



Universidad
Rey Juan Carlos



Escuela Técnica Superior de
Ingeniería de Telecomunicación

Mejoras en entorno de robótica educativa para niños

Trabajo de fin de grado

Rubén Álvarez Martín

José María Cañas Plaza

Índice

1. Introducción

2. Objetivos

3. Herramientas

4. Mejoras a WebSim

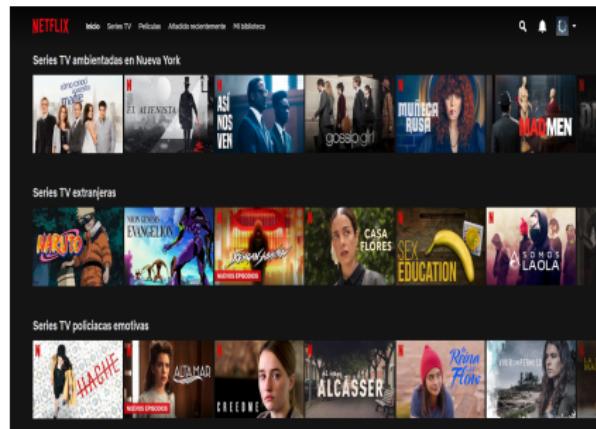
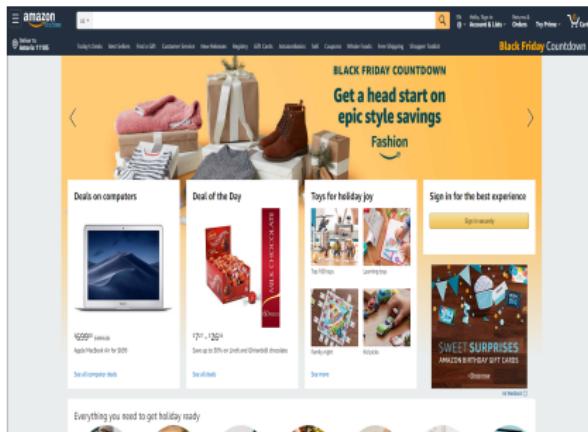
- ▶ Soporte a drones en WebSim
- ▶ Teleoperadores en WebSim
- ▶ Ejercicios individuales
- ▶ Ejercicios competitivos

5. Conclusiones

Introducción

Tecnologías web

- HTTP
- Tecnologías cliente: HTML5, CSS3 y JS.
- Tecnologías servidor: Node, Django y Spring.



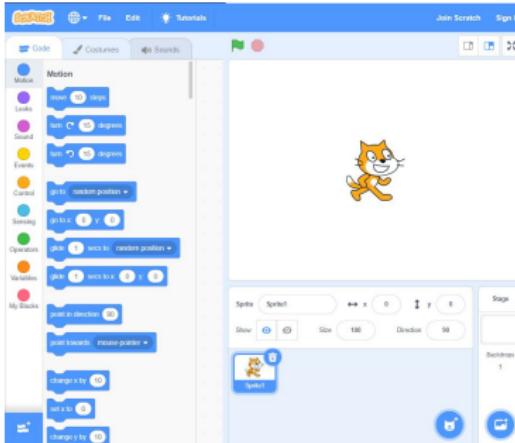
Robótica



- ▶ *robot = hardware + software*
- ▶ *hardware = sensores + actuadores + procesadores*

Robótica educativa

- Plataformas hardware: *LEGO mindstorm*, *mBot* o *Arduino*.



- Lenguajes de programación visual:
Scratch, *Snap!* o *Kodu*.

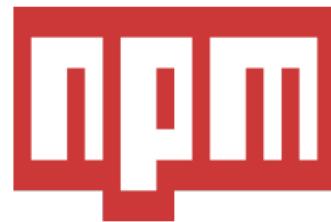
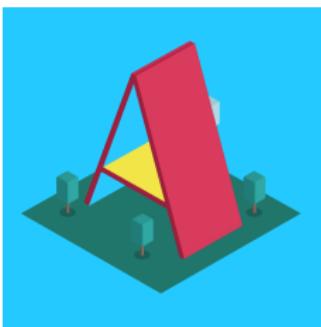
Objetivos

