R.I.T.A.

Agenda

- Objetivo de la presentación
- Robocode
- R.I.T.A.

Objetivo de la presentación

- Aprender el uso de R.I.T.A. como herramienta que permita transmitir conceptos básicos de programación, sin necesidad de tener conocimiento de un lenguaje de programación en particular.

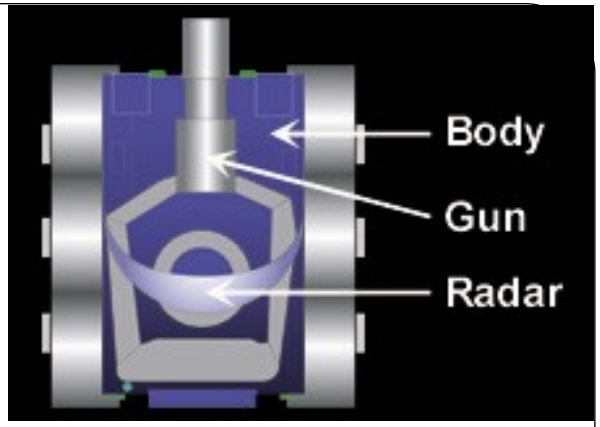
Robocode

- ¿Qué es Robocode?
- ¿Cómo está conformado un robot?
- ¿Qué puede hacer un robot?
- ¿Qué facilita Robocode?
- Consideraciones
- Estrategias
- Más información
- Ventajas
- Desventajas

¿Qué es Robocode?

- Proyecto Open Source iniciado en el año 2000
- Robocode permite la competencia entre robots y equipos de robots. Cada participante escribe su propio robot y compiten entre ellos
- Hay distintos tipos de Robots, en particular se sugiere empezar con el “JuniorRobot” para aprender a programar
- Robocode está escrito en Java, existe una versión para .Net

¿Qué es un robot?



Un Robot es un **Tanque** formado por:

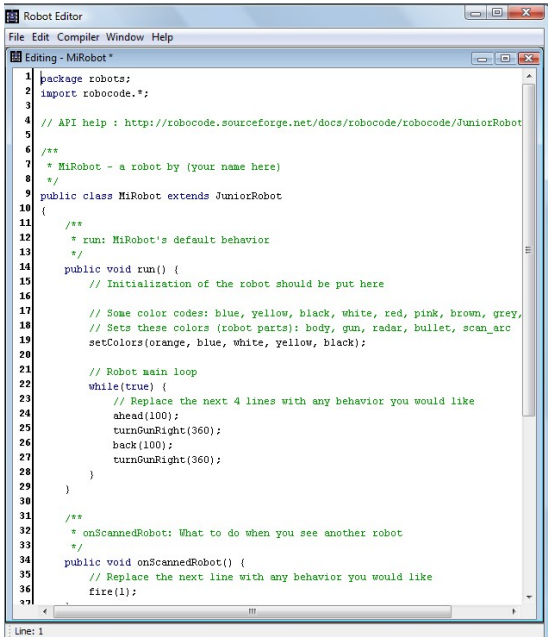
- **Body (Cuerpo):** Lleva encima el arma con el radar. Los movimientos que puede hacer el cuerpo son hacia adelante, hacia atrás, hacia la izquierda o derecha.
- **Gun (Arma):** Montada sobre el cuerpo, es usada para disparar balas. Los movimientos que puede hacer son hacia la izquierda o derecha.
- **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. Este radar genera “avisos o señales” cuando un robot es detectado.

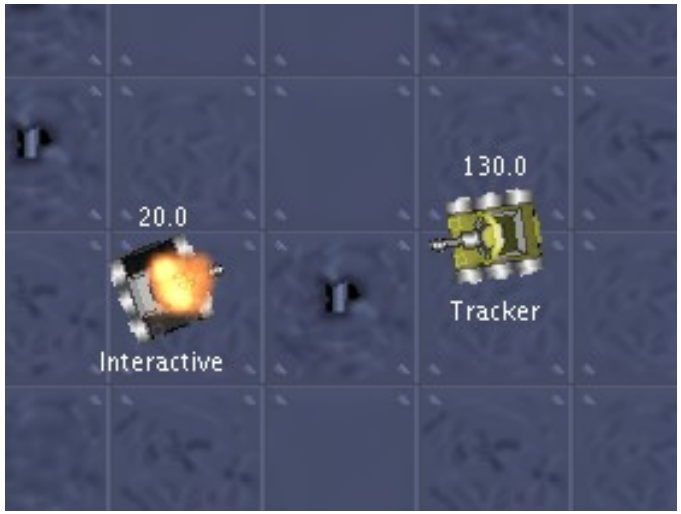
¿Qué puede hacer un Robot?

