



Programar juegos de robots
**Jornadas “Programar batallas de
robots con RITA”**

Entrenamiento en la construcción de
batallas con RITA



Instalación de RITA

- ✓ ¿Qué es RITA?
- ✓ Instalando RITA
- ✓ Comenzando con RITA



¿Qué es RITA?

- RITA es un programa que te permitirá **crear juegos de robots**
- El robot tiene la forma de un **TANQUE** de guerra



El “TANQUE” ganador será el primero que destruya al resto, para esto debe atacar y defenderse

Tu Tanque en Combate...



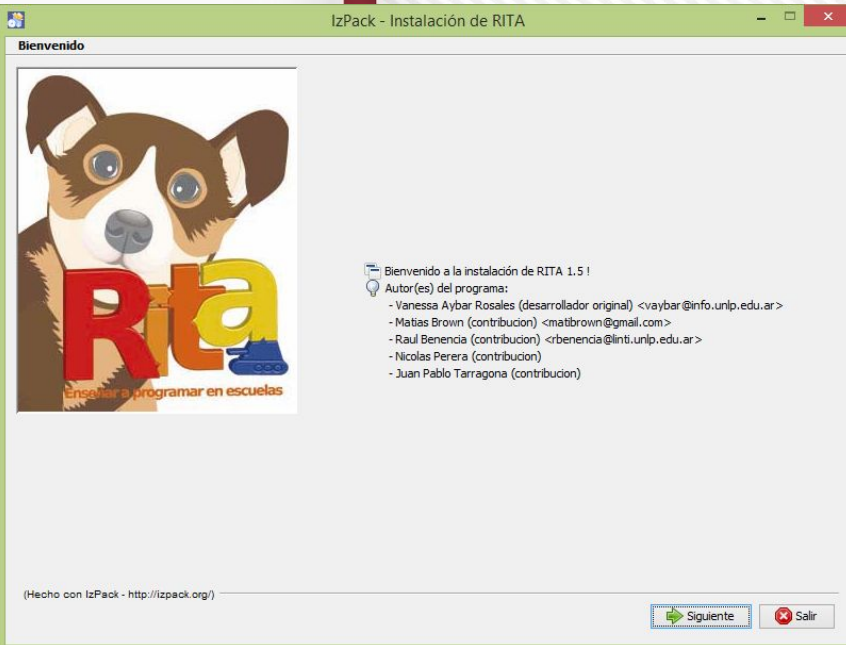
Instalando RITA

Verificar si está instalado Java comprobando si existe la carpeta: **C:\Archivos de programa\Java\jreX** siendo X la versión de Java (6, 7 u 8)

- En caso de que no exista:
 - ✓ Descargar el archivo jre-XuXX-windows-i586.exe de INTERNET
 - ✓ Instalar Java haciendo doble clic en el ejecutable descargado.

Copiar la carpeta RITA

- Ejecutar el archivo **install.bat**



Comenzando con RITA

- Desde el menú Inicio de Windows, **ingrese a la aplicación RITA.**
- La aplicación le solicitará ingresar un nombre para su robot. **Ingrese un nombre.**



RITA - la Aplicación

The screenshot shows the RITA application window titled "robot: 'Robotito' @ RITA - Robot Inventor to Teach Algorithms". The interface includes a menu bar with "Robots", "Edición", and "Información". A search bar labeled "Buscar bloque" is present. On the left, a sidebar lists categories: "Bloques Robocode" (A Codificar, Movimiento, Acciones, Información, Colores), "Sentencias Básicas" (Texto, Lógica, Matemática), and "Nuevas Definiciones" (Método, Variable). The main workspace, labeled "Área de trabajo", contains a script for a robot named "Robotito". The script starts with a "run" block, followed by a "setColors" block with parameters: bodyColor (orange), gunColor (blue), radarColor (white), bulletColor (yellow), and scanArcColor (black). This is followed by a "while" loop with a "condition" of "true". Inside the loop, there is a "do" block containing: "ahead" (steps: 100), "turnGunRight" (degrees: 360), "back" (steps: 100), and "turnGunRight" (degrees: 360). To the right of the script, there are three event-driven blocks: "onScannedRobot" (fire power: 1.0), "onHitByBullet" (back steps: 10), and "onHitWall" (back steps: 20). In the top right corner, a "Minimapa" (mini-map) shows the robot's position. At the bottom right, there is a "Tacho" (trash can icon) and a "Ver Código" button. A label "Para probar el tanque" points to the robot's position in the mini-map.

robot: "Robotito" @ RITA - Robot Inventor to Teach Algorithms

Robots Edición Información

Buscar bloque

Bloques Robocode

- A Codificar
- Movimiento
- Acciones
- Información
- Colores

Sentencias Básicas

- Texto
- Lógica
- Matemática

Nuevas Definiciones

- Método
- Variable

Barra de menú

Área de trabajo

Minimapa

Para probar el tanque

Tacho



Grupos de Bloques Disponibles

Bloques Robocode

A Codificar

Movimiento

Acciones

Información

Colores

Sentencias Básicas

Texto

Lógica

Matemática

Nuevas Definiciones

Método

Variable

Bloques que permiten girar y **mover** el robot, **detectan movimientos** de otros robots, obstáculos, ataques, etc.

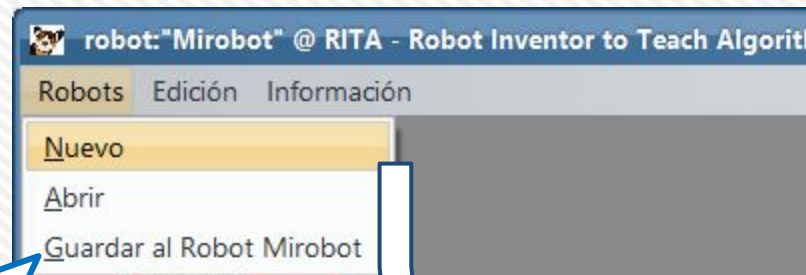
Bloques para **recuperar información** del robot, por ejemplo la energía, las coordenadas cartesianas de la posición del robot en el campo de batalla. También se puede cambiar la apariencia (color) del robot.

Bloques para realizar **cálculos** matemáticos, crear secuencias de acciones en función de **condiciones**, indicar **repeticiones**, etc.

Esta categoría la analizaremos más adelante.

