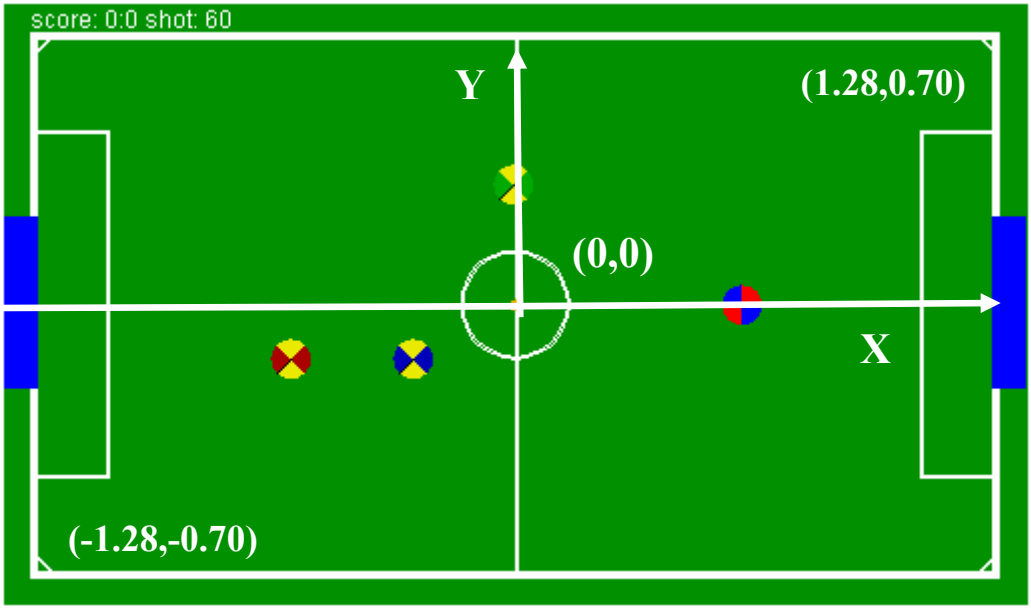
# Algunas aclaraciones con respecto a la práctica 1

*Luis de la Ossa.*

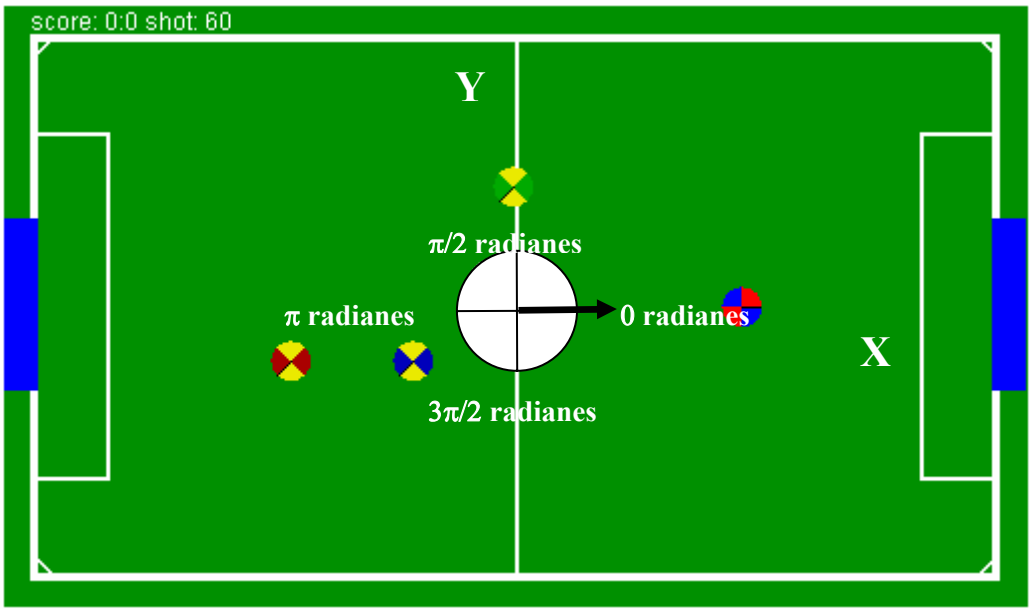
- **Posición del robot.**

Es relativa a los ejes X e Y que parten del centro del campo.




- **Dirección del robot.**

La lectura de la dirección actual del robot devuelve esta en radianes **CON RESPECTO AL EJE X DEL CAMPO**. Antes de introducirla en el objeto rete, es transformada a grados.