- Un Robot
  - Se **inicializa**
  - **Busca** a otros robots
  - **Ataca** a otros robots
  - **Se defiende** de otros robots

¿Qué facilita Robocode?

- Un editor de texto para escribir el comportamiento del robot





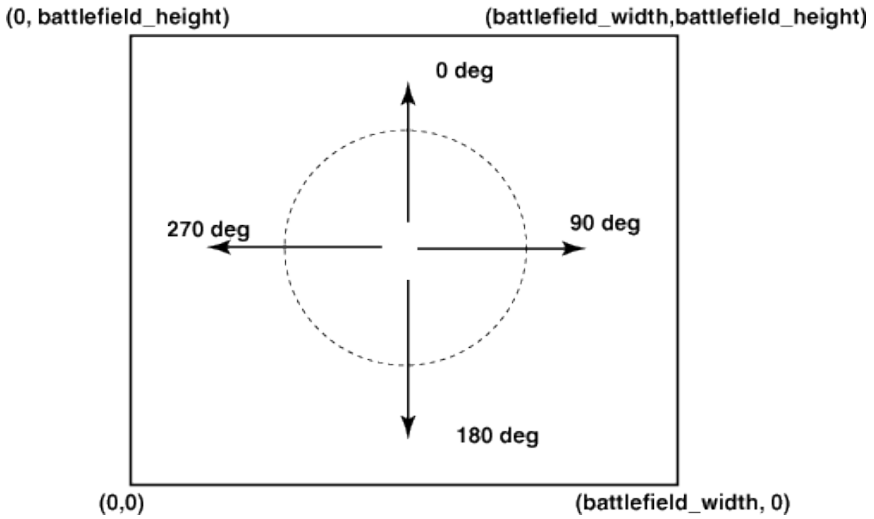
- Un campo de batalla

Robocode-consideraciones

Coordenadas y convenciones de dirección

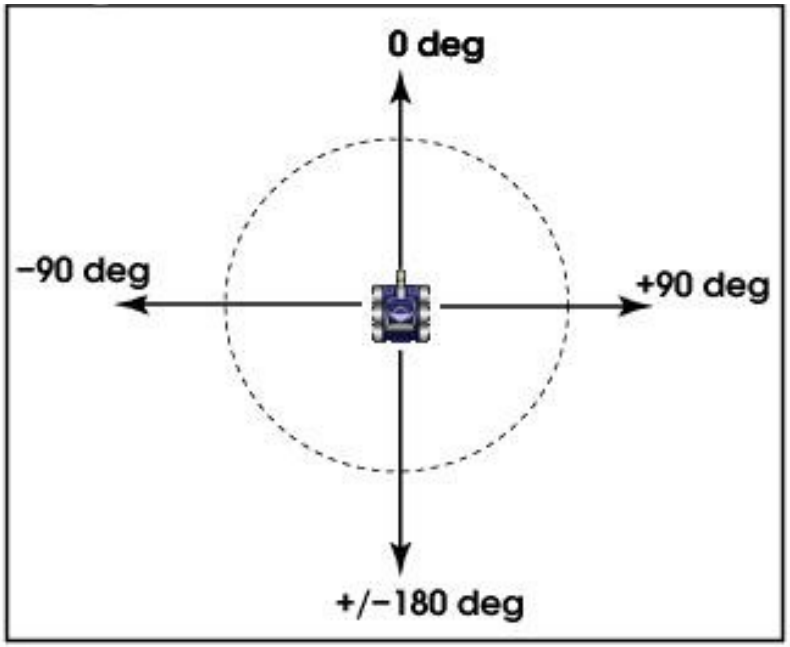
- Robocode usa el sistema de coordenadas cartesianas. Es decir la coordenada (0, 0) está ubicada abajo a la izquierda del campo de batalla.

- Dirección según las agujas del reloj: 0 / 360 grados hacia el "Norte", 90 grados hacia al “Este", 180 grados hacia el “Sur", y 270 grados hacia el “Oeste".



Bearing (ángulo relativo al cuerpo del robot)

- Rango de grado, va desde los -180 a 180 grados
- Es relativa a la posición en grados a la orientación de nuestro robot



Balas

- Daño: 4 * poder de fuego.
  - Si poder de fuego > 1, hace un daño adicional = 2 * (power - 1)
- Velocidad: 20 - 3 * poder de fuego
- Calor generado por el arma: 1 + poder de fuego/ 5.
  - No se puede disparar si el calor generado > 0.
  - Todas las armas están calientes al comienzo de cada “round”.
- Poder devuelto al acertar al enemigo: 3 * poder de fuego

Colisiones

- Si colisiona con otro robot:
  - Cada robot recibe 0.6 de daño.
  - Si un robot se estaba alejando del punto donde ocurrió la colisión, no será detenido.