Barra de Menú

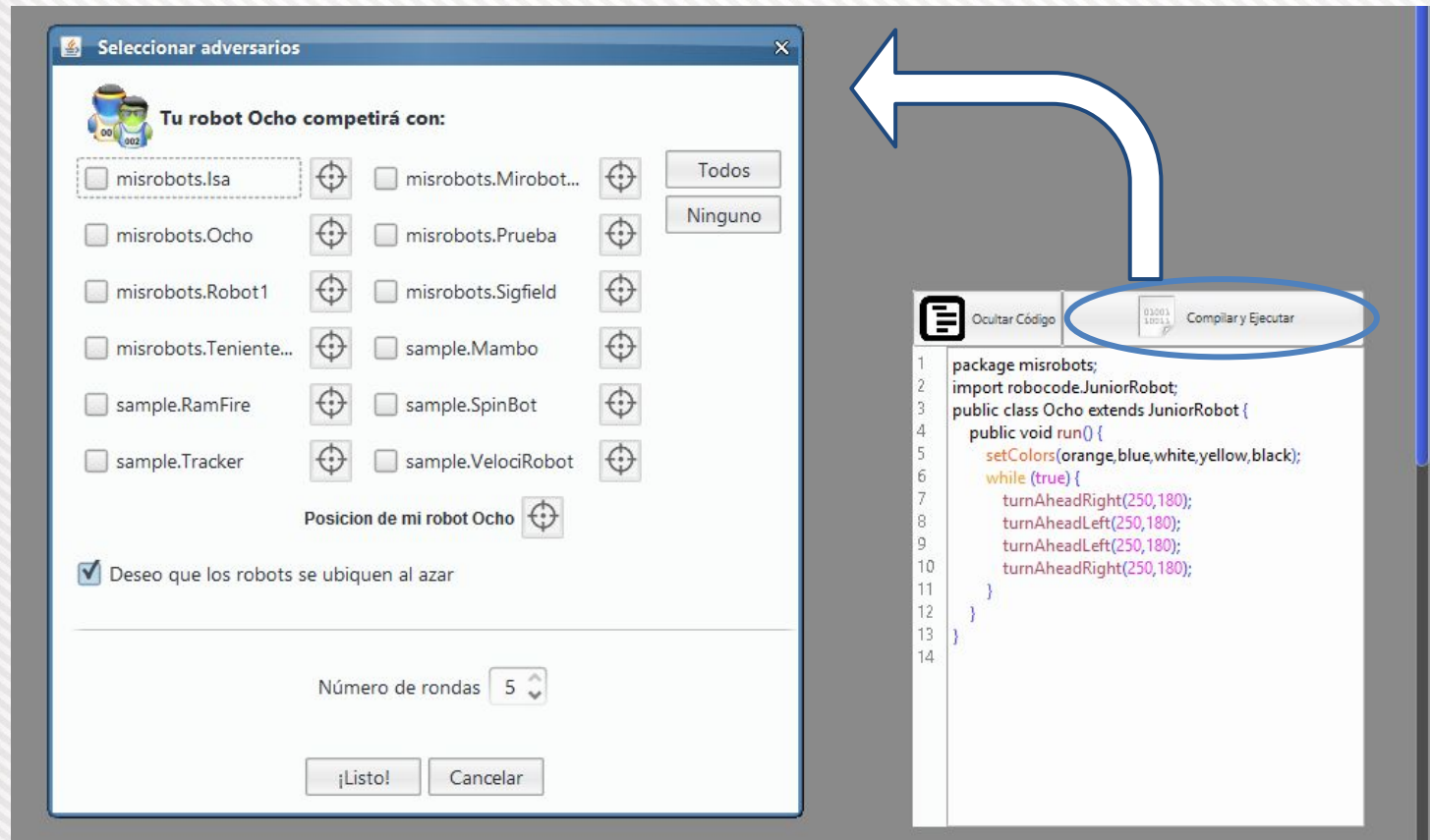
- Arriba y a la izquierda un menú con opciones básicas
 - Opción **Robots**: permite crear uno nuevo, guardarlo o abrir uno ya existente



Si vamos a “Guardar al Robot” usaremos el nombre del robot creado.



Probar el tanque!



The image shows a screenshot of the 'Seleccionar adversarios' (Select Opponents) dialog box and the code editor. A large white arrow points from the 'Compilar y Ejecutar' (Compile and Run) button in the code editor to the dialog box.

Seleccionar adversarios

Tu robot Ocho competirá con:

<input type="checkbox"/> misrobots.Isa		<input type="checkbox"/> misrobots.Mirobot...	
<input type="checkbox"/> misrobots.Ocho		<input type="checkbox"/> misrobots.Prueba	
<input type="checkbox"/> misrobots.Robot1		<input type="checkbox"/> misrobots.Sigfield	
<input type="checkbox"/> misrobots.Teniente...		<input type="checkbox"/> sample.Mambo	
<input type="checkbox"/> sample.RamFire		<input type="checkbox"/> sample.SpinBot	
<input type="checkbox"/> sample.Tracker		<input type="checkbox"/> sample.VelociRobot	

Posicion de mi robot Ocho

☒ Deseo que los robots se ubiquen al azar

Número de rondas: 5

¡Listo! Cancelar

Code Editor:


Ocultar Código

01001
10011

Compilar y Ejecutar

```
1 package misrobots;
2 import robocode.JuniorRobot;
3 public class Ocho extends JuniorRobot {
4     public void run() {
5         setColors(orange,blue,white,yellow,black);
6         while (true) {
7             turnAheadRight(250,180);
8             turnAheadLeft(250,180);
9             turnAheadLeft(250,180);
10            turnAheadRight(250,180);
11        }
12    }
13 }
14
```


Si la posición no es al azar...

 **Posición de Ocho** ✕

Indique en el área donde desea ubicar a Ocho

?	?	?
?	?	?
?	?	?

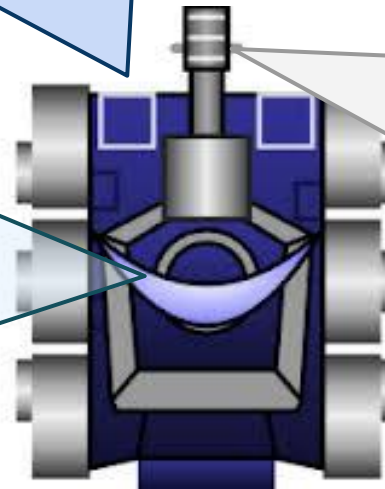
ó ingrese valores X Y

La orientación del robot es ... ☐ Norte ☐ Sur ☐ Este ☐ Oeste

Composición de un TANQUE

Body (Cuerpo): Lleva encima el arma con el radar. Los movimientos que puede hacer el cuerpo son hacia adelante, hacia atrás, girar hacia la izquierda o derecha

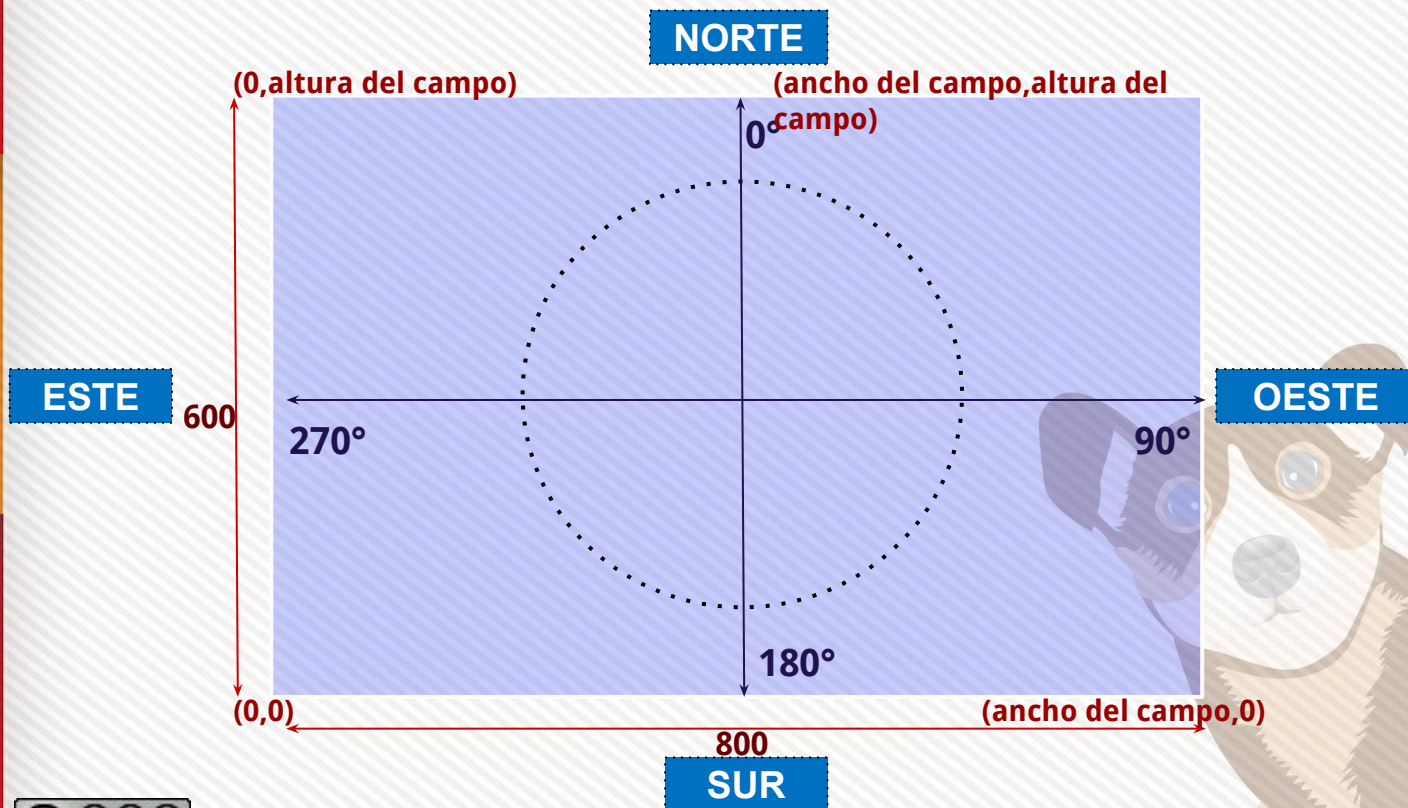
Radar: Montado sobre el arma, es usado para "escanear" otros robots mientras se mueve. El movimiento que puede realizar es hacia la izquierda o derecha. Genera "avisos o señales" cuando un robot es detectado.