Objetivos

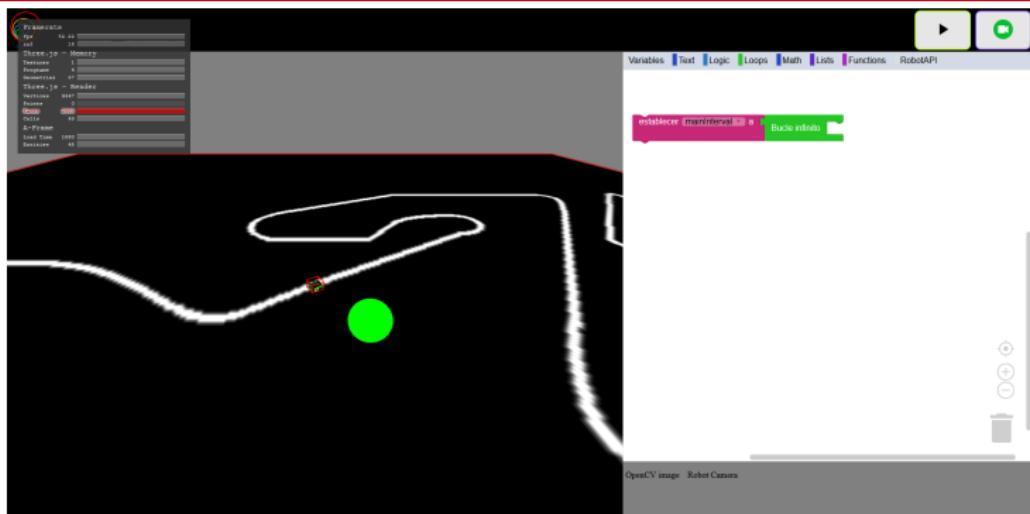
- ▶ Mejorar *WebSim*:
 1. Soporte a *drones*.
 2. Teleoperadores y ficheros de configuración.
 3. Ejercicios individuales.
 4. Ejercicios competitivos y evaluadores automáticos.

Herramientas

Herramientas



WebSim



- *WebSim* -> simulador web robótico para enseñar conceptos básicos de tecnología, robótica y programación (Álvaro Paniagua).

Mejoras a WebSim

Soporte a drones: Modelo 3D



- Formato *A-Frame*
- Modelo *low-poly*
- Animación hélices

Soporte a drones: Drivers

- **HAL API:** Sensores y actuadores.

Método	Descripción
.setL(integer)	Comanda velocidad ascendente
.getL()	Devuelve velocidad ascendente
.despegar()	Despega <i>drone</i>
.aterrizar()	Aterriza <i>drone</i>
.move(integer, integer, integer)	Comanda velocidades
setVelocity()	Materializa velocidad y posición

Soporte a drones: Bloques Scratch

Eleva myRobot a velocidad

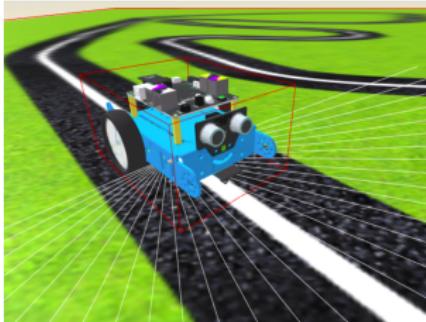
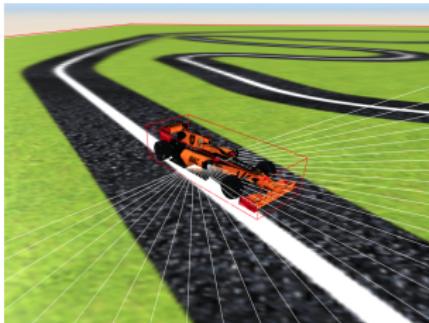
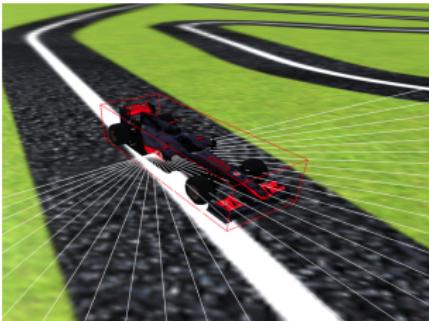
Obtener la velocidad de elevación de myRobot