- Si colisiona con un muro:
  - Los AdvancedRobots pierden abs(velocidad) * 0.5 - 1; (nunca es menor a 0).

Pérdida de energía

- Al producir un disparo.
- Chocar contra un robot.
- Chocar contra los muros.
- Recibir un impacto. Esta es la mayor penalización que ocurre.
- Utilización intensiva del radar.
- Tiempo de inactividad.

Incremento de energía

- Nuestro robot empieza con una cierta cantidad de energía (100)
- Acierto sobre el enemigo. Si una bala de nuestro robot acierta sobre un robot enemigo, nuestro robot recibe una cantidad de energía adicional.
- Los robots en estado “agotado” por disparar demasiado son deshabilitados (Si alcanza a otro robot, gana nuevamente energía)

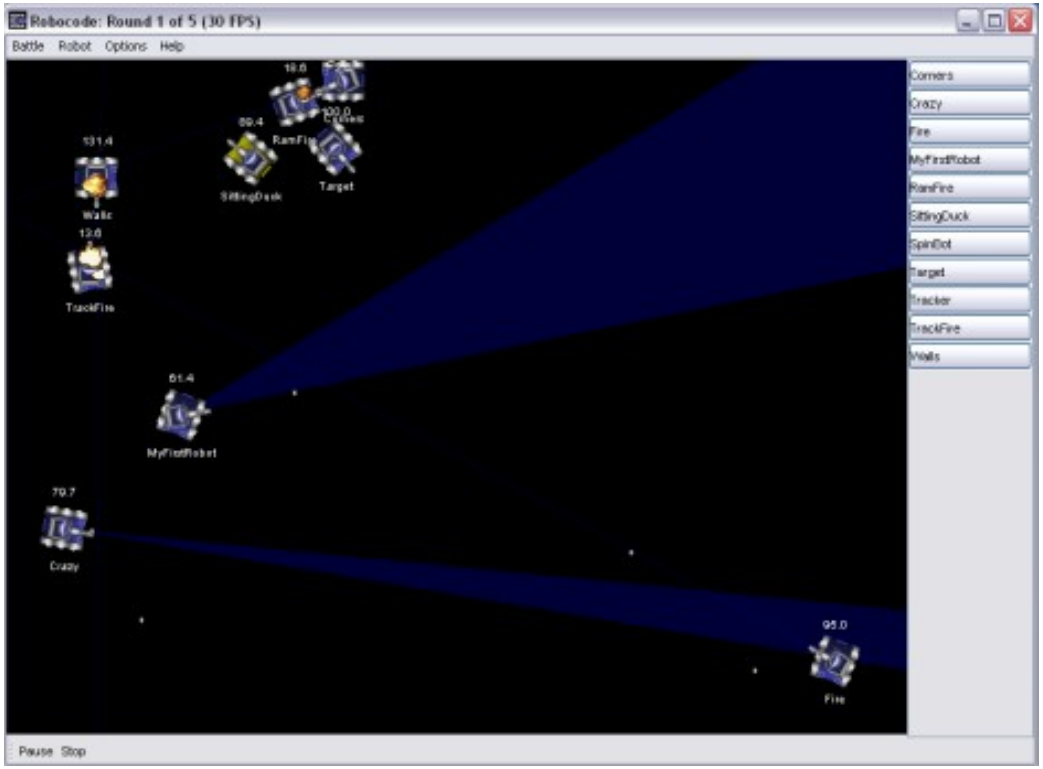
Robocode-estrategias

Estrategia de radar

- **Simple**. Apuntar el radar en la misma dirección que el arma.
- **Circular**. Mantener el radar girando constantemente.
- **Tracking**. Mantener el radar fijo sobre un objetivo.
- **Optimal**. Mover el radar de modo de escanear todos los robots tan rápido como sea posible.

