Requirements

Extend the ***robotchrono*** system by adding a new command: - loop: the robot must be able to run and measure the arena perimeter

Requirement analysis

I assume the robot lays on a plane within a rectangular box. There are no other obstacles.
I also assume the loop command is sent when the robot is in one of the 4 corners of the box.
The perimeter measurement problematic is left fot the Problem analysis stage.

```
System looper

...

Dispatch loop   : loop(X)  // req-loop

...

QActor robotmind context ctxMind {
    ["var loopCounter = 0;"]

    State idle {
        println("idle")
    }
    Transition tWork
        ...
        whenMsg loop -> sLoop

    ...

    State sLoop {
        println("sLoop")
        forward basicrobot -m cmd : cmd(w)
    }
    Transition tLoop
        whenEvent obstacle and "loopCounter < 3" -> sObstacleLoop
        whenEvent obstacle and "loopCounter == 3" -> sEndLoop

    State sObstacleLoop {
        println("sObstacleLoop")
        ["loopCounter += 1;"]
        forward basicrobot -m cmd : cmd(a)
    }
    Goto sLoop

    State sEndLoop {
        println("sEndLoop")
        ["loopCounter = 0;"]
        forward basicrobot -m cmd : cmd(a)
    }
    Goto idle
}
```

Problem analysis

The problematic of deriving a distance from the elapsed movement time requires a good measure of the robot speed. I assume the accelleration to be instantanious (i.e. non continuous) and the speed constant and known. This is pretty straightforward in a virtual environment while it is to be considered impossible in the real environment. In the real world assumptions and approximations are needed to complete the task.

```
System looper

...

Dispatch loop   : loop(X)  // req-loop

...
```

```
QActor robotmind context ctxMind {
    ["var StepTime = 0L;"]
    ["var start = 0L;"]
    ["var elapsed = 0L;"]
    ["var loopCounter = 0;"]

    State idle {
        println("idle")
    }
    Transition tWork
        ...
        whenMsg loop -> sLoop

    ...

    State sLoop {
        println("sLoop")
        ["start = System.currentTimeMillis();"] // start the chronometer
        forward basicrobot -m cmd : cmd(w)
    }
    Transition tLoop
        whenEvent obstacle and "loopCounter < 3" -> sObstacleLoop
        whenEvent obstacle and "loopCounter == 3" -> sEndLoop

    State sObstacleLoop {
        println("sObstacleLoop")
        ["val runtime = System.currentTimeMillis() - start;"] // stop the chronometer
        ["elapsed += runtime;"]
        ["loopCounter += 1;"]
        forward basicrobot -m cmd : cmd(a)
    }
    Goto sLoop

    State sEndLoop {
        println("sEndLoop")
        ["loopCounter = 0;"]
        ["elapsed = 0L;"]
        ["val perimeter = elapsed * 0.2;"] // stop the chronometer
        println("elapsed time: ${elapsed}")
        println("perimeter: ${perimeter} meters")
        forward basicrobot -m cmd : cmd(a)
    }
    Goto idle
}
```

Deployment

Build both BasicRobot and RobotMind into deployable zip files with the commands: gradle -b build_ctxMind.gradle distZip gradle -b build_ctxBasicRobot.gradle distZip Copy any *.pl file into the bin sub directory and execute the executable scripts.

By Michele Proverbio email: michele.proverbio@studio.unibo.it