Robocode(用游戏来学习Java技术还是用Java来玩游戏？)  
用你的JAVA编程技术来玩游戏吧！不会JAVA？那就用游戏来学习JAVA吧！  
什么是Robocode?

其实我对机器人一直很感兴趣。我记得在我还是初中的时候，就知道 AplleⅡ上有一个程序，用它来编写简单的机器人程序，然后相互作战。当时自己还完全不懂编程，总是向往着，那神秘的编程高手玩的游戏是怎样的？

Robocode就是这样一个东西，但是更好一些。它是一个基于Java语言的机器人作战游戏。 其代码的编写和建模都不错，玩起来也很有趣。Robocode是很多”编程游戏”软件中的一个，他们共同的特征是在没有用户输入的状态下许多机器人在一个及竞技场中比赛，用户必须编制一个高效的机器人来取胜。Robocode特别的像一场机器人坦克的大混战，它们互相开火直到只剩一个胜利者。程序完全是由JAVA编写，并且玩家必须要创造一个继承自Robot类的类。

你希望在玩游戏的过程中、在闪躲炮弹、执行精确攻击的演练中学会Java编程的 继承、多态性、事件处理以及内部类这些内容吗？Robocode 这个游戏为全世界的 Java 开发者实现这个愿望，它把游戏风潮变成了教学工具，人们对它的上瘾程度令人吃惊。下面，我参考网友 Sing Li 以前写的文章，让我们一起来拆解 Robocode，同时着手建造属于自己的、定制的、小而精悍的战斗机器吧！

Robocode 是一个很容易使用的机器人战斗仿真器，可以在所有支持 Java 2 的平台上运行。您创建一个机器人，把它放到战场上，然后让它同其他开发者们创建的机器人对手拼死战斗到底。Robocode 里有一些预先做好的机器人对手让你入门，但一旦您不再需要它们，就可以把您自己创建的机器人加入到正在世界范围内形成的某个联盟里去和世界最强手对阵。

每个 Robocode 参加者都要利用 Java 语言元素创建他或她的机器人，这样就使从初学者到高级黑客的广大开发者都可以参与这一娱乐活动。初级的 Java 的开发者们可以学习一些基础知识：调用 API 代码、阅读 Javadoc、继承、内部类、事件处理等等。高级开发者们可以在构建“最优品种”的软件机器人全球竞赛中提高他们的编程技巧。在本文中，我们将介绍 Robocode，并指导您从构建您平生第一个 Robocode 机器人开始征服世界。我们还将看一下迷人的“后台”机制，正是它使得 Robocode 起作用。

首先当然是下载和安装 Robocode 啦

Robocode 是 Mathew Nelson 的智慧之作，他是 IBM Internet 部门 Advanced Technology 的软件工程师。现在Robocode的主页已经搬迁到sourceforge这个开源网站上了，大家可以在这里下载RobotCode的最新版<http://robocode.sourceforge.net/> 到3月21日为止最新版本是1.0.7，大小为3.2M。  
好了，下载回来后当然还要在你的电脑上安装JAVA运行库才行的哦～地址是<http://java.sun.com/getjava>  
1.先安装好JAVA运行库，好像需要重启的？忘记了……  
2.把下载回来的robocode-setup.jar复制到c盘根目录  
3.打开 开始菜单 的“运行”，输入 java -jar “c:\robocode-setup.jar” 进行安装  
4.安装完后就可以在开始菜单中找到Robocode的菜单了，来～我们进入战场咯！

安装完成后，您也可以通过 shell 脚本（robocode.sh）、批处理文件（robocode.bat）或桌面上的图标来启动 Robocode 系统。此时，战场将会出现。在此，您可以通过菜单调用 Robot Editor 和 compiler。

Robocode 系统组件  
当您启动 Robocode 时，将看到两个GUI窗口，这两个窗口构成了 Robocode 的 IDE：

图 1. Robocode IDE

战场是机器人之间进行战斗直至分出胜负的场地。主要的仿真引擎被置于其中，并且允许您在这里创建战斗、保存战斗以及打开新建的或现有的战斗。通过界面区域内的控件，您可以暂停或继续战斗、终止战斗、消灭任何机器人个体或获取任何机器人的统计数据。此外，您可以在此屏幕上激活 Robot Editor。