Estrategia de radar

- Escanear consume tiempo



Un escaneo simple, 360 grados

```
public class MiPrimerRobot extends JuniorRobot
  {
        public void run() {
                while (true) {
                ahead(100);
                turnGunRight(360);
           }
      }
      public void onScannedRobot() {
         fire(1);
      }
}
```

Estrategias de movimiento

- **Movimiento en una línea recta**. Fácil de predecir por enemigos.
- **Movimiento en curva (circular)**. Evita ser alcanzado por un robot que apunta de manera estacionaria o lineal. Fácil de predecir por enemigos.
- **Moverse hacia adelante o atrás, oscilante**.
- **Moverse en una dirección random**. En general disminuye el riesgo de ser alcanzado por cualquier estrategia, aunque aumenta las probabilidades de choque.
- **Anti-gravedad:** mantenerse lejos de áreas peligrosas

Estrategias de disparo

- **Estacionario**. Dispara hacia la actual ubicación del objetivo. Es la más simple y menos efectiva de las estrategias.
- **Linear**. Dispara donde se asume estará el objetivo.
  - Considera que se mueve a una velocidad constante en linea recta.
  - Efectivo cuando el objetivo está cerca.
  - Los movimientos de un robot pueden ser aproximados en una linea recta por un periodo muy corto de tiempo. Se vuelven poco confiables rapidamente.
- **Circular**. Dispara donde se asume estará el objetivo.
  - Asume que se mueve a una velocidad constante y a una velocidad angular constante.
  - Funciona un poco mejor que la estrategia lineal.
  - Recomendaciòn: cambiar la velocidad rápidamente, detenerse y arrancar, moverse hacia atrás y adelante rápidamente.

Estrategias de disparo

- **Oscilatorio**. Se asume que el robot objetivo está moviendose hacia atrás y adelante continuamente.
  - Pocos robots implementan esta estrategia.

- **Movimiento “pattern matching”**. Dispara prediciendo una posición.
  - Guarda los movimientos pasados del robot y asume que éste se moverá siguiendo ese patrón.
  - Es mejor que las estrategias anteriores
  - Consume más tiempo de procesamiento por las búsquedas exhaustivas.

Ejecución de Robocode

Una batalla en Robocode repite los siguientes pasos

1. La vista en pantalla es (re)pintada.
2. Todos los robots ejecutar su código hasta que se tomen medidas (y entonces hacen una pausa).
3. El tiempo es actualizado (tiempo= tiempo + 1).
4. Todas las balas se mueven y se chequea si hay colisión. Incluye la acción de disparar la bala.

Ejecución de Robocode

1. Todos los robots se mueven (arma, radar, orientación, aceleración, velocidad, distancia, en ese orden).
2. Todos los robots escanean (y colectan mensajes de equipo).
3. Todos los robots toman nuevas medidas
4. Cada robot procesa su cola de eventos.

Mi primer Robot

```
public class MiPrimerRobot extends JuniorRobot
   {
   public void run() {
         while(true) {
                ahead(100);
                turnGunRight(360);
                back(100);
                turnGunRight(360);
          }
       public void onScannedRobot(ScannedRobotEvent e)
         { fire(1);
       }
   }
}
```

Más Información... Tiempo y distancia en Robocode

- Tiempo (t): es medido en "ticks". Cada robot tiene un turno por “tick”. 1 tick = 1 turno.
- Medidas de distancia: En general en pixels. Hay 2 excepciones:
  - Todas las distancias tienen precision en numeros reales, es decir, es posible moverse una fraccion de pixel
  - Robocode escala el campo de batalla para encajar en la pantalla.

Más Información... Física del movimiento del Robot

- Aceleracion (**a**):
  - Los Robots aceleran a razón de 1 pixel/turno
  - Los Robots desaceleran a razón de 2 pixels/turno
  - Robocode determina la aceleración, basado en la distancia que el robot se quiere mover
- Ecuación de la Velocidad (**v**): v = at.
  - La velocidad nunca puede exceder 8 pixels/turno.
- Ecuación de la distancia (**d**): d = vt.

Más Información... Rotación del Robot, Arma y Radar

- Máxima velocidad de rotación de un robot:
  - (10 - 0.75 * abs(velocidad)) grados/ turno.
  - A más velocidad, más lento el turno.
- Máxima velocidad de rotación de arma:
  - 20 grados/turno (sumado a la velocidad de rotación del robot)
- Máxima velocidad de rotación de radar:
  - 45 grados/turno. (sumado a velocidad de rotación de radar)

Ventajas de Robocode

- **Motivación.** Los alumnos se sienten motivados por escribir el robot que le gane al resto de la clase. Programar se convierte en el medio y no en el objetivo.

Desventajas de Robocode

- **Obstáculo.** Es necesario saber un lenguaje de programación, y del proceso de compilación previo a la ejecución.

- El alumno no sólo debe pensar en la idea sino en como escribirla con las **particularidades del lenguaje**.

R.I.T.A.