Gun (Arma): Montada sobre el cuerpo, es usada para disparar balas. Los movimientos que puede hacer son girar hacia la izquierda o derecha

Coordenadas del Campo de Batalla

- Usaremos el sistema de coordenadas cartesianas
- El campo de batalla tiene 600x800
- Dirección según las agujas del reloj:



Bloques

- Bloques de **Movimientos**
- Bloques de **Información**
- Bloques que son “**Estructuras de Control**”



Ejemplo: Colores del Tanque

El bloque **setColors** permite cambiar el color de varios elementos de nuestro Tanque



Bloques de ejemplo

robot: "Robotito" @ RITA - Robot Inventor to Teach Algorithms

Robots Edición Información

Buscar bloque

Bloques Robocode

- A Codificar
- Movimiento
- Acciones
- Información
- Colores

Sentencias Básicas

- Texto
- Lógica
- Matemática

Nuevas Definiciones

- Método
- Variable

run

setColors

- bodyColor: orange
- gunColor: blue
- radarColor: white
- bulletColor: yellow
- scanArcColor: black

while

- condition: true
- do:
 - ahead steps 100
 - turnGunRight degrees 360
 - back steps 100
 - turnGunRight degrees 360

onScannedRobot

- fire power 1.0

onHitByBullet

- back steps 10

onHitWall

- back steps 20

Robotito

Ver Código

Bloques Disponibles

Categoría Movimiento

Buscador
de bloques



Vamos a encontrar todos los bloques que permiten que nuestro tanque pueda **avanzar, retroceder, girar**, etc.



Bloques ahead y back



cantidad de pasos

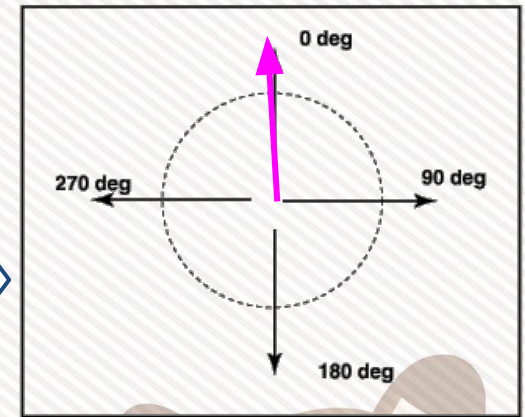
Permite mover el robot **HACIA ADELANTE** la cantidad de pasos que se indiquen



cantidad de pasos

Permite mover el robot **HACIA ATRÁS** la cantidad de pasos que se indiquen

Bloque TURNT



Permite orientar el robot al ángulo indicado respecto del campo de batalla. Es un **POSICIONAMIENTO ABSOLUTO**.

Ejercitación

- Cree un robot llamado **"Cuadrado"** que dibuje un cuadrado alrededor del robot "Mambo", desplazándose 500 pasos por el campo de batalla. Use los bloques **turnTo** y **ahead**.



Ejercitación

- Cree un robot llamado **"Zeta"** dibuje la letra Z. Sólo podrá usar los bloques **turnTo**, **ahead** y **back**



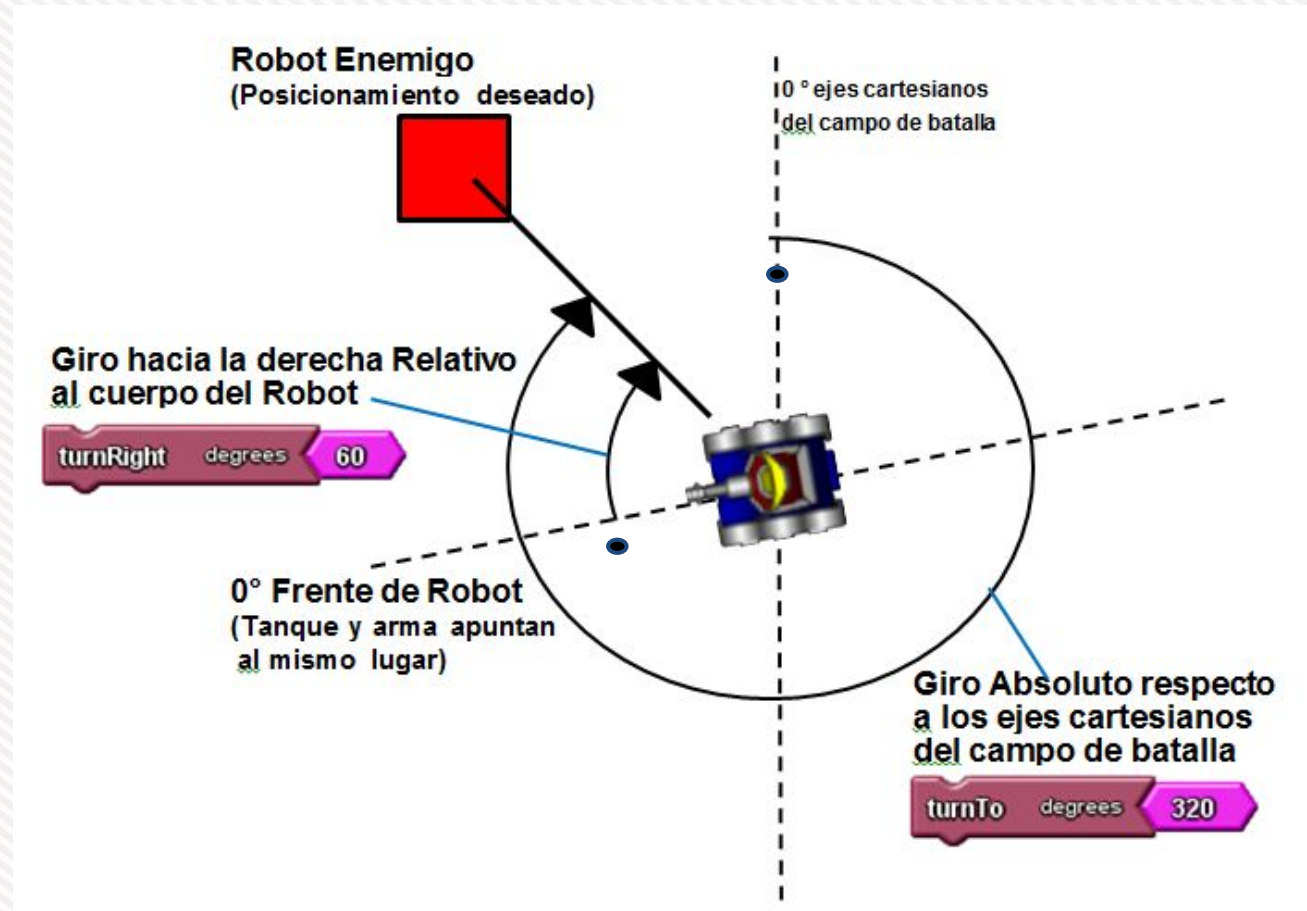


Programar juegos de robots
Jornadas “Programar batallas de
robots con RITA”

