Simulation Engineering Exercises 07

Name: Chenfeng Zhu
Matrikelnummer: 450485
University: TU-Clausthal
Program: ITIS
Language: JAVA
Lecturer: Dr. Umut Durak

1. Distributed Trapeze Simulation by UDP

Assumptions:
If the artist does not move, the rope would keep swinging itself.
If the artist changes his position, the pendulum would be affected.

Design:

a) TrapezeSimulation: it would start a thread to run the simulation and send the data (angle) outside. Before this, it would also start another thread to receive data (position) through UDP.

b) ArtistSimulation: it would start a thread to receive the data (angle) from the TrapezeSimulation. According to the angle, it would change the position and send it back to change the length of the Trapeze.

Main code:
TrapezeSimulation: run the simulation.

```
ByteArrayOutputStream out = null;
ObjectOutputStream os = null;
DatagramPacket sendPacket = null;
byte[] sendData;
for (int i = 0; t <= end_time; i++) {
    // delay
    try {
        Thread.sleep((int) (time_step_size * 1000));
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    // calculate the state
    double th_temp = th;
    th = th + th_v * time_step_size;
    th_v += (-g / l * Math.sin(th_temp)) * time_step_size;
    t += time_step_size;
    // print the state every 0.2 second.
    if (i % ((int) (0.2 / time_step_size)) == 0) {
        Platform.runLater(new Runnable() {
            public void run() {
                String str = printState(Math.round(t * 1000) / 1000.0, th, th_v, l);
                System.out.println(str);
                taLog.appendText(str + "\n");
            }
        });
    }
```

TrapezeSimulation : send the angle.

```
if (IPAddress == null) {
    continue;
}
// send the angle to the Artist.
data = new DataPacketAngle(th);
try {
    out = new ByteArrayOutputStream();
    os = new ObjectOutputStream(out);
    os.writeObject(data);
    sendData = out.toByteArray();
    sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, port);
    socket.send(sendPacket);
} catch (IOException e) {
    e.printStackTrace();
}
```

TrapezeSimulation: receive the position.

```
});
try {
    byte[] receiveData = new byte[1024];
    for (; socket != null;) {
        // receive the data by UDP.
        DatagramPacket receivePacket = null;
        receivePacket = new DatagramPacket(receiveData, receiveData.length);
        socket.receive(receivePacket);
        ByteArrayInputStream in = new ByteArrayInputStream(receiveData);
        ObjectInputStream is = new ObjectInputStream(in);
        Object obj = is.readObject();
        if (!(obj instanceof DataPacketPosition)) {
            // if the data is not position, ignore it.
            continue;
        }
        data = (DataPacketPosition) obj;
        // set the IP address and port.
        InetAddress address = receivePacket.getAddress();
        simulation.IPAddress = address;
        simulation.port = receivePacket.getPort();
        Platform.runLater(new Runnable() {
            public void run() {
                simulation.taLog.appendText("From " + address + ": "
                        + data.getValues() + ".\n");
            }
        });
        // change the length of the mass in the simulation.
        simulation.l = simulation.length + data.getValues();
    }
} catch (Exception e) {
    e.printStackTrace();
}
```

ArtistSimulation: receive the angle.

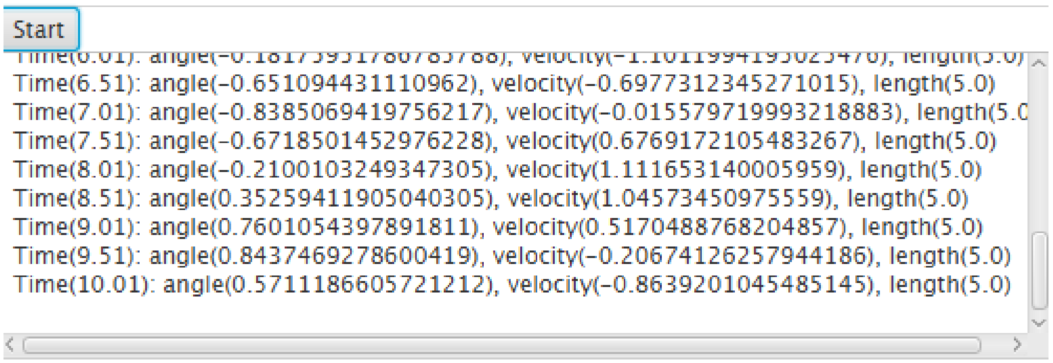
```
for (; socket != null;) {
    // receive angle from the Trapeze.
    receivePacket = new DatagramPacket(receiveData, receiveData.length);
    socket.receive(receivePacket);
    ByteArrayInputStream in = new ByteArrayInputStream(receiveData);
    ObjectInputStream is = new ObjectInputStream(in);
    Object obj = is.readObject();
    if (!(obj instanceof DataPacketAngle)) {
        continue;
    }
    angleData = (DataPacketAngle) obj;
    IPAddress = receivePacket.getAddress();
    double angle = angleData.getValues();
    Platform.runLater(new Runnable() {
        public void run() {
            taLog.appendText("From " + IPAddress + ": " + angle + ".\n");
        }
    });
```