- Objetivo
- ¿Qué es R.I.T.A.?
- R.I.T.A - la aplicación
- Ejemplo de implementación de una estrategia

Objetivo

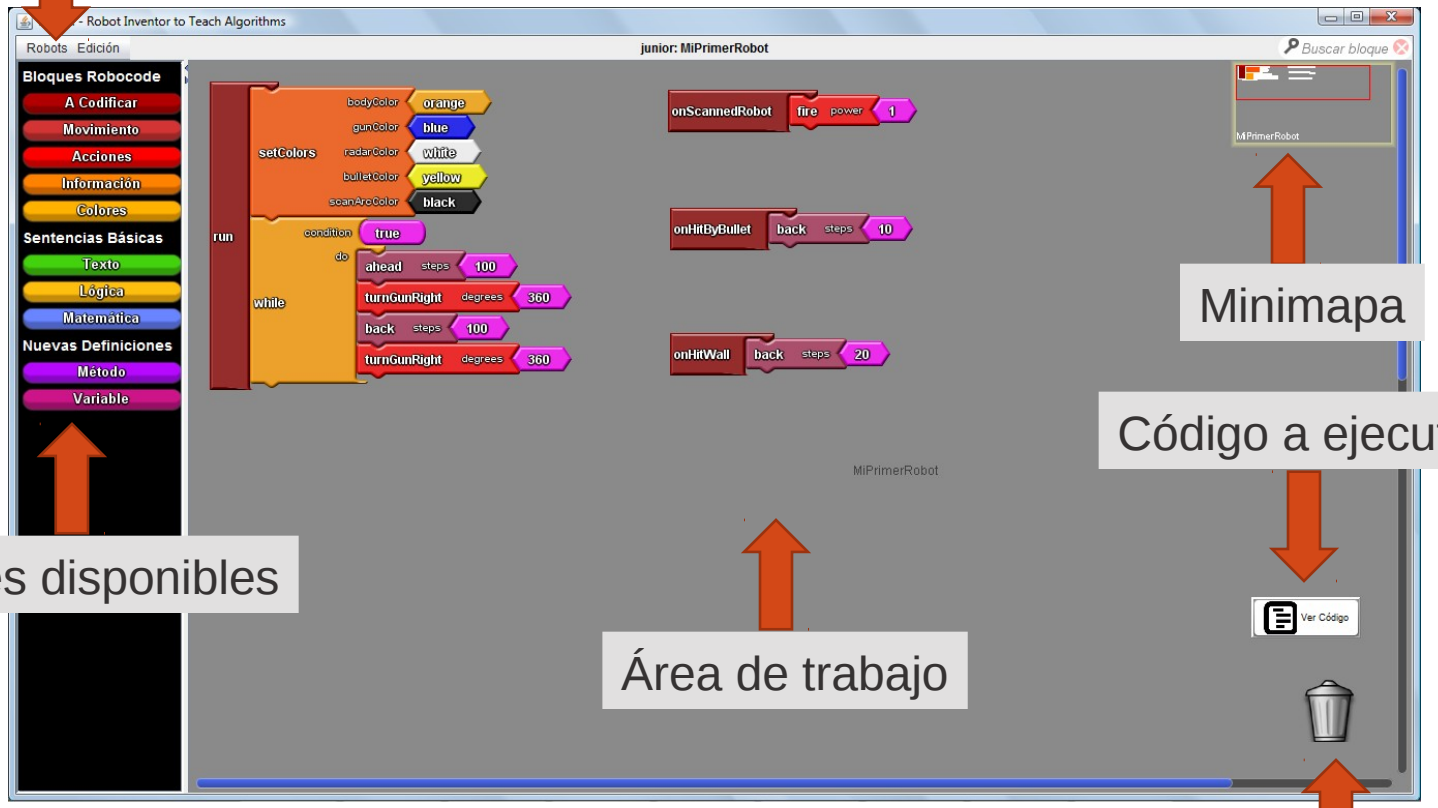
- Permitir la enseñanza de programación a alumnos sin conocimiento previo, eliminando en una primer instancia el aprendizaje de un lenguaje de programación específico para concentrarse en la lógica de la programación.

¿Qué es R.I.T.A.?