Aterrizar myRobot

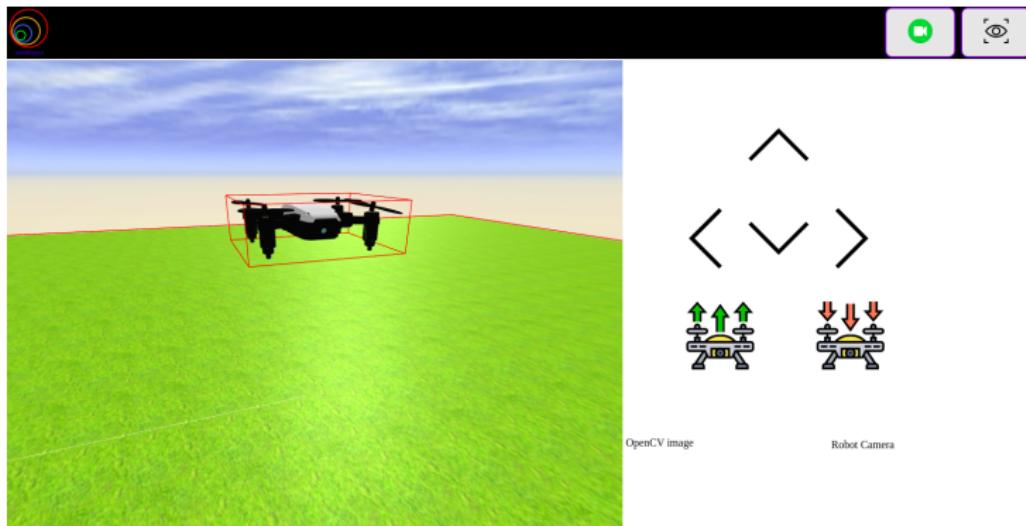
Despegar myRobot

```
export default function initTakeoffBlock(){
  var takeoffBlock = {
    "type": "takeoff",
    "message0": "%{BKY_TAKEOFF_TEXT}",
    "args0": [
      {
        "type": "field_variable",
        "name": "ROBOT_VAR",
        "variable": "myRobot"
      }
    ],
    "previousStatement": null,
    "nextStatement": null,
    "colour": "%{BKY_ROBOT_MOTORS_HUE}",
    "tooltip": "%{BKY_TAKEOFF_TOOLTIP}",
    "helpurl": ""
  }
  Blockly.Blocks['takeoff'] = {
    init: function() {
      this.jsonInit(takeoffBlock);
    }
  };
  Blockly.JavaScript['takeoff'] = function(block) {
    var robotvar = Blockly.JavaScript.variableDB_.getName(block.getFieldValue('ROBOT_VAR'), Blockly.Variables.NAME_TYPE);
    var code = robotvar + '.setL(3); \nawait sleep(0.5); \n' + robotvar + '.setL(0); \n';
    return code;
  };
  Blockly.Python['takeoff'] = function(block) {
    var robotvar = Blockly.Python.variableDB_.getName(block.getFieldValue('ROBOT_VAR'), Blockly.Variables.NAME_TYPE);
    var code = robotvar + '|despegar|; \n\n' + 'time.sleep(0.5)\n';
    return code;
  };
}
```

Otros modelos



Teleoperadores

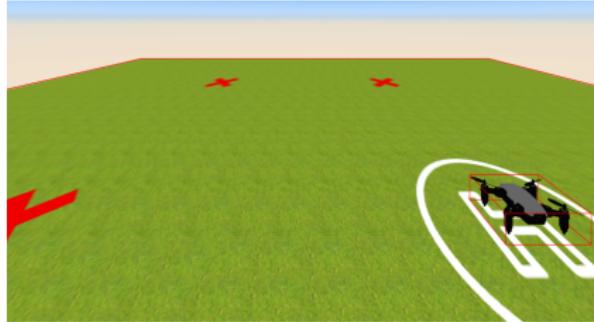
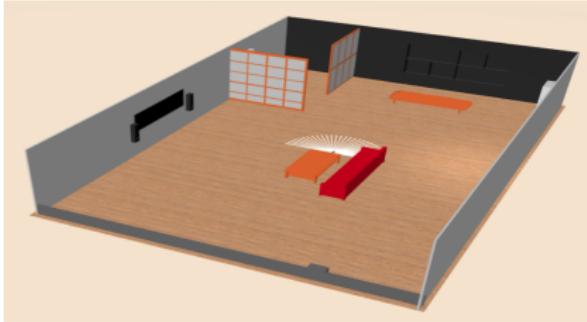
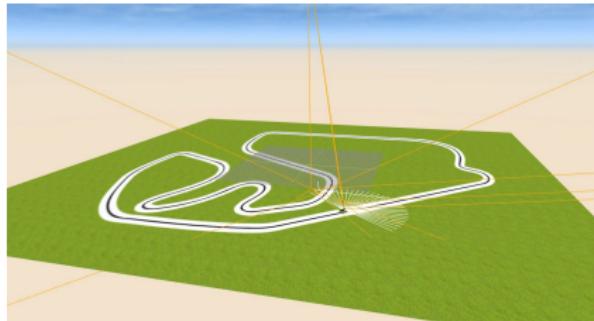
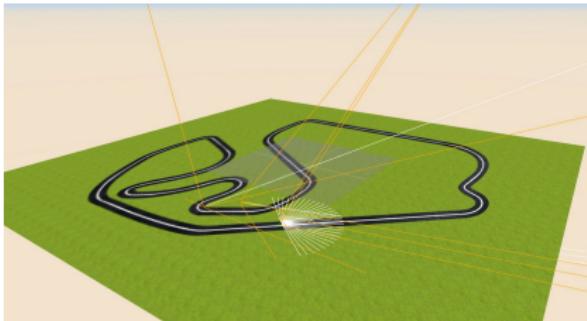


