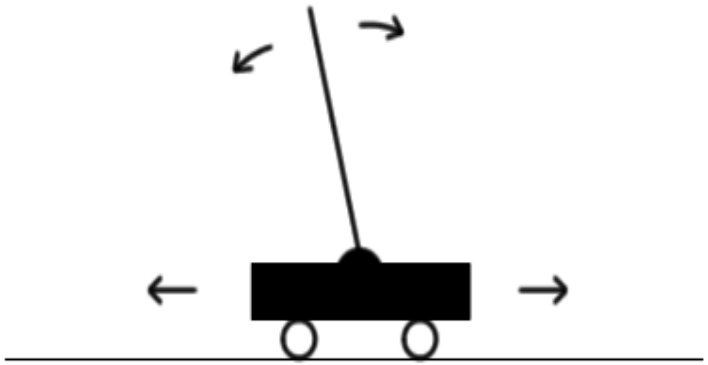
Inverted Pendulum

- Check Bitbucket: https://bitbucket.org/albertwj/inv_pendulum_simulator

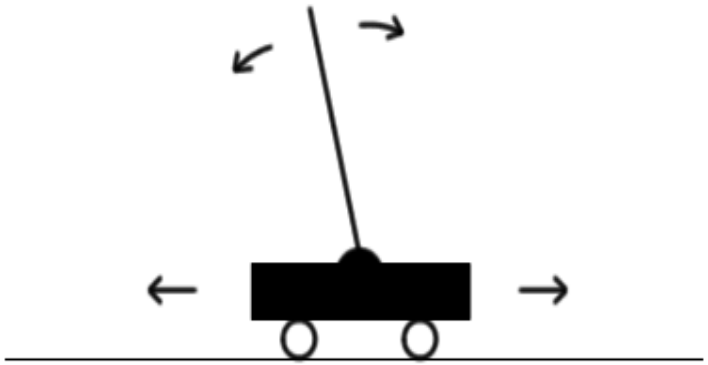
Inverted Pendulum

- A control system benchmark historically used in feedback control engineering.
- A pole affixed to a cart via a joint which allows movement along a single axis. The cart is able to move along a track of fixed length.



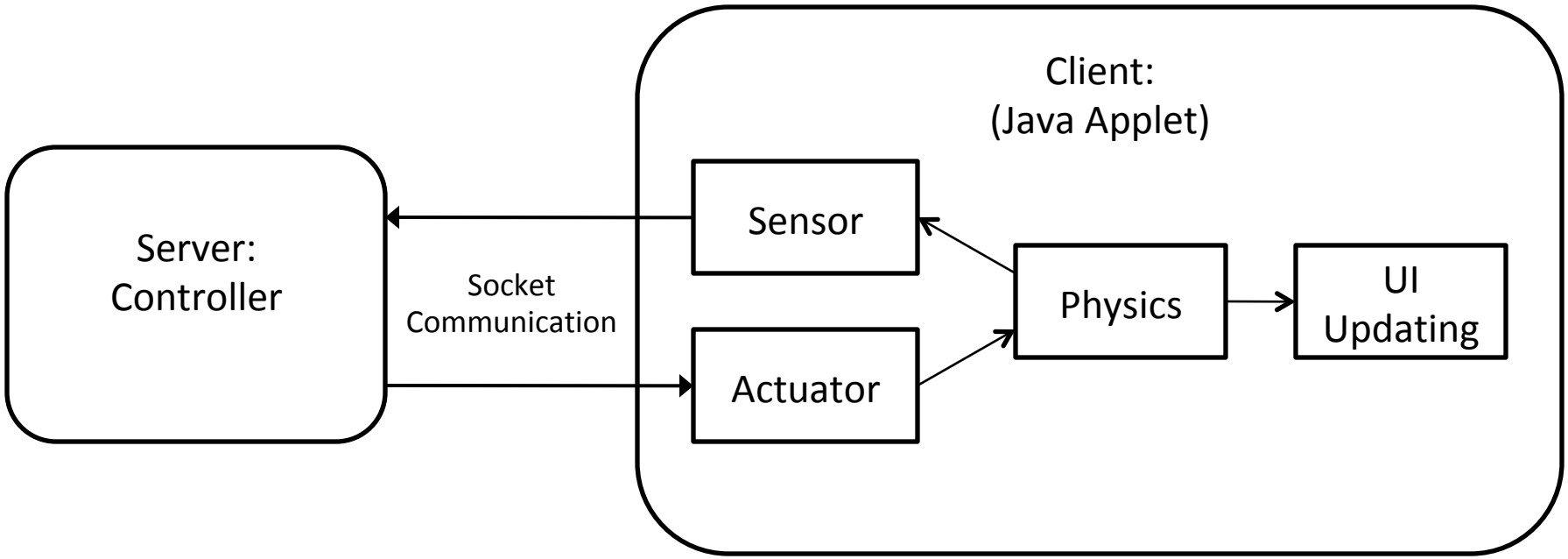
Single Pole Balancing

- A trial typically begins with the pole off-center by a certain number of degrees.
- The goal is to keep the pole from falling over by moving the cart in either direction, without the cart falling off either edge of the track.