- Robot Inventor to Teach Algorithms
  - Permite escribir la estrategia de un robot mediante la técnica de programación en bloques
  - En todo momento el alumno puede ver la equivalencia entre sus bloques y el código Java equivalente generado por R.I.T.A.
  - Ambiente integrado que permite implementar la estrategia con bloques y ponerla a prueba enfrentando al robot creado en el campo de batalla

R.I.T.A - la aplicación

Barra de menú



Minimapa

Código a ejecutar

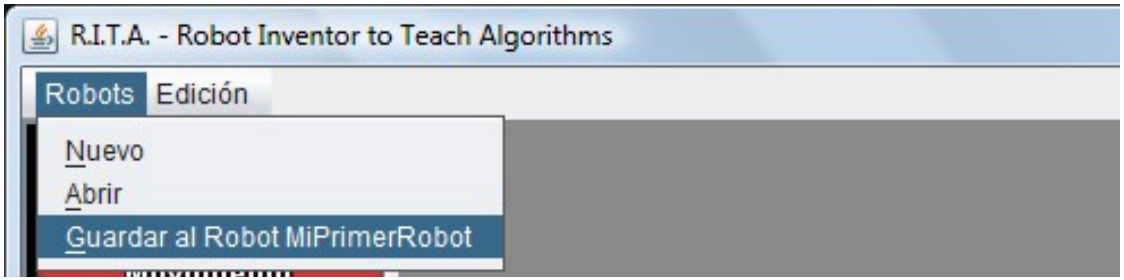
Bloques disponibles

Área de trabajo

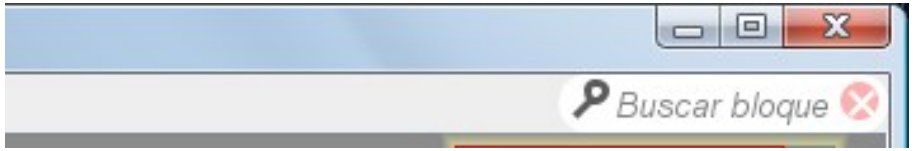
Tacho

Barra de menú

- A la izquierda un menú con opciones básicas para creacion de robots y guardarlos
  - Permite crear un nuevo robot, guardarlo o abrir uno ya existente