Robot Editor 是一个定制的文本编辑器，它可以用于编辑生成机器人的 Java 源文件。在它的菜单里集成了 Java 编译器（用于编译机器人代码）以及定制的 Robot 打包器。由 Robot Editor 创建并成功编译的所有机器人都会处于战场上一个部署就绪的位置。

Robocode 里的每个机器人都由一个或多个 Java 类构成。这些类可以被压缩成一个 JAR 包。为此，Robocode 的最新版本提供了一个可以在战场 GUI 窗口中激活的“Robot Packager”。

对 Robocode 机器人的详细分析  
在写这篇文章时，Robocode 机器人是一个图形化的坦克。图 2 是一个典型的 Robocode 机器人的图解。

图 2. 对 Robocode 机器人的详细分析

请注意，机器人有一门可以旋转的炮，炮上面的雷达也是可以旋转的。机器人坦克车（Vehicle）、炮（Gun）以及雷达（Radar）都可以单独旋转，也就是说，在任何时刻，机器人坦克车、炮以及雷达都可以转向不同的方向。缺省情况下，这些方向是一致的，都指向坦克车运动的方向。

我们先不考虑怎么编程来实现机器人战斗，我们先用自带的例子机器人来一场战斗吧

单击菜单上的Battle，然后选New，出现了New Battle对话框

图 3. New Battle 对话框

左边的框是Packages，相当于一个文件夹，里面包含多个Robots（机器人）  
我们选择sample这个包，里面有Corners、Crazy、Fire等等很多例子的机器人了  
随便选择几个你喜欢的，然后按Add添加到Selected Robots框，进了这个框就是准备要上战场的机器人了～选择好后，按 StartBattle 开战吧！

现在你已经知道怎样可以使用机器人去战斗并且也构建好你的战场了，好，下面我们学习怎样来编写属于自己的战斗机器人！！

战场是机器人之间进行战斗直至分出胜负的场地。主要的仿真引擎被置于其中，并且允许在这里创建战斗、保存战斗以及打开新建的或现有的战斗。通过界面区域内的控件，可以暂停或继续战斗、终止战斗、消灭任何机器人个体或获取任何机器人的统计数据。此外，我们可以在此屏幕上的Robot菜单打开 Editor，就是我们机器人的代码编辑器了！Robot Editor 是一个定制的文本编辑器，它可以用于编辑生成机器人的 Java 源文件。在它的菜单里集成了 Java 编译器（用于编译机器人代码）以及定制的 Robot 打包器。由 Robot Editor 创建并成功编译的所有机器人都会处于战场上一个部署就绪的位置。我们就是要在这里编写机器人了。  
选择“File”》“New”》“Robot”来新建一个机器人。它会首先要你输入这个机器人的名字（注意名字首字母要大写哦），然后要你输入包的名字（就是保存这个机器人的文件夹名称），这样就生成了一个蠢蠢的机器人XForce的代码了～因为我们还没替它加上人工智能，呵呵！

现在单击菜单的Complie下的Complie进行编译，保存好，我们的机器人已经生产出来咯～  
现在关闭Editor，在进入New Battle，Pakeage下选择你刚才的包的名字，Robot下就有了我们新建的XForce机器人了～添加进去吧，然后选择多几个其他的机器人，开始战斗！

看～我们的XForce在战斗了！

是否觉得它太蠢了点呢？来，继续来学习～～

Robocode 机器人是一个图形化的坦克，请注意，机器人有一门可以旋转的炮，炮上面的雷达也是可以旋转的。机器人坦克车（Vehicle）、炮（Gun）以及雷达（Radar）都可以单独旋转，也就是说，在任何时刻，机器人坦克车、炮以及雷达都可以转向不同的方向。缺省情况下，这些方向是一致的，都指向坦克车运动的方向。

附：Robot 命令  
　　Robocode 机器人的命令集都收录在 Robocode API Javadoc 中。这些命令都是 robocode.Robot 类的公共方法。