Entrenamiento en la construcción de
batallas con RITA



Giros del Cuerpo del Robot



El cuerpo del robot y su arma, podrían tener orientaciones diferentes. Esto permitiría girar el cuerpo y el arma en forma independiente.

Bloques TURNLEFT y TURNRIGHT

turnLeft degrees 45

cantidad de grados

Permite girar el robot **HACIA LA IZQUIERDA** la **CANTIDAD DE GRADOS** que se indiquen

turnRight degrees 45

cantidad de grados

Permite girar el robot **HACIA LA DERECHA** la **CANTIDAD DE GRADOS** que se indiquen

Ejercitación

- Cree un robot llamado **"Cuadrado2"** que dibuje un cuadrado alrededor del robot **"Mambo"**, desplazándose 500 pasos por el campo de batalla. Pero ahora use los bloques **turnRight** y **ahead**.



Bloques **TURN**AHEADLEFT y **TURN**AHEADRIGHT



Permite que el robot **AVANCE** y gire
SIMULTÁNEAMENTE HACIA LA IZQUIERDA

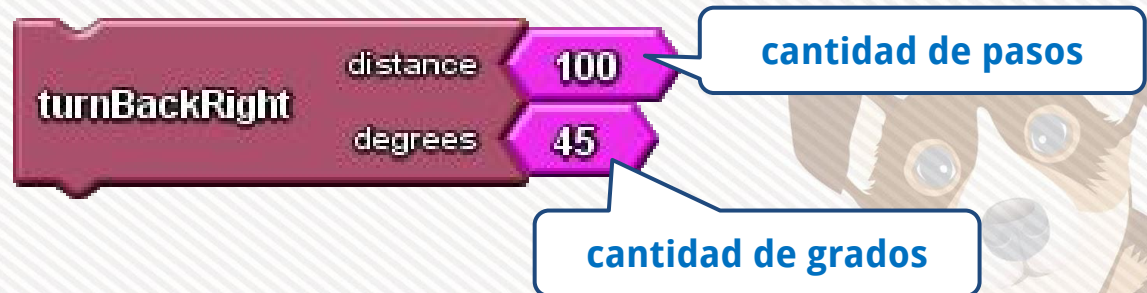


Permite que el robot **AVANCE** y gire
SIMULTÁNEAMENTE HACIA LA DERECHA

Bloques TURNBACKLEFT y TURNBACKRIGHT



Permite que el robot **RETROCEDA** y gire
SIMULTÁNEAMENTE HACIA LA IZQUIERDA



Permite que el robot **RETROCEDA** y gire
SIMULTÁNEAMENTE HACIA LA DERECHA

Ejercitación

- Cree un robot llamado “**Infinito**” que dibuje la siguiente forma:



- Puede usar los bloques **turnAheadLeft** y **turnAheadRight**.



Acciones y Reacciones del Tanque

Nuestro tanque puede **reaccionar** ante algunos **eventos** que puedan sucederle, como ser:

Al **chocar** contra otro **tanque**

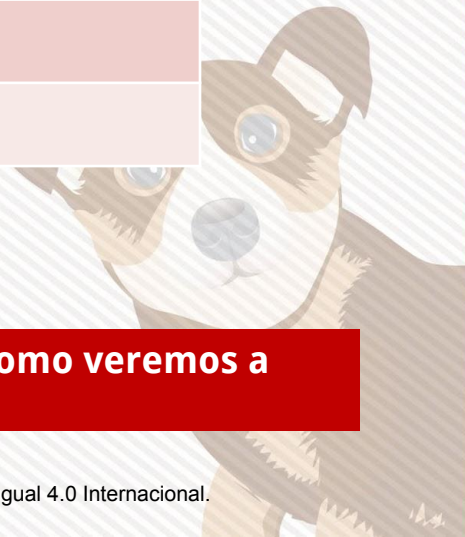
Al ser **alcanzado por una bala** enemiga

Al **chocar** contra un **muro**

Al escanear (**detectar**) un tanque



A esas reacciones las podemos programar, como veremos a continuación



Acciones y Reacciones del Tanque

Al chocar contra otro tanque

onHitRobot

back

steps

100

reacción

Cuando **nuestro tanque choque a otro**, realizará las acciones que le indiquemos. En el ejemplo, el tanque retrocederá 100 pasos.

Acciones y Reacciones del Tanque

Al ser alcanzado por
una bala enemiga

onHitByBullet

back

steps

100

reacción

Cuando **nuestro tanque sea alcanzado por una bala enemiga**, realizará las acciones que le indiquemos. En el ejemplo, el tanque retrocederá 100 pasos.

Acciones y Reacciones del Tanque

Al chocar contra un muro



reacción

Cuando **nuestro tanque choque contra un muro**, realizará las acciones que le indiquemos. En el ejemplo, el tanque retrocederá 100 pasos.

Acciones y Reacciones del Tanque

Al detectar (escanear)
otro tanque

onScannedRobot

fire

power

3.0

reacción

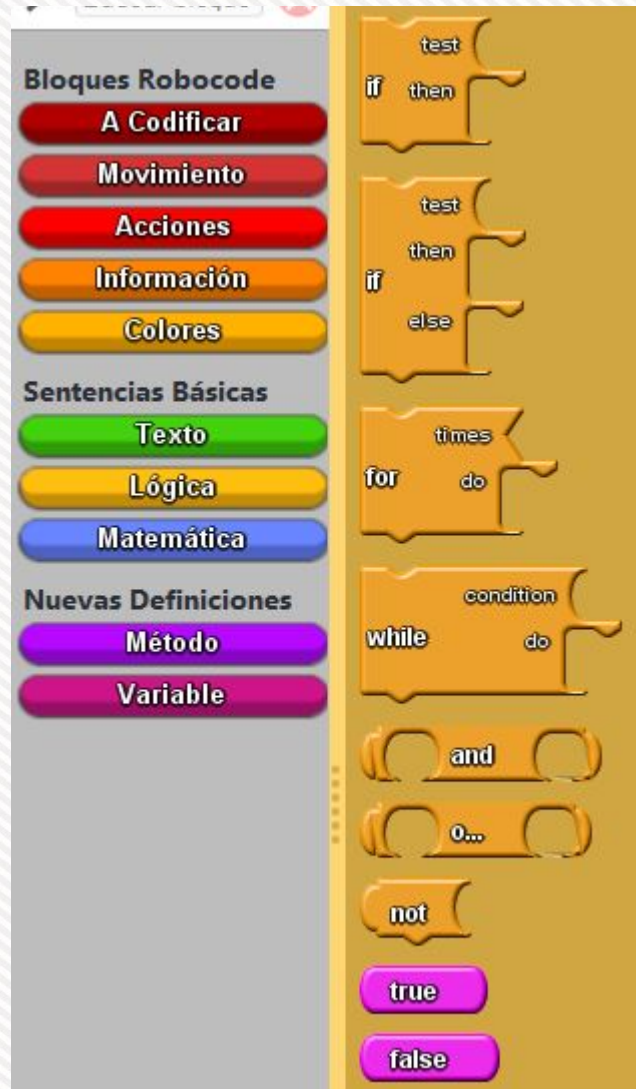
Cuando **nuestro tanque detecte a otro tanque**, realizará las acciones que le indiquemos. En el ejemplo, el tanque disparará con potencia máxima(3).

