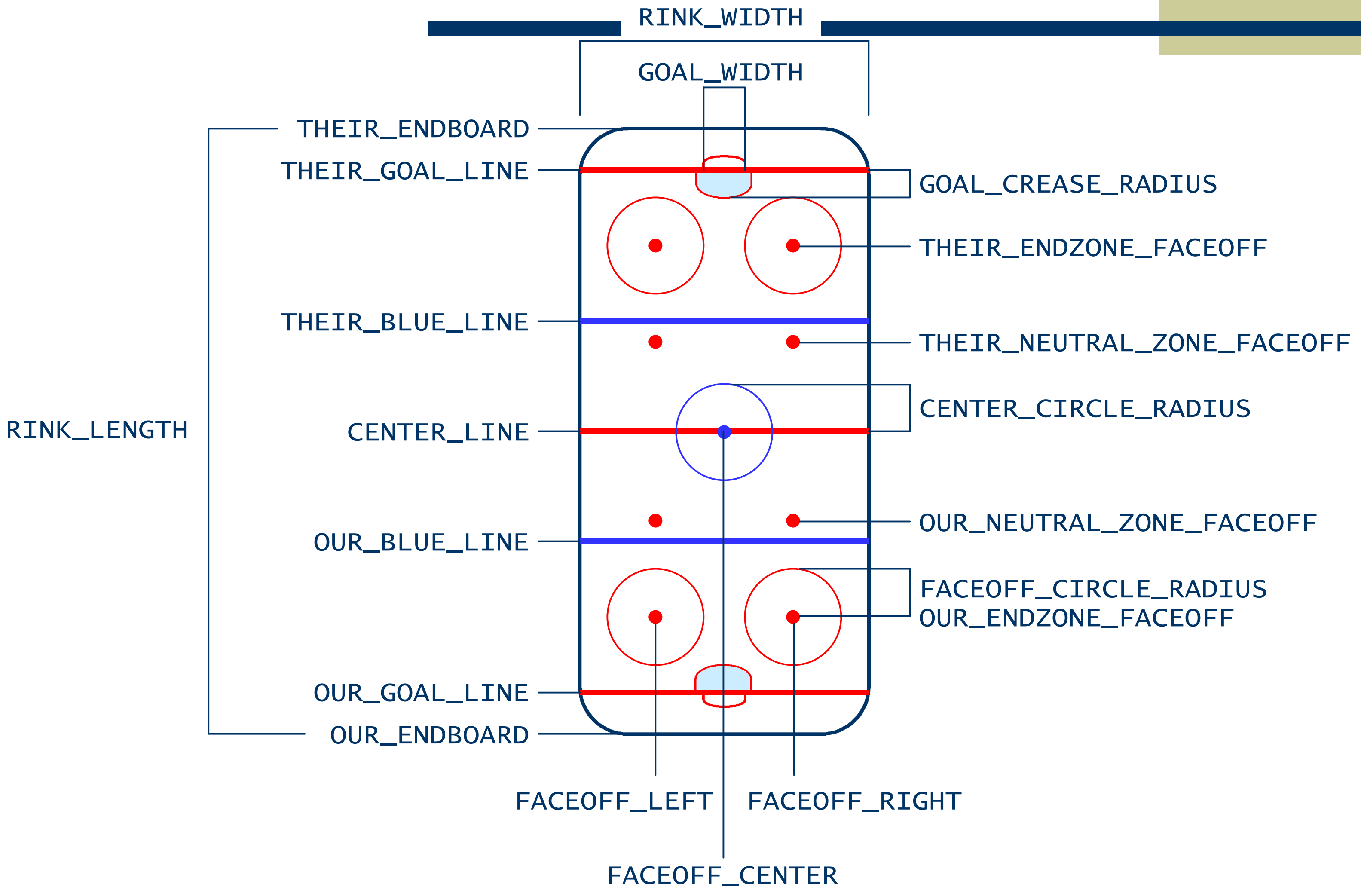
`say(m)`

`m` = 64-bit long word

Assorted inputs

- `double[] player( boolean us, int p)`
- `double[] puck()`
- `long[] messages( boolean us)`
- `int getPosition()`
- `int faceoff()`
- `boolean isGameStopped()`
- `boolean isIcing( boolean us)`
- `boolean isOffside( boolean us)`
- `int score( boolean us)`
- `long time()`

Constants


RINK_WIDTH
GOAL_WIDTH
THEIR_ENDBOARD
THEIR_GOAL_LINE
GOAL_CREASE_RADIUS
THEIR_ENDZONE_FACEOFF
THEIR_BLUE_LINE
THEIR_NEUTRAL_ZONE_FACEOFF
CENTER_CIRCLE_RADIUS
RINK_LENGTH
CENTER_LINE
OUR_NEUTRAL_ZONE_FACEOFF
OUR_BLUE_LINE
FACEOFF_CIRCLE_RADIUS
OUR_ENDZONE_FACEOFF
OUR_GOAL_LINE
OUR_ENDBOARD
FACEOFF_LEFT
FACEOFF_RIGHT
FACEOFF_CENTER

Example: MyAI.java

```java
import fi.utu.cs.hockey.ai.*;

public class MyAI extends AI implements Constants {
    public void react() {
        if (isPuckWithinReach())  {
            head(headingTo(0.0, THEIR_GOAL_LINE));
            brake(1.0);
            shoot(0.4);
            say(1L);
        } else {
            dash(1.0);
            head(headingTo(puck()));
}   }   }
```

Configuration files

```
SERVER_PORT 2345
INIT_PORT 3456
GROUP_ADDRESS 239.123.213.231
GROUP_PORT 4567
# only in a client
SERVER_ADDRESS 127.0.0.1
```

Team configuration in a client

```
TEAM DataCity Scientists

HELMET 0x0022FF
JERSEY 0xFFFFFF

LEFT_WING MyAI 22 L. Uuseri
CENTER    MyAI 10 Visa Koivu
```

Installation and use 1 (2)

- Installation, either (a) or (b):
  - (a) copy `AIsHockey.jar` into the JDK directory tree to the directory `/jre/lib/ext/`
  - (b) leave `AIsHockey.jar` in the work directory
- Starting the server, either (a) or (b)
  - (a) `java Server` *<configuration file>*
  - (b) `java -cp AIsHockey.jar;. Server` *<configuration file>*

Installation and use 2 (2)

- Starting a client, either (a) or (b)
  - (a) `java Client` *<configuration file>*
  - (b) `java –cp AIsHockey.jar;. Client` *<configuration file>*
- Compiling an AI file, either (a) or (b)
  - (a) `javac` *<file>*`.java`
  - (b) `javac -classpath AIsHockey.jar` *<file>*`.java`

How to run AIsHockey

(1) Start the server
(2) Start the clients
(3) Wait until the players have joined the game
(4) Start the game with a center spot faceoff
(5) When the game is over, shut down the server and the clients
(6) Goto 1

The small print