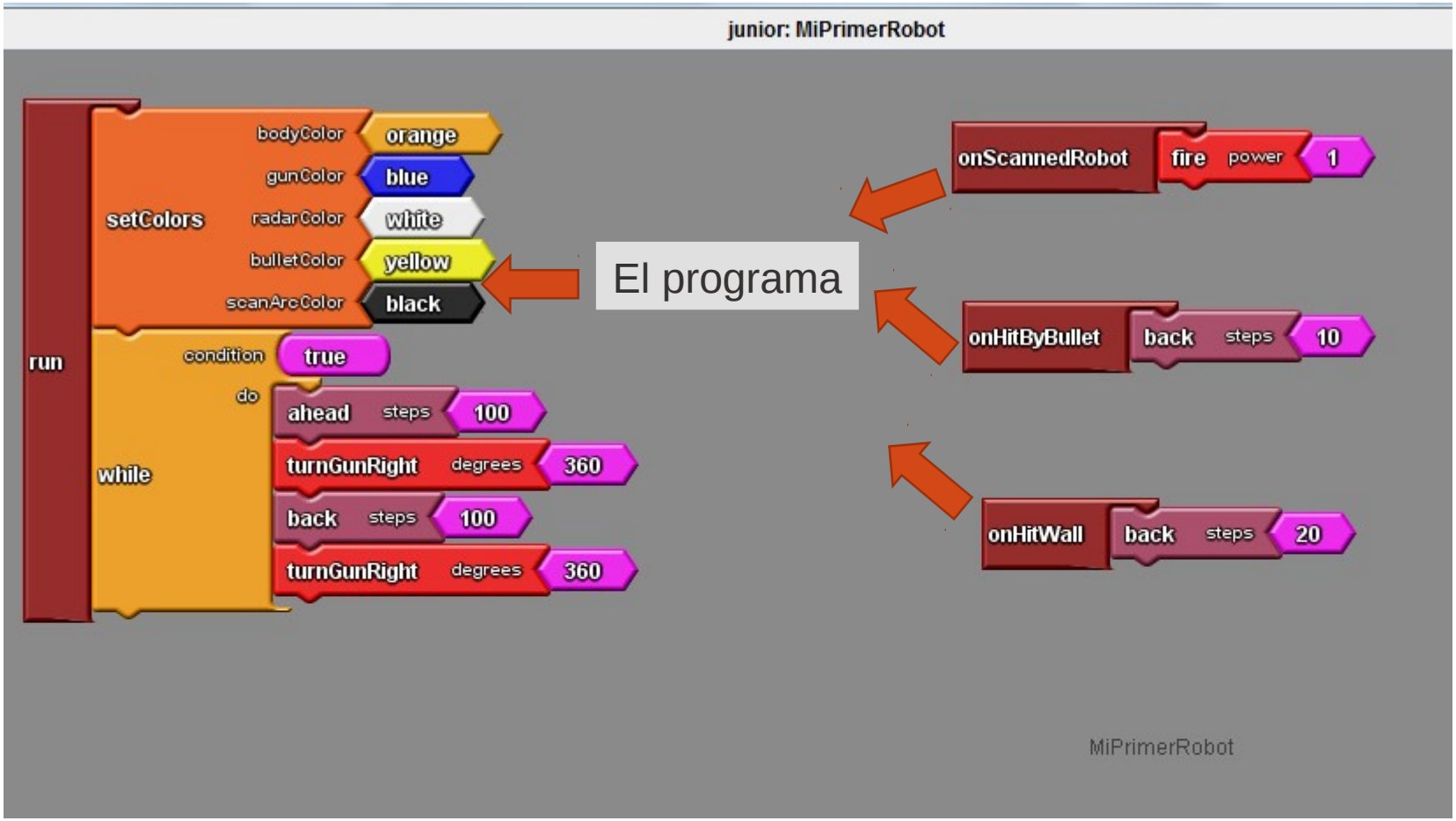
- A la derecha la opción de Búsqueda



Bloques disponibles

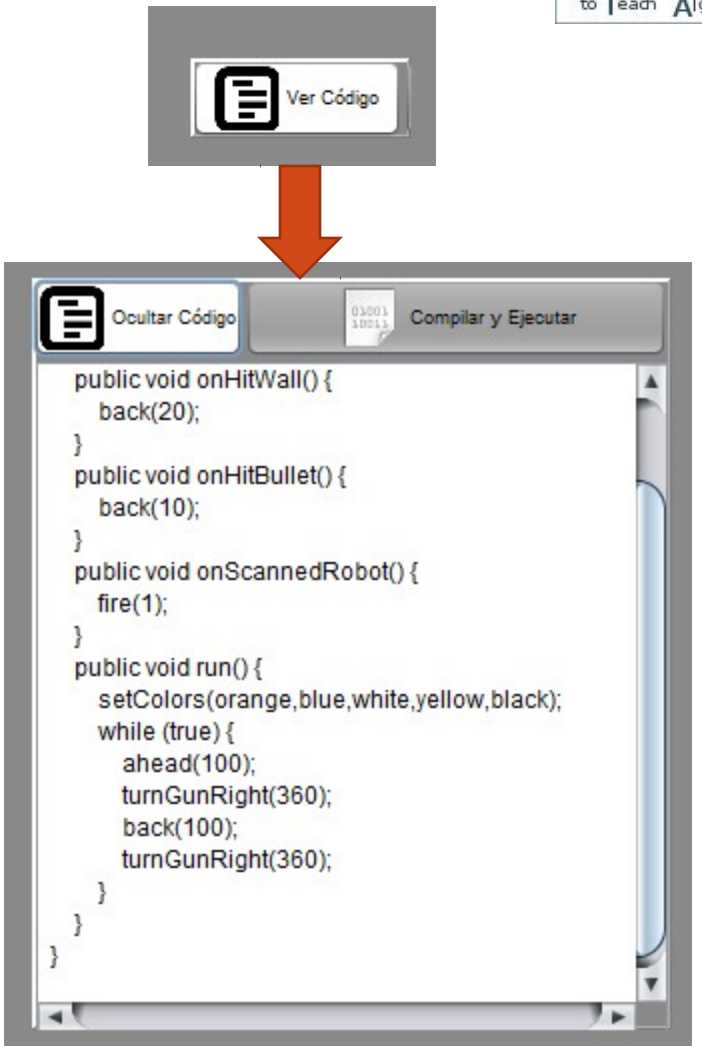
- Agrupados por funcionalidad
- Contienen los bloques básicos que permiten construir la estrategia de mi robot