Ejercitación

- Crear un robot llamado “Destructor” que...
 - **ataque** hacia la dirección por donde fue disparado
 - cuando choca contra la pared **retroceda** 20 pasos **y gire** 180°
 - cuando **escanea** un robot enemigo **dispare** hacia la dirección en donde se encuentra
 - cuando **choca** contra otro robot, **gire** hacia donde se encuentra el enemigo y **dispare**, luego **retroceda** 50 pasos.



Menú Lógica



Las opciones del menú **lógica** permiten evaluar una condición y ejecutar un bloque de comandos asociado .



Menú Lógica

true

Representa el valor
de "verdadero"

false

Representa el valor
de "falso"

Los valores **true** y **false** pueden utilizarse en un bloque **if** ó **while** si se quiere especificar que una condición debe ser verdadera ó falsa.



Menú Lógica - bloques que devuelven V ó F



¿la energía de mi robot es menor a 10?



¿la posición en X de mi robot es 500?



Menú Lógica



AND es un bloque acompañado de 2 condiciones. Si las dos condiciones se cumplen, AND devuelve un valor de verdadero



OR es un bloque acompañado de 2 condiciones. Si ALGUNA de las dos condiciones se cumple, OR devuelve un valor de verdadero

Los bloques condicionales **and** y **or** pueden formar parte de la condición de un bloque **if** ó **while**.



Menú Lógica - bloques que devuelven V ó F

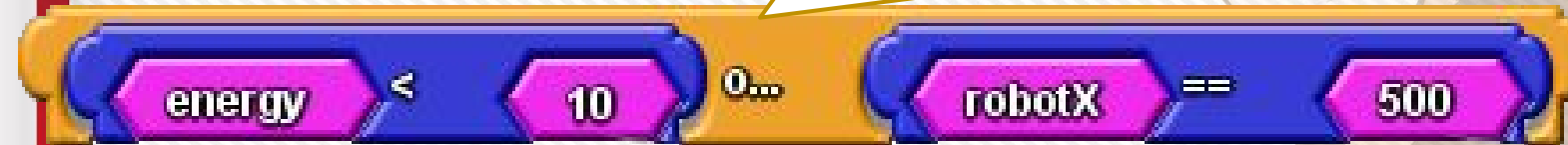
¿la energía de mi robot es menor a 10 **y** la posición en X de mi robot es 500?

AND es un bloque acompañado de 2 condiciones. Si las dos condiciones se cumplen, AND devuelve un valor de verdadero



¿la energía de mi robot es menor a 10 **ó** la posición en X de mi robot es 500?

OR es un bloque acompañado de 2 condiciones. Si **ALGUNA** de las dos condiciones se cumple, OR devuelve un valor de verdadero



Bloques de Información

Para que nuestro Tanque realice determinadas **acciones**, es importante basarnos en la información que podamos obtener de bloques destinados a tal fin.

A estos bloques podemos agruparlos de acuerdo al tipo de información que pueden brindar, en las siguientes categorías:

Para obtener información de **mi robot**

Para obtener información de **otros robots**

Para obtener información del **entorno**

Bloques de Información

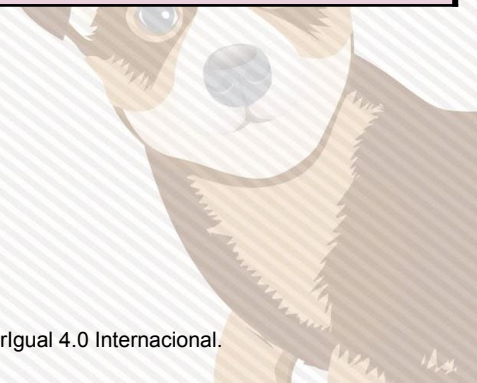
Para obtener información de **mi robot**

robotX	Ubicación horizontal actual del robot (en pixeles)
robotY	Ubicación vertical actual del robot (en pixeles)
energy	Cantidad actual de energía del robot (100 significa carga completa)
heading	Ángulo actual del robot en grados
hitWallBearing	Último ángulo desde el cual el robot golpeo una pared , comparado con el cuerpo del robot
hitWallAngle	Último ángulo desde el cual el robot golpeo una pared
hitRobotBearing	Último ángulo desde el cual el robot alcanzó a otro comparado al cuerpo de éste
hitByBulletAngle	Último ángulo desde el cual el robot fue alcanzado por una bala
hitRobotAngle	Último ángulo desde el cual el robot alcanzó a otro
hitByBulletBearing	Último ángulo desde el cual el robot fue alcanzado por una bala. El ángulo será relativo a su cuerpo

Bloques de Información




Para obtener información de **otros robots**

scannedAngle	Ángulo del robot más cercano que fue escaneado
scannedBearing	Ángulo del robot más cercano que fue escaneado. El ángulo será relativo a su cuerpo
scannedEnergy	Energía actual del robot más cercano que fue escaneado
scannedDistance	Distancia actual del robot más cercano que fue escaneado
scannedHeading	Ángulo que representa el rumbo actual del robot más cercano que fue escaneado
scannedVelocity	Velocidad actual del robot más cercano que fue escaneado



Bloques de Información

Para obtener información del **entorno**

	Contiene la altura del campo de batalla
	Contiene el ancho del campo de batalla
	Contiene la cantidad de otros robots que hay en el campo de batalla



Menú Lógica



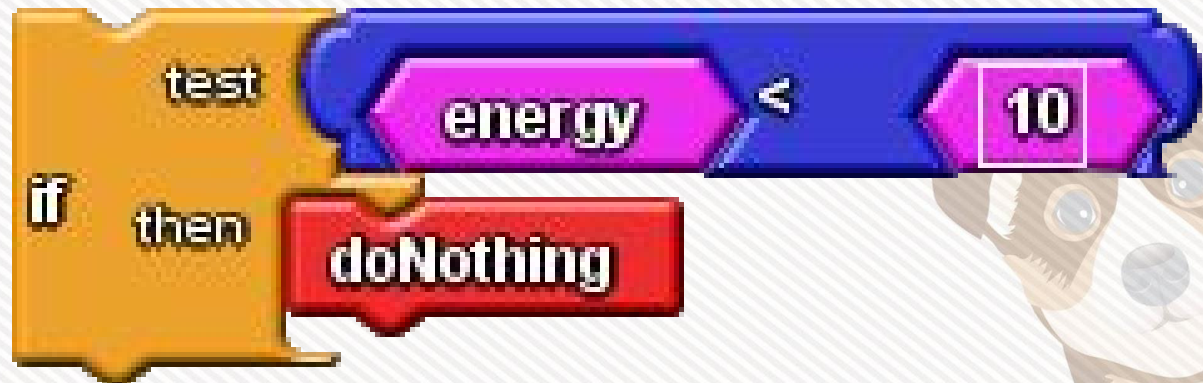
Condición a verificar

Bloque de comandos
a ejecutar cuando se
cumple la condición
(test)

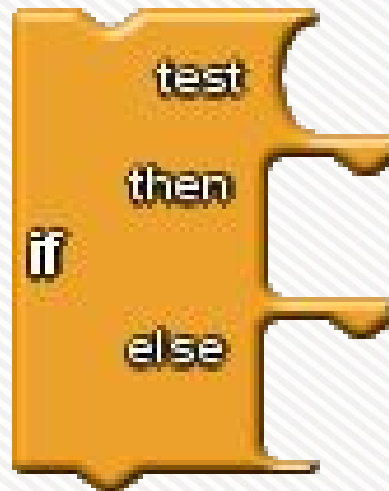
El bloque **if** ("si") se utiliza para evaluar un condición. Si la condición (test) **se cumple** (es verdadera), entonces se ejecutará un **bloque de comandos** (then).