Single Pole Balancing

Architecture



- ControlServer.java
  - Wait for client's connection
  - Receive the measurement data from the sensor
  - Calculate the force to be applied on the cart
    - Currently it only consider the pole angle to calculate the force

- Client.java
  - Main class for the client Applet
  - Takes simulation parameters from the user
  - Connect to the server
  - Start the four threads on client
- Physics.java
  - Contains the physics data of the simulation environment
  - Simulate the physical process
- Pendulum.java
  - Contains all the physics parameters of one inverted pendulum (mass, dimension, position …)

- UpdatingUIThread.java
  - According to the data in Physics, update the image in the user interface
- Sensor.java
  - Take measurements on Physics
  - Send data to controller
  - Currently measures pole angle, pole angle derivate, cart position, cart position derivate
- Actuator.java
  - Once receive the data from the controller, update the force applied in Physics

Parameters for simulation

- Simulation Speed
  - the ratio of simulation clock's speed over the real time clock speed
- Simulation step size (in simulation second)
  - the size of each step in this time-driven simulation
- Sensor Type
  - Time-Based sensor or Event-Based sensor
- Sensor sampling rate (per simulation second)
  - samples the sensor will measure per simulation second
- Threshold for the event based sensor

How to Run the Demo Code

- Install Java
- To Compile:
  - javac *.java
- To Run:
- First, run the controller (server)
  - make server
  - java ControlServer
- Second, Run the client (Applet):
  - make client
  - appletviewer Client.java

- To solve the “java.security.AccessControlException”:
  - Add a “.java.policy” file in your home directory
  - Add the following policy in the “.java.policy”:

```
grant {
    permission java.net.SocketPermission
    “127.0.0.1:25533”,
    “connect, resolve”;
};
```

Some Hints

Demo Video:
http://www.cs.utexas.edu/~mok/cs378/Videos/demo_hw2.mp4

About HW2: Task 1 and Task 2

- Mainly modify ControlServer.java
  - Need to modify the controller in the server side to calculate “force” so that the pole can be balanced
  - calculate_action():
    - Default implementation is table-based controller
    - You can use PID control or table-based controller or both

```
// Get sensor data of each pole and calculate the action to be
// applied to each inverted pendulum
// TODO: Current implementation assumes that each pole is
// controlled independently. This part needs to be changed if
// the control of one pendulum needs sensing data from other
// pendulums.
for (int i = 0; i < NUM_POLES; i++) {
  angle = data[i*4+0];
  angleDot = data[i*4+1];
  pos = data[i*4+2];
  posDot = data[i*4+3];

  System.out.println("server < pole["+i+"]: "+angle+"  "
      +angleDot+"  "+pos+"  "+posDot);
  actions[i] = calculate_action(angle, angleDot, pos, posDot);
```

About HW2: Test 3

- Show two carts: check Bitbucket

```
5. Configuration:
   To set the number of the pendulums and their initial position, change the
   the following lines accordingly:

        Server Side: (in ControlServer.java)
                private static final int NUM_POLES = 2;

        Client Side: (in Physics.java)
                public final int NUM_POLES = 2;
                public final double[] pole_init_pos = {-2.0, 2.0};
```

About HW2: Test 3

- Still, modify ControlServer.java
  - You could use the position of the other cart/pole as a hint

```
// Get sensor data of each pole and calculate the action to be
// applied to each inverted pendulum
// TODO: Current implementation assumes that each pole is
// controlled independently. This part needs to be changed if
// the control of one pendulum needs sensing data from other
// pendulums.
for (int i = 0; i < NUM_POLES; i++) {
  angle = data[i*4+0];
  angleDot = data[i*4+1];
  pos = data[i*4+2];
  posDot = data[i*4+3];

  System.out.println("server < pole["+i+"]: "+angle+"  "
      +angleDot+"  "+pos+"  "+posDot);
  actions[i] = calculate_action(angle, angleDot, pos, posDot);
```