ArtistSimulation: change the position and send it back.

```java
// changing the position.
if (System.currentTimeMillis() % 1000 == 0 && flag != 0) {
    // sometimes artist would stands.
    d = -h / 2;
    flag = 0;
} else {
    if (angle >= Math.toRadians(40) && flag != 1) {
        // at the highest point:
        d = 0;
        flag = 1;
    } else if (angle <= -Math.toRadians(40) && flag != 2) {
        // at the other point:
        d = 0;
        flag = 2;
    } else if (Math.abs(angle) <= Math.toRadians(5) && flag != 3) {
        // at the lowest point: artist would hang
        d = h / 2;
        flag = 3;
    } else {
        // if position isn't changed, continue.
        continue;
    }
}
positionData = new DataPacketPosition(d);
// send the position to the Trapeze.
port = receivePacket.getPort();
out = new ByteArrayOutputStream();
os = new ObjectOutputStream(out);
os.writeObject(positionData);
sendData = out.toByteArray();
sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, port);
socket.send(sendPacket);
Platform.runLater(new Runnable() {
    public void run() {
        taLog.appendText("Sent to " + address + ": " + d + ".\n");
    }
});
```

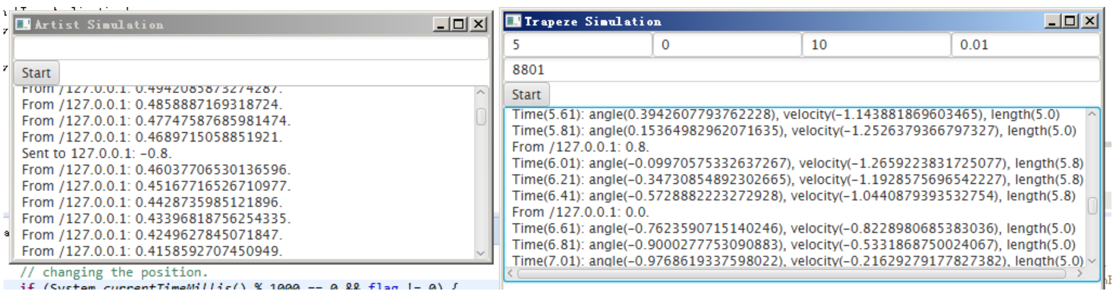
Images:

Run the simulation itself:



```
Time(6.01): angle(-0.18173931780783788), velocity(-1.1011994193025476), length(5.0)
Time(6.51): angle(-0.651094431110962), velocity(-0.6977312345271015), length(5.0)
Time(7.01): angle(-0.8385069419756217), velocity(-0.015579719993218883), length(5.0
Time(7.51): angle(-0.6718501452976228), velocity(0.6769172105483267), length(5.0)
Time(8.01): angle(-0.2100103249347305), velocity(1.111653140005959), length(5.0)
Time(8.51): angle(0.35259411905040305), velocity(1.04573450975559), length(5.0)
Time(9.01): angle(0.7601054397891811), velocity(0.5170488768204857), length(5.0)
Time(9.51): angle(0.8437469278600419), velocity(-0.20674126257944186), length(5.0)
Time(10.01): angle(0.5711186605721212), velocity(-0.8639201045485145), length(5.0)
```

Run the distributed simulation.



```
From /127.0.0.1: 0.4942085873274287.
From /127.0.0.1: 0.4858887169318724.
From /127.0.0.1: 0.47747587685981474.
From /127.0.0.1: 0.4689715058851921.
Sent to 127.0.0.1: -0.8.
From /127.0.0.1: 0.46037706530136596.
From /127.0.0.1: 0.45167716526710977.
From /127.0.0.1: 0.4428735985121896.
From /127.0.0.1: 0.43396818756254335.
From /127.0.0.1: 0.4249627845071847.
From /127.0.0.1: 0.4158592707450949.
```

```
5    0    10    0.01
8801
Time(5.61): angle(0.3942607793762228), velocity(-1.143881869603465), length(5.0)
Time(5.81): angle(0.15364982962071635), velocity(-1.2526379366797327), length(5.0)
From /127.0.0.1: 0.8.
Time(6.01): angle(-0.09970575332637267), velocity(-1.2659223831725077), length(5.8)
Time(6.21): angle(-0.34730854892302665), velocity(-1.1928575696542227), length(5.8)
Time(6.41): angle(-0.5728882223272928), velocity(-1.0440879393532754), length(5.8)
From /127.0.0.1: 0.0.
Time(6.61): angle(-0.7623590715140246), velocity(-0.8228980685383036), length(5.0)
Time(6.81): angle(-0.9000277753090883), velocity(-0.5331868750024067), length(5.0)
Time(7.01): angle(-0.9768619337598022), velocity(-0.21629279177827382), length(5.0)
```

```java
// changing the position.
```