Ejemplo Bloque “If - then”

“Si la **energía del robot es menor a 10**, **entonces** no hago nada “ (porque sino gasto energía y encima corro el riesgo que alguien me detecte!!!!)



Menú Lógica



Condición a verificar

Bloque de comandos
a ejecutar cuando se
cumple la condición
(test)

Bloque de comandos
a ejecutar cuando **NO**
se cumple la
condición (test)

El bloque **if** ("si") se utiliza para evaluar un condición. Si la condición (test) **se cumple** (es verdadera), entonces se ejecutará un **bloque de comandos** (then), de no ser así se ejecuta el otro **bloque de comandos** (else).

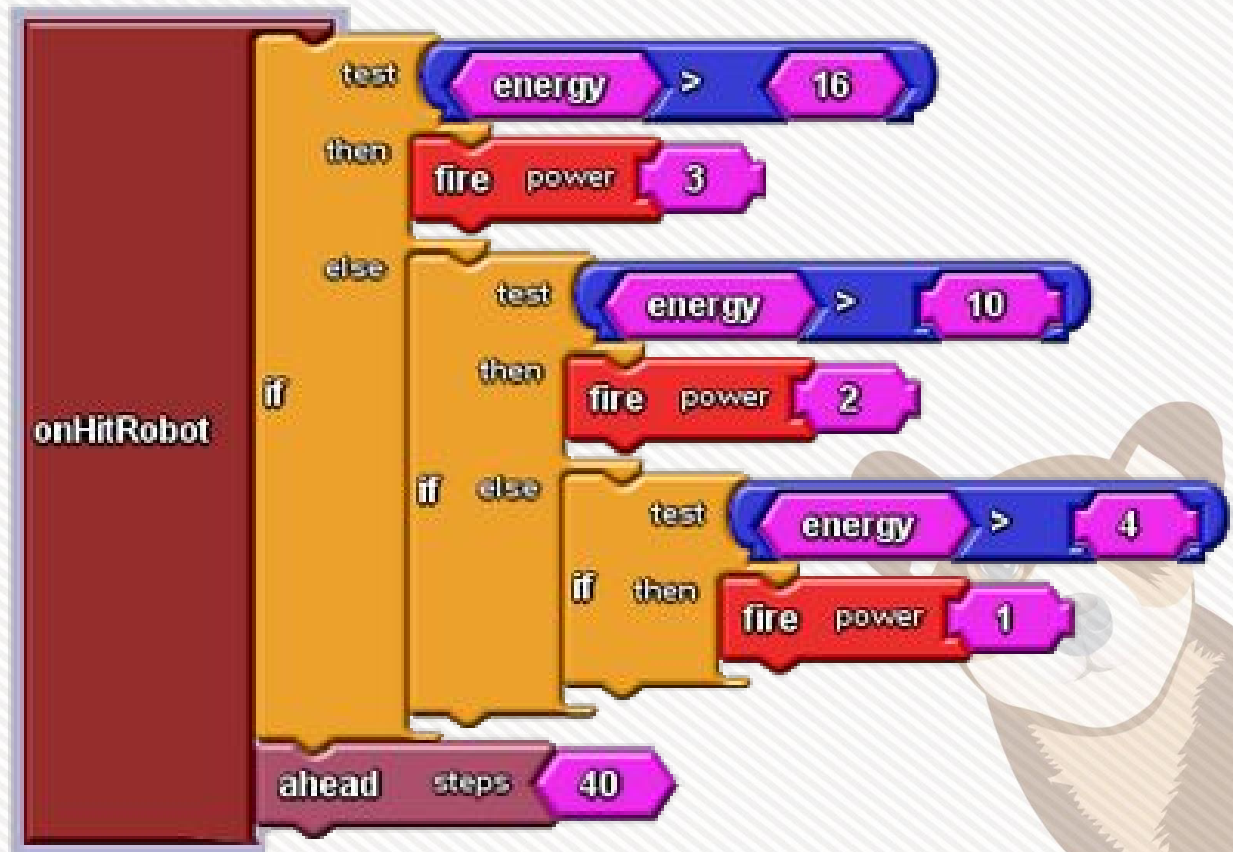
Ejemplo Bloque “If - then - else”

“Si la **energía del robot es menor a 10**, **entonces** no hago nada **sino** me muevo hacia adelante”



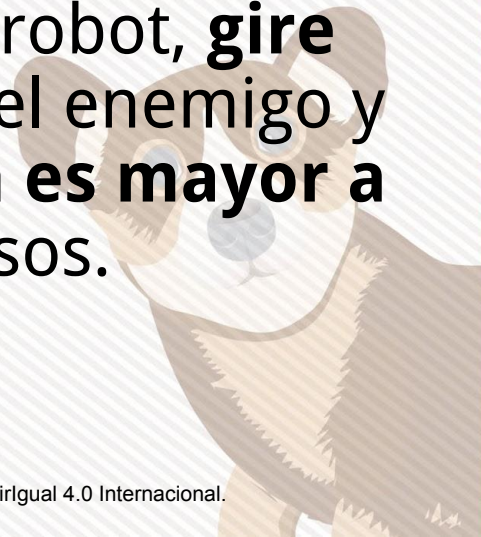
Ejemplo de estrategia de disparo

- Cuando somos impactados por un robot, podemos dispararle según la cantidad de energía que tengamos. Si disparamos siempre con una potencia muy alta, podemos perder mucha energía, por lo que cuando ya no tenemos mucha energía conviene disparar con poco poder.



Ejercitación

- Modifique su robot “Destructor” de modo que...
 - **ataque** hacia la dirección por donde fue disparado **sólo SI la cantidad de robots enemigos es menor a 3**.
 - cuando **escanea** un robot enemigo **dispare** hacia la dirección en donde se encuentra **sólo SI el arma está lista**.
 - cuando **choca** contra otro robot, **gire** hacia donde se encuentra el enemigo y **dispare sólo SI la energía es mayor a 50, SINO retrocede 50 pasos**.



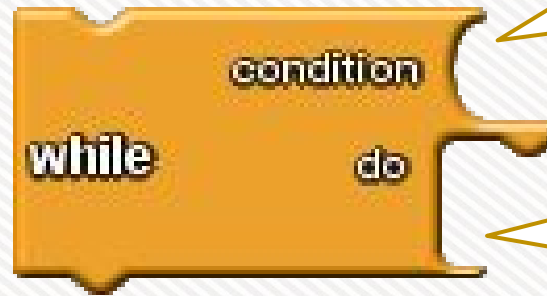


Programar juegos de robots
**Jornadas “Programar batallas de
robots con RITA”**

Entrenamiento en la construcción de
batallas con RITA



Menú Lógica



El bloque **while** permite repetir un bloque de comandos mientras la **condición** sea verdadera (se cumpla).



