Virtual robot 2021

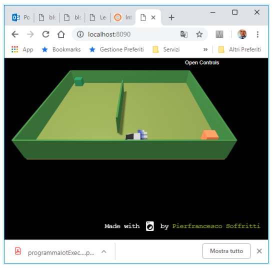
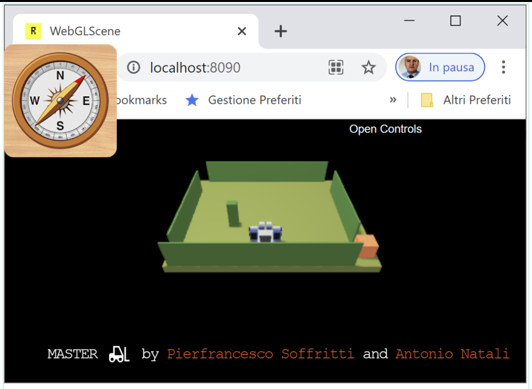
ISS-lectures site

INTRODUCTION

Our company (Unibo) has developed a virtual environment (named *WEnv*) that includes a virtual robot.
The virtual environment has been mainly built by *Pierfrancesco Soffritti* using the three.js JavaScript library (see ConfigurableThreejsApp).

THE SCENE

The scene of the WEnv is built from a description can be easily defined by an application designer. An example (related to the scene on the right of the figure below) can be found in sceneConfig.js.

THE VIRTUAL ROBOT

The virtual robot is equipped with two impact sensors, one put in front and one in the rear of the robot.
The robot can perform simple moves, each lasting a finite time (specified by the user) as explained in the nexet section.

INTERACTION

Robot as receiver of commands

The virtual robot uses the Node library https://github.com/einaros/ws to accept:

- commands sent as HTTP POST messages on port 8090
- commands sent on a websocket at port 8091

The cril (concrete-robot interaction language)

The commands are expressed as JSON strings:

```
{"robotmove":"MOVE", "time":T}        with MOVE=turnLeft | turnRight | moveForward | moveBackward | alarm, T=naturalNum
```

The set of strings written in this form constitute the **concrete-robot interaction language** (**cril**)

Each command is redirected (using socket.io) to the WebGLScene.

Each command sends a response

After the execution of a command, the robot sends to the caller (both using POST and the websocket) an answer expressed in JSON:

```
{"endmove":"RESULT", "move":MOVE}   with RESULT = true | false
```

A RESULT **false** means that the move has aborted (e.g. because the robot has hit an obstacle).

Robot as sender of information

Moreover, the WEnv ***sends to the clients*** connected to the ws port 8091:

- Data emitted by the sonars included in the scene when they detect some object (the moving robot)
- Data emitted by the impact sensors put in front and in the rear of the robot, when they detect an obstacle (fixed or mobile). For example:

```
{ "sonarName": "sonarName", "distance": 1, "axis": "x" }

{ "collision" : "false", "move": "moveForward"}
```

DEPLOYOMENT (on docker)

```
docker-compose -f  virtualrobotandcontrol.yaml up
```

With **virtualrobotandcontrol.yaml** defined as follows (take care of indent):

```
version: '3'
services:
  wenv:
    image: docker.io/natbodocker/virtualrobotdisi:1.0
    ports:
      - 8090:8090
      - 8091:8091
  vrobotcontrol:
    image: docker.io/natbodocker/virtualrobotcontrol:1.0
    ports:
      - 3000:3000
    depends_on:
      - wenv
```

By AN Unibo-DISI