- ❖ Permiten controlar robots sin programarlos.

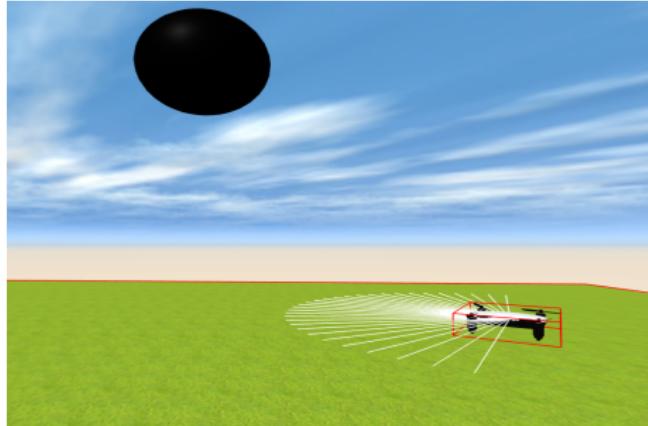
Teleoperadores: configuración

- Ficheros de configuración -> crear escenario sin tener elementos en el código fuente.
- Formato JSON -> escenario, robot, gravedad, elementos, etc.

Ejercicios individuales



Ejercicios individuales: Sigue pelota



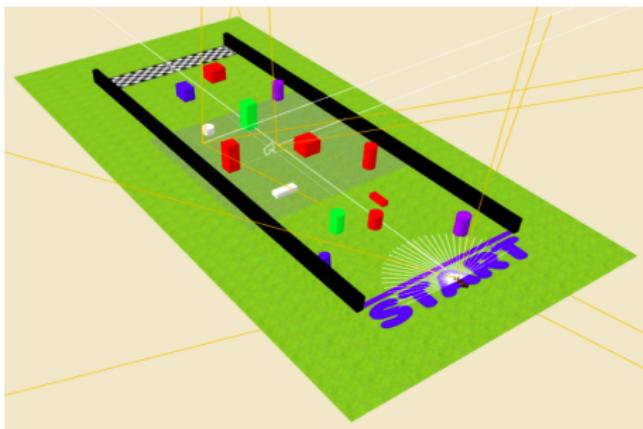
```
Bucle infinito
  establecer [centroX v] a [De la cámara de myRobot obtener centroX del objeto de color black]
  establecer [centroY v] a [De la cámara de myRobot obtener centroY del objeto de color black]
  establecer [area v] a [De la cámara de myRobot obtener area del objeto de color black]
  establecer [errorX v] a [(65 - centroX) / 0.0005]
  establecer [errorY v] a [(centroY - 30) / 0.0005]

  si [centroX < 57]
    hacer [Girar (myRobot) a la izquierda a velocidad |error|]
  sino si [centroX > 63]
    hacer [Girar (myRobot) a la derecha a velocidad |error| * -0.5]
  sino
    hacer [Girar (myRobot) a la izquierda a velocidad 0.2]

  si [centroY > 60]
    hacer [Eleva (myRobot) a velocidad -0.2]
  sino si [centroY < 48]
    hacer [Eleva (myRobot) a velocidad 0.1]
  sino si [centroY > 55]
    hacer [Eleva (myRobot) a velocidad 0.2]
  sino
    Parar (myRobot)

  si [area < 150]
    hacer [Avanza (myRobot) a velocidad 0.5]
  sino si [area > 160]
    hacer [Retrocede (myRobot) a velocidad 0.5]
```