Menú Lógica



Número de veces que
se repite el bloque
DO

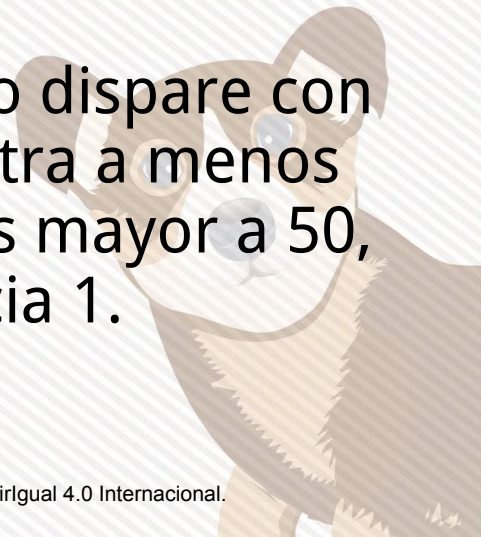
Bloque de comandos
que se repetirá

El bloque **for** permite repetir un bloque de comandos la **cantidad de veces** especificada (times).



Ejercitación

- Modificar el robot **"Destructor"** para que:
 - Cuando escanea un enemigo dispare solamente si éste se encuentra a menos de 100 pasos.
 - Cuando escanea un enemigo dispare solamente si éste se encuentra a menos de 100 pasos y su energía sea mayor a 30.
 - Cuando escanea un enemigo dispare con potencia 3 si éste se encuentra a menos de 100 pasos o su energía es mayor a 50, sino que dispare con potencia 1.





Programar juegos de robots
**Jornadas “Programar batallas de
robots con RITA”**

Estrategias



¿Qué debo tener en cuenta para escribir una estrategia de combate?

- Energía
- Balas y Colisiones
- Algunas ideas para la estrategia



Energía de un tanque

- Nuestro tanque empieza con una cierta cantidad de energía (100)
- Las acciones que realiza hacen que pierda o gane energía



¡Si el tanque se queda sin energía, queda fuera de combate!



Pérdida de energía

Nuestro tanque **pierde energía** cuando...

- Realiza un disparo
- Choca contra un robot
- Choca contra los muros
- Recibe un impacto
- Utiliza intensivamente el radar
- No hace nada (tiempo de inactividad)

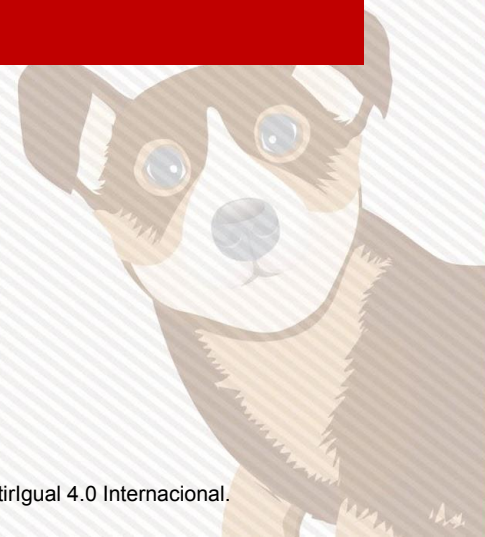


Incremento de energía

- Nuestro robot aumenta su energía cuando acierta con sus balas sobre un tanque enemigo.



¡Los tanques que se dedican sólo a disparar, quedan en estado “agotado”, por ende son deshabilitados!

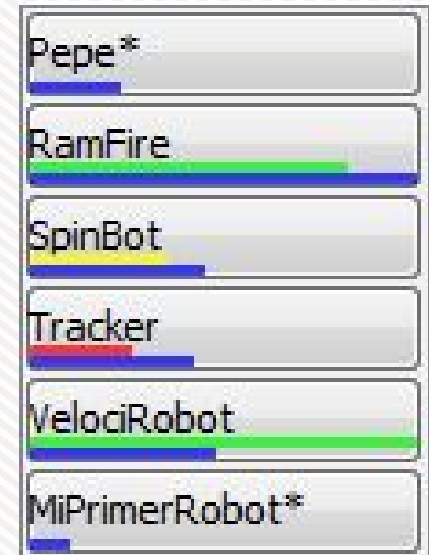


¿Cómo me va en la batalla?

2 barras por robot:

- La primera indica **la energía que me queda:**

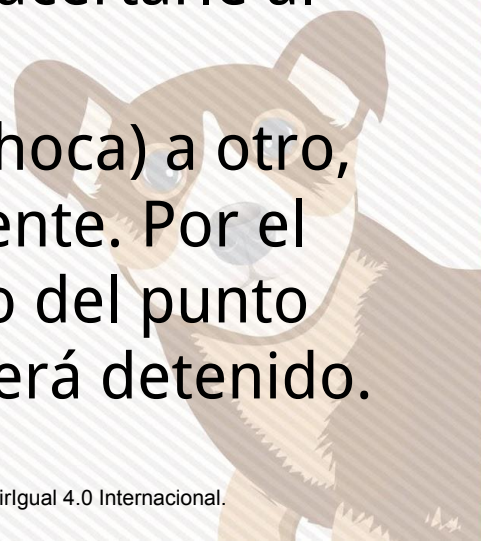
- **>50**
- **20-50**
- **<20**



- La segunda, en **azul**, indica el puntaje que va ganando en el round

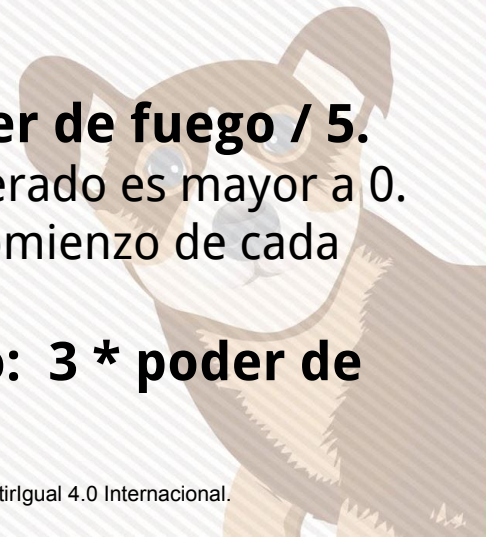
Balas y Colisiones

- Disparar genera **calor** en el arma. Un arma caliente no dispara, hay que esperar que enfríe.
- Las balas viajan a una velocidad constante, por lo que se debe tener en cuenta el tiempo que tardan en llegar a su objetivo, se pierde cierta energía con cada disparo por lo que se debería asegurar de acertarle al enemigo.
- Si nuestro tanque **colisiona** (choca) a otro, será detenido momentáneamente. Por el contrario, si se estaba alejando del punto donde ocurrió la colisión, no será detenido.



Balas

- **Poder de fuego: 1, 2 o 3**
- **Daño: $4 * \text{poder de fuego}$. Daño es la pérdida de energía**
 - Si el poder de fuego = 1, el daño es 4
 - Si el poder de fuego = 2 o 3 hay un daño adicional que se calcula $2 * (\text{poder de fuego} - 1)$
 - Si poder de fuego = 2, el daño = $4 * 2 + 2 * (2 - 1) = 10$
 - Si poder de fuego = 3, el daño = $4 * 3 + 2 * (3 - 1) = 16$
- **Velocidad: $20 - 3 * \text{poder de fuego}$**
 - Velocidad: 17, 14, 11
- **Calor generado por el arma: $1 + \text{poder de fuego} / 5$.**
 - No se puede disparar si el calor generado es mayor a 0.
 - Todas las armas están calientes al comienzo de cada "round".
- **Poder devuelto al acertar al enemigo: $3 * \text{poder de fuego}$**



Choques

Si nuestro robot choca contra otro robot ambos robots pierden un 0.6 de su energía.



