Technische Universität München

Faculty of Informatics

Robotics and Embedded Systems

http://www6.in.tum.de
Lab Course: Human Robot Interaction
Manuel Giuliani, Thomas Müller, Prof. Alois Knoll

April 28, 2010
Sheet 2
giuliani@in.tum.de, muelleth@in.tum.de

Submission Deadline: 05 May 2010

Exercise 2

In the second exercise you will control the robot for the first time. For that, we have defined Slice interfaces that can be used to tell the robot to execute a set of actions. These interfaces are shown in Figure 1. You can download the interfaces and all other files you need from here: `http://www6.in.tum.de/pub/Main/TeachingSs2010LabCourseHRI/hri-lab-course-sheet-2.zip`

In the first part of this exercise, please write a dummy robot server that implements the robot interfaces. You can follow the steps from the Ice documentation example for that. The dummy robot server does not have to really control the robot. It is enough if each method of the robot dummy server prints a message so that you know when the methods gets called by a client. The robot dummy server will be the first part of a development environment that we will build up step by step for the lab course.

When you have the robot dummy server, implement a client programme that establishes an Ice connection to your robot server and executes the following three action sets:

1. The robot shows its left hand and then its right hand
   Hint: use the interface `show(Hand.HandLeft)` and `show(Hand.HandRight)`

2. The robot picks up a cube, shows it to the user, and puts it back on the table.
   Hint 1: use `moveaway()` to move the robot to its default position before and after the action set.
   Hint 2: use the following table coordinates to pick up the cube and put it back on the table: `TableCoordinates(-0.248387, 0.0848238, 1.5708)`

3. Finally, you will programme the first “interaction” of human and robot. For this, let the robot show its left hand, open it, wait for five seconds, and then close it again to take a cube from a human. After that the robot should put the cube on the table at table coordinates `TableCoordinates(0.201529, 0.0364782, 0)`.

We will run your programme on the real robot in the third lesson of the course. Thus, **it is very important that you only use the coordinates we gave you**. Otherwise the robot or the table in front of the robot might get damaged.

When you implemented the three robot action sets, copy the code for calling the robot Ice interfaces into the file `RobotClient.java`, which is part of the source code files you downloaded above. In

`RobotClient.java`, the proxy to the robot is called `bodyPrx`, maybe you have to adjust your variable name.

Please send your robot dummy server, your own robot client, and the altered `RobotClient.java` that contains your code to `giuliani@in.tum.de` and `muelleth@in.tum.de` with subject `HRI Lab Course`.

```
 #ifndef _JAST_OUTPUT_BODY_ICE_
 #define _JAST_OUTPUT_BODY_ICE_

 #include <jast/common/Coordinates.ice>

module jast {
    module output {
        /** The robot hands **/
        enum Hand {
            HandLeft,
            HandRight,
            HandAny
        };

        /** Indicates that the operation failed for some other reason. **/
        exception OperationFailure {
            string details;
        };

        /** Indicates that the target was out of reach. **/
        exception CoordsOutOfReach extends OperationFailure {
           ::jast::common::TableCoordinates coords;
        };

        /** Indicates that the operation was interrupted. **/
        exception OperationAborted extends OperationFailure {
        };

        /** Methods to control the robot arms. **/
        interface Body {
            ["ami"] void close(Hand h) throws OperationFailure;
            double getGripperWidth(Hand h);
            ["ami"] void give(Hand h) throws OperationFailure;
            ["ami"] void insert(Hand h) throws OperationFailure;
            bool isReachable(Hand h, ::jast::common::TableCoordinates coords);
            ["ami"] void moveaway() throws OperationFailure;
            ["ami"] void open(Hand h) throws OperationFailure;
            ["ami"] void pickUp(Hand h, ::jast::common::TableCoordinates coords)
                 throws OperationFailure;
            ["ami"] void point(Hand h, ::jast::common::TableCoordinates coords)
               throws OperationFailure;
            ["ami"] void pointAtWorld(Hand h, ::jast::common::WorldCoordinates
               coords) throws OperationFailure;
            ["ami"] void putDown(Hand h, ::jast::common::TableCoordinates coords
               ) throws OperationFailure;
            ["ami"] void screw(Hand h) throws OperationFailure;
            ["ami"] void show(Hand h) throws OperationFailure;
            void stop();
            ["ami"] void take(Hand h) throws OperationFailure;
        };
    };
};

#endif // _JAST_OUTPUT_BODY_ICE_
```

Figure 1: Ice interface definitions to control the arms of the JAST robot