Ejercicios individuales: Atraviesa bosque

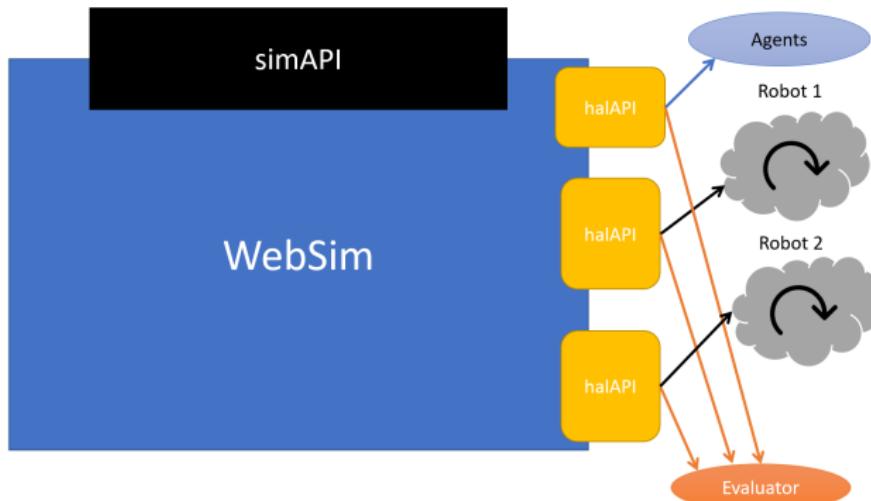


```
Bucle infinito establecer distancias a Para myRobot obtener las distancias en arco 180 grados
Avanza myRobot a velocidad 0.25
Gira myRobot a la izquierda a velocidad 0
establecer menor a 100
establecer id a 15
contar con id desde 10 hasta 20 de a 1
hacer si en la lista distancias obtener < menor
hacer establecer menor a en la lista distancias obtener
establecer id a 15
hacer si menor < 5
hacer id = 15
hacer Avanza myRobot a velocidad 0.75
Gira myRobot a la derecha a velocidad 0.5
sino si id = 15
hacer Avanza myRobot a velocidad 0.5
Gira myRobot a la izquierda a velocidad 0.05
sino si id = 15
hacer Avanza myRobot a velocidad 0.5
Gira myRobot a la derecha a velocidad 0.05
```

<https://youtu.be/z3n47wWHDFc>

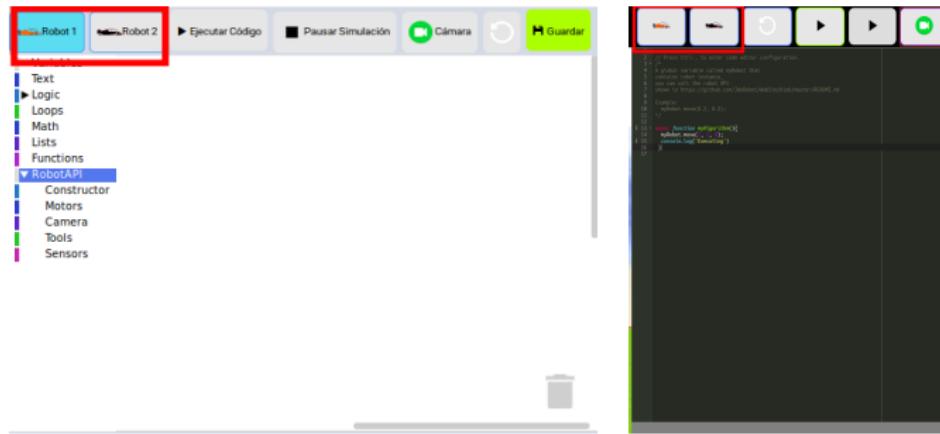
Ejercicios competitivos: Arquitectura

- Módulo **brains** ampliado.
- Módulo **evaluators**: `runEvaluator(arrayRobots,evaluator)`
- Módulo **agents**: `runAgent(idRobot, path)`