¡Si un robot se estaba alejando del punto donde ocurrió la colisión, no será detenido.!



Algunas ideas para la estrategia

- El movimiento del tanque debería ser un tanto **“errático”** o cambiar entre distintos tipos:
 - **Lineal**: hacia adelante/atrás. Sería muy fácil para los robots enemigos predecir la futura posición, en base a la dirección y velocidad.
 - **Circular**: avanzar y girar algunos grados siempre en la misma dirección (derecha o izquierda)
 - **Oscilatorio**: avanzar y girar a izquierda/derecha
- **Alejar** nuestro tanque de las zonas donde detectamos enemigos



Algunas ideas para la estrategia

Mantener girando el radar de modo de escanear la mayor cantidad de tanques. El radar gira cuando el arma gira directa o indirectamente. El radar detecta velocidad, posición, orientación y energía restante del otro robot.

turnGunRight

degrees

360

turnAheadRight

distance

100

degrees

360

Algunas ideas para la estrategia

Utilizar el radar para escanear la ubicación de los demás robots, elegir el más cercano y dispararle. Hay que elegir bien la potencia del disparo basado en la distancia del enemigo.



No se puede disparar muy seguido, porque el cañón se calienta, por lo que los tiros deberían ser precisos.

Algunas ideas para la estrategia

En el método onHitByBullet() que se ejecuta cuando se recibe un disparo, es común reaccionar disparando hacia donde vino el disparo, por lo que luego deberíamos movernos.



Algunas ideas para la estrategia

Al detectar un tanque, girar el arma manteniendo el cuerpo del tanque en posición de escape y no enfrente del enemigo.



Cambiar de dirección cuando ocurre una colisión.



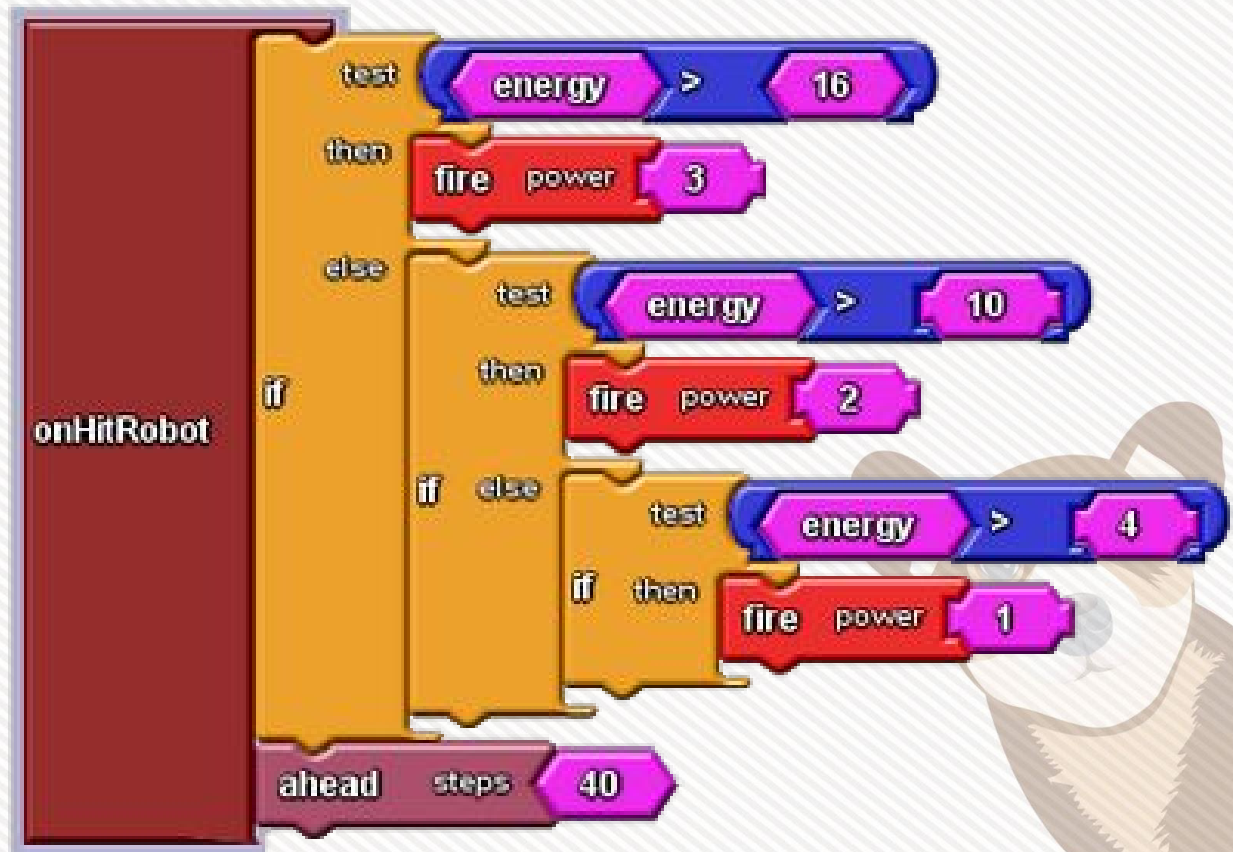
Algunas ideas para la estrategia

En el evento onHitRobot() que se dispara cuando chocamos contra un robot se tiene la oportunidad perfecta para dispararle con todo el poder.



Ejemplo de estrategia de disparo

- Cuando somos impactados por un robot, podemos dispararle según la cantidad de energía que tengamos. Si disparamos siempre con una potencia muy alta, podemos perder mucha energía, por lo que cuando ya no tenemos mucha energía conviene disparar con poco poder.



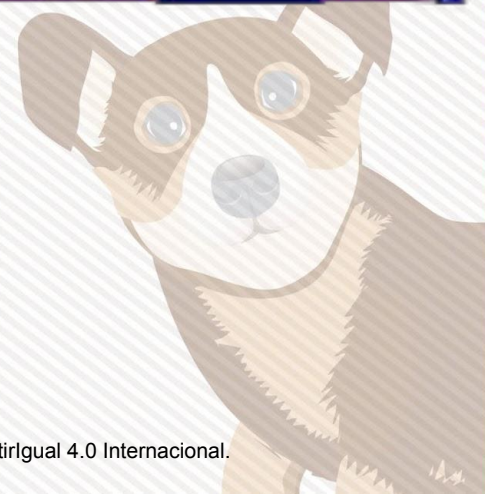
Dr. Jekyll y Mr. Hyde

- **Estrategia de Hyde:** se trata de perseguir al enemigo y dispararle de cerca, con toda la fuerza que se pueda. Aunque se pierde más energía, también es más seguro acertarle y hacerle bastante daño.



Dr. Jekyll y Mr. Hyde

- **Estrategia de Jekyll:** es una estrategia defensiva que consiste en moverse perpendicularmente al cañón del enemigo de manera de poder alejarse lo más rápido posible y no ser impactado.



¿Qué estrategia usamos?

Es muy conveniente fusionar varias estrategias, de manera que nuestro robot sea lo suficientemente inteligente para adaptarse y pueda cambiar dinámicamente ciertos comportamientos, dependiendo de las circunstancias.



¡Muchas Gracias!

Claudia Queiruga: claudiaq@info.unlp.edu.ar

Matías Brown: mbrown@linti.unlp.edu.ar

Isabel Kimura: ikimura@linti.unlp.edu.ar

Vanessa Aybar: vaybar@info.unlp.edu.ar

JETS: <http://jets.linti.unlp.edu.ar/>