Output results:

The results:

a) Run the Trapeze Simulation without an artist:

```
Time(0.01): angle(0.7853981633974483), velocity(-0.013859292911256331), length(5.0)
Time(0.51): angle(0.6135974032834047), velocity(-0.6655415469723961), length(5.0)
Time(1.01): angle(0.16877266416588152), velocity(-1.0549428192264978), length(5.0)
Time(1.51): angle(-0.35812157280706897), velocity(-0.9650046131105184), length(5.0)
Time(2.01): angle(-0.7266968639282565), velocity(-0.4471514005896795), length(5.0)
Time(2.51): angle(-0.7826813777824618), velocity(0.2440380793286929), length(5.0)
Time(3.01): angle(-0.5028742500670679), velocity(0.8515159095718431), length(5.0)
Time(3.51): angle(0.005080053292219812), velocity(1.1053385658841517), length(5.0)
Time(4.01): angle(0.5155811465222296), velocity(0.8507575791156752), length(5.0)
Time(4.51): angle(0.7996520361197942), velocity(0.23633715235418878), length(5.0)
Time(5.01): angle(0.7438071405317093), velocity(-0.46775704129667284), length(5.0)
Time(5.51): angle(0.3659556560477669), velocity(-1.002690839295754), length(5.0)
Time(6.01): angle(-0.18175951786785788), velocity(-1.1011994195025476), length(5.0)
Time(6.51): angle(-0.651094431110962), velocity(-0.6977312345271015), length(5.0)
Time(7.01): angle(-0.8385069419756217), velocity(-0.015579719993218883), length(5.0)
Time(7.51): angle(-0.6718501452976228), velocity(0.6769172105483267), length(5.0)
Time(8.01): angle(-0.2100103249347305), velocity(1.111653140005959), length(5.0)
Time(8.51): angle(0.35259411905040305), velocity(1.04573450975559), length(5.0)
Time(9.01): angle(0.7601054397891811), velocity(0.5170488768204857), length(5.0)
Time(9.51): angle(0.8437469278600419), velocity(-0.20674126257944186), length(5.0)
Time(10.01): angle(0.5711186605721212), velocity(-0.8639201045485145), length(5.0)
```

b) Run the Trapeze Simulation with an artist:

```
From /127.0.0.1: 0.0.
Time(0.01): angle(0.7853981633974483), velocity(-0.013859292911256331), length(5.0)
From /127.0.0.1: 0.0.
Time(0.21): angle(0.7564090623426171), velocity(-0.28847495916514126), length(5.0)
Time(0.41): angle(0.6737256150577959), velocity(-0.5473097084354767), length(5.0)
Time(0.61): angle(0.5420977423870589), velocity(-0.7733113800521476), length(5.0)
From /127.0.0.1: -0.8.
Time(0.81): angle(0.368937905798503), velocity(-0.9651064341550287), length(4.2)
Time(1.01): angle(0.1624461202346037), velocity(-1.0907017934837056), length(4.2)
……
Time(8.01): angle(-0.42545635452164565), velocity(1.2048898877137906), length(5.0)
Time(8.21): angle(-0.17168452645641868), velocity(1.3230081227605557), length(5.0)
From /127.0.0.1: 0.8.
Time(8.41): angle(0.09625884949445371), velocity(1.340562110946195), length(5.8)
Time(8.61): angle(0.3587581321064584), velocity(1.2663213601362584), length(5.8)
Time(8.81): angle(0.5985715353701804), velocity(1.112081586976281), length(5.8)
From /127.0.0.1: 0.0.
Time(9.01): angle(0.8004457529596385), velocity(0.8767801824295406), length(5.0)
Time(9.21): angle(0.9478573771383446), velocity(0.5760666588518212), length(5.0)
Time(9.41): angle(1.032185129500184), velocity(0.24772364571039587), length(5.0)
Time(9.61): angle(1.0495630148985462), velocity(-0.09153323143668715), length(5.0)
Time(9.81): angle(0.9991466881207066), velocity(-0.4278049072542087), length(5.0)
Time(10.01): angle(0.8828907104104623), velocity(-0.7463145831611551), length(5.0)
```

Conclusion:
NULL.

Code:

https://github.com/sampig/SimulationEngineering