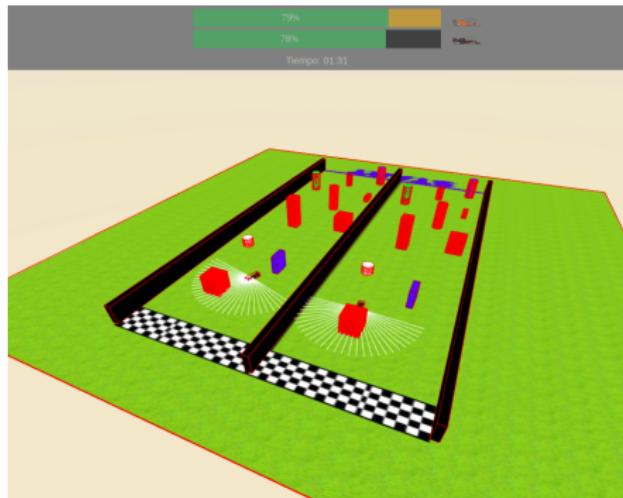


Ejercicios competitivos: Editores

► Editores dobles:



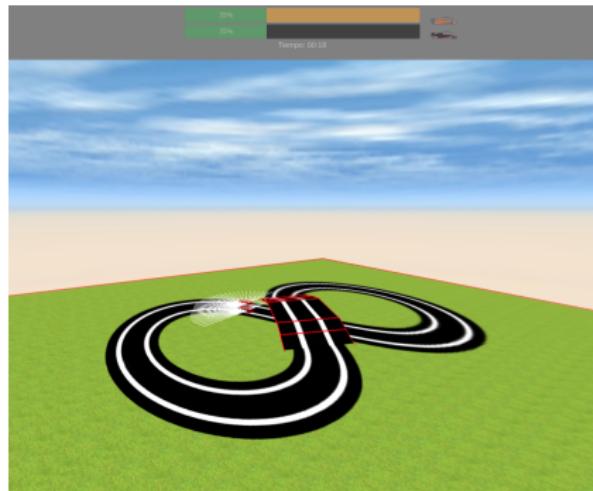
Ejercicios competitivos: Atraviesa bosque



- % distancia recorrida
- Mismo recorrido

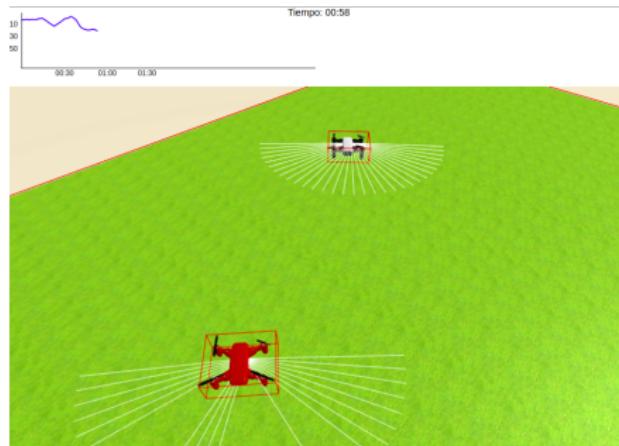
Ejercicios competitivos: Sigue líneas visión

- ❖ % distancia recorrida
- ❖ Puente primitivas
A-Frame



https://youtu.be/0aA7_wsXhk8

Ejercicios competitivos: Gato-ratón



- ▶ Distancia *drones* en gráfica
- ▶ Módulo *agents*

<https://youtu.be/Ez9MStthNWqA>

Ejercicios competitivos: Gato-ratón evaluador

- ❖ Distancia
- ❖ Gráfica
- ❖ Cronómetro

```
evaluator.setEvaluator = (arrayRobots) => {
    var robot1 = Websim.robots.getHalAPI(arrayRobots[0]);
    var robot2 = Websim.robots.getHalAPI(arrayRobots[1]);
    if(robot1.velocity.x!=0 || robot2.velocity.x!= 0 || robot1.velocity.y != 0 || robot2.velocity.y != 0){
        clock = true;
    }
    if(!clock){
        timeInit = new Date();
    }else{
        var time= document.getElementById("time");
        var realTime = new Date(new Date() - timeInit);
        var formatTime = timeFormatter(realTime);
        time.innerHTML = "Tiempo: " + formatTime;
        var pos1 = robot1.getPosition();
        var pos2 = robot2.getPosition();
        var dist = Math.sqrt(Math.pow(pos2.x-pos1.x,2)+Math.pow(pos2.y-pos1.y,2)+Math.pow(pos2.z-pos1.z,2));
        line.addPointXY(x,dist+10);
        x+=1;
        myPanel.addElement(line);
    }
}
```

Conclusiones

Conclusiones

- Soporte a *drones*. ✓
- Teleoperadores y ficheros de configuración. ✓
- Ejercicios individuales. ✓
- Ejercicios competitivos: agentes y evaluadores. ✓

Líneas futuras

- *WebWorkers.*
- Control en posición interrumpible.