Área de trabajo

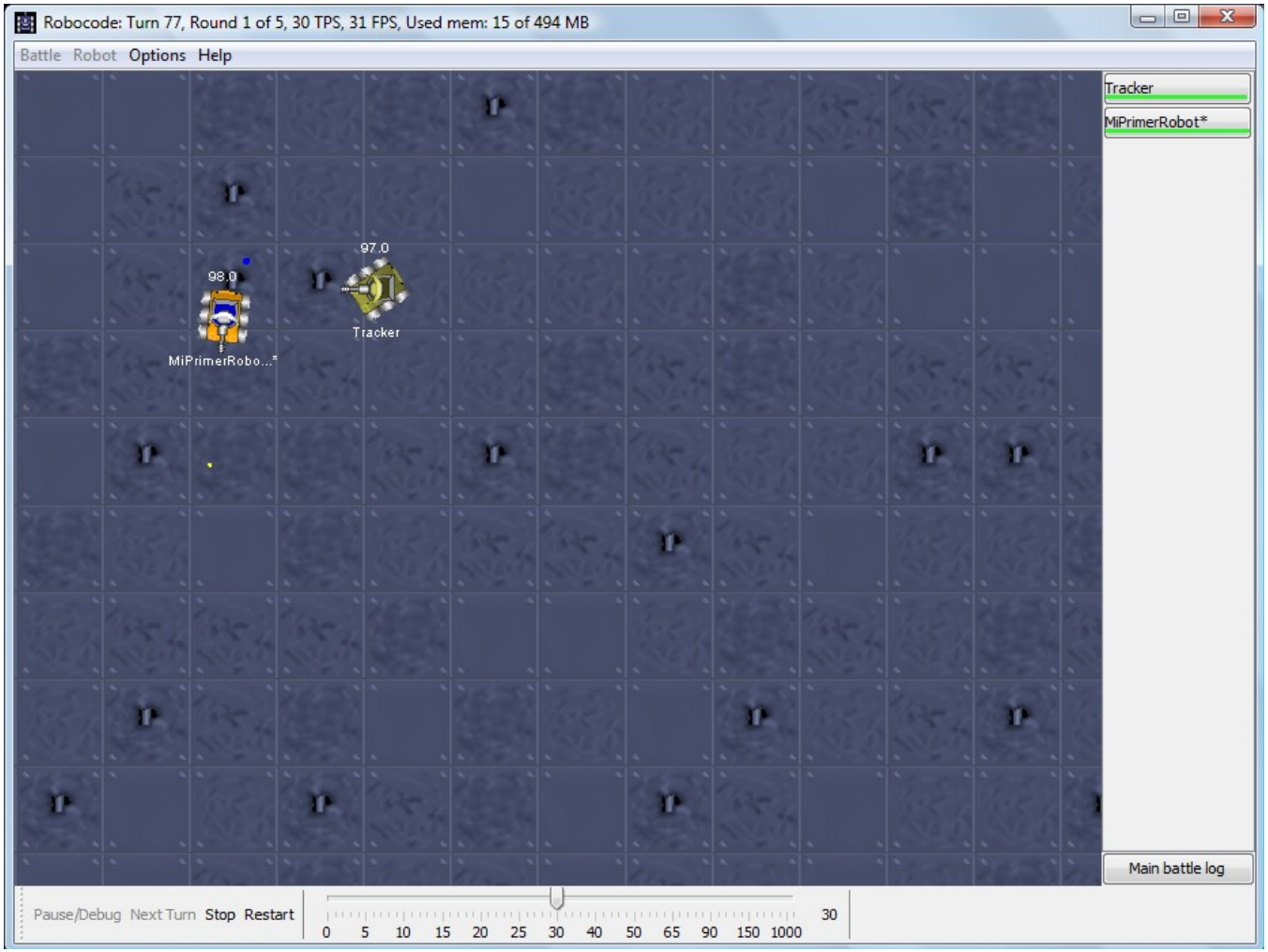


Código generado

- Panel disponible que muestra la estructura de bloques traducida a código Java
- La opción “Compilar y Ejecutar” pone a nuestro robot en el campo de batalla

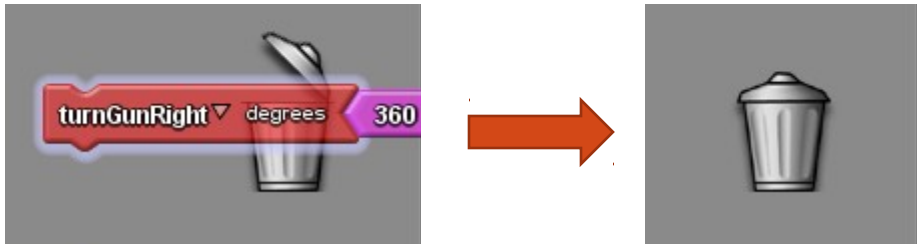


Campo de Batalla



Complementos

- Minimapa: Visualizador de todo el código, permite acceso directo a un sector en particular
- Tacho
- Posee además una serie de alertas (para bloques y código)



Ventajas

- El alumno **se concentra en el problema** y no en el lenguaje en particular
- El alumno siempre tiene la **opción de ver el código java** generado
- El alumno no necesita saber que existe un compilador, ni un .class, ni que es una JVM
- El alumno puede ver su idea en funcionamiento inmediatamente

Desventajas

- Actualmente implementa un subconjunto de Java
- R.I.T.A. no está pensado para enseñar diseño orientado a objetos
- No traduce desde el código fuente a bloques