La inferencia devuelve también una dirección que es la que debe adoptar el robot. En lugar de en grados, lo hace (por facilidad) mediante un vector de posición **RELATIVO AL CENTRO DEL ROBOT**. Por ejemplo, los siguientes vectores darían lugar a estos ángulos con respecto al eje X.

- (0,0) →
- (1,0) →
- (3,3) ↗
- (0,2) ↑
- (-1,1) ↖
- (-4,0) ←
- (-2,-2) ↙
- (0,-1) ↓
- (4,-4) ↘

Así, si quiero ir a un punto definido en las coordenadas del plano, solamente tengo que restarlo con el centro del robot y me da el vector que uso para orientarme.

Como considero directamente el vector de movimiento del robot, para determinar la nueva dirección en radianes hay que cambiar estas líneas:

```
coord.x = -x.floatValue(null);
coord.y = -y.floatValue(null);
double t = Math.atan2(coord.x,coord.y);
```

Por estas:

```
coord.setx(x.floatValue(null));
coord.sety(y.floatValue(null));
double t = coord.t;
```

Donde t es directamente el argumento del vector en coordenadas polares.

- **Posición de la pelota, porterías, compañeros y contrarios**

Para obtener datos sobre el resto de los objetos lee también vectores **RELATIVOS A SU POSICIÓN,** es decir, que si el robot está en la posición (0.5, 0.5) y la posición de la pelota es (0.7, 0.7), leerá un vector (0.2, 0.2), y con él calculará el ángulo como se ha descrito anteriormente.

Por tanto, en caso de tener que orientarnos donde esté la pelota o las porterías, no es necesario hacer ninguna operación.

El siguiente ejemplo, es un archivo .clp que haría que los jugadores fueran detrás de la pelota.

```
(deffacts obtenir_datos
      (x_pelota (fetch "x_pelota"))
      (y_pelota (fetch "y_pelota"))
    )

(defrule inicial

    (x_pelota ?xm)
    (y_pelota ?ym)

   =>
      (store "coordx" ?xm)
      (store "coordy" ?ym)
      (store "velocidad" 0.5)
      (store "chutar" 0.3)
 )
```

Existe un problema (bug) que hace que, al principio de la ejecución, algunos vectores sean nulos, y se produce un error. Por ejemplo, si al fichero anterior le añadimos una linea con un test para que solo persiga la pelota si el módulo del vector es menor que 0.5:

```
(deffacts obtenir_datos
      (x_pelota (fetch "x_pelota"))
      (y_pelota (fetch "y_pelota"))
    )

(defrule inicial
    (x_pelota ?xm)
    (y_pelota ?ym)
    (test (<= (distancia ?xm ?ym) 0.5))

   =>
      (store "coordx" ?xm)
      (store "coordy" ?ym)
      (store "velocidad" 0.5)
      (store "chutar" 0.3)
 )
```

La regla funcionaría, pero nos daría  dos o tres salidas como esta:

```
TeamBots 2.0e (c)2000 Tucker Balch, CMU and GTRC
1298
maxtimestep statement read, treated as timestep
Jess reported an error in routine Value.javaObjectValue
      while executing (<= ?xm 10)
      while executing 'test' CE
      while executing rule LHS (Node2)
      while executing rule LHS (MTELN)
      while executing rule LHS (TECT)
      while executing assert from deffacts MAIN::obtenir_dades
      while executing (reset).
  Message: '10' is an integer, not  a Java object.
  Program text: ( reset )  at line 1.
```

Para solucionarlo se pueden adoptar varios tipos de soluciones.

Lo más elegante sería tratar los objetos nil desde dentro del código, con condiciones, reglas específicas, etc.

Otra opción sería “complicar ligeramente” el inicio para que ese pequeño retardo de tiempo al hilo que lee los valores. Así, con una simple regla ya estaría solucionado:

```
(deffacts obtenir_datos
      (x_pelota (fetch "x_pelota"))
      (y_pelota (fetch "y_pelota"))
)

(defrule comienza
    (inicio)
    =>
    (assert (permitetests))
    )

(defrule persigue-pelota
    (permitetests)
    (x_pelota  ?xm)
    (y_pelota  ?ym)
    (test (<= (distancia ?xm ?ym) 0.5))
=>
      (store "coordx" ?xm)
      (store "coordy" ?ym)
      (store "velocidad" 1)
      (store "chutar" 0.3)
)
```