（1）移动机器人、炮和雷达

　　移动机器人及其装备的基本命令：

* turnRight(double degree) 和 turnLeft(double degree) 使机器人转过一个指定的角度。
* ahead(double distance) 和 back(double distance) 使机器人移动指定的像素点距离；这两个方法在机器人碰到墙或另外一个机器人时即告完成。
* turnGunRight(double degree) 和 turnGunLeft(double degree) 使炮可以独立于坦克车的方向转动。
* turnRadarRight(double degree) 和 turnRadarLeft(double degree) 使炮上面的雷达转动，转动的方向也独立于炮的方向（以及坦克车的方向）。

　　这些命令都是在执行完毕后才把控制权交还给程序。此外，转动坦克车的时候，除非通过调用下列方法分别指明炮（和雷达）的方向，否则炮（和雷达）的指向也将移动。

* setAdjustGunForRobotTurn(boolean flag)：如果 flag 被设置成 true，那么坦克车转动时，炮保持原来的方向。
* setAdjustRadarForRobotTurn(boolean flag)：如果 flag 被设置成 true，那么坦克车（和炮）转动时，雷达会保持原来的方向。
* setAdjustRadarForGunTurn(boolean flag)：如果 flag 被设置成 true，那么炮转动时，雷达会保持原来的方向。而且，它执行的动作如同调用了 setAdjustRadarForRobotTurn(true)。

（2）获取关于机器人的信息

* getX() 和 getY() 可以捕捉到机器人当前的坐标。
* getHeading()、getGunHeading() 和 getRadarHeading() 可以得出坦克车、炮或雷达当前的方向，该方向是以角度表示的。
* getBattleFieldWidth() 和 getBattleFieldHeight() 可以得到当前这一回合的战场尺寸。

（3）射击命令

　　一旦掌握了移动机器人以及相关的武器装备的方法，我们就该考虑射击和控制损害的任务了。每个机器人在开始时都有一个缺省的“能量级别”，当它的能量级别减小到零的时候，我们就认为这个机器人已经被消灭了。射击的时候，机器人最多可以用掉三个能量单位。提供给炮弹的能量越多，对目标机器人所造成的损害也就越大。 fire(double power) 和 fireBullet(double power) 用来发射指定能量（火力）的炮弹。调用的 fireBullet() 版本返回 robocode.Bullet 对象的一个引用，该引用可以用于高级机器人。（也就是说，当你确定能击中对方，火力越大越好咯^\_^）

（4）事件

　　每当机器人在移动或转动时，雷达一直处于激活状态，如果雷达检测到有机器人在它的范围内，就会触发一个事件。作为机器人创建者，我们有权选择处理可能在战斗中发生的各类事件。基本的 Robot 类中包括了所有这些事件的缺省处理程序。但是，我们可以覆盖其中任何一个“什么也不做的”缺省处理程序，然后实现自己的定制行为。下面是一些较为常用的事件：

* ScannedRobotEvent。通过覆盖 onScannedRobot() 方法来处理 ScannedRobotEvent；当雷达检测到机器人时，就调用该方法。
* HitByBulletEvent。通过覆盖 onHitByBullet() 方法来处理 HitByBulletEvent；当机器人被炮弹击中时，就调用该方法。
* HitRobotEvent。通过覆盖 onHitRobot() 方法来处理 HitRobotEvent；当您的机器人击中另外一个机器人时，就调用该方法。
* HitWallEvent。通过覆盖 onHitWall() 方法来处理 HitWallEvent；当您的机器人撞到墙时，就调用该方法。

很多研究Robocode的 玩家都被其中的方向及坐标弄糊涂了。整个屏幕哪个是0度角，整个是坐标原点呢？ 顺时针与逆时针的方向如何区分？

一段英文的翻译及说明：

* heading – absolute angle in degrees with 0 facing up the screen, positive clockwise. 0 <= heading < 360.
* bearing – relative angle to some object from your robots heading, positive clockwise. -180 < bearing <= 180
* heading:是机器人方向与屏幕正上方的角度差,方向在0到360之间.
* bearing：是机器人的某个部件如雷达发现的目标与方向的角度差，顺时针为正角度在-180到180之间

几个在Robocode中很重要的概念：

* 坐标系：Robocode整个坐标系都是战场屏幕以左下角为原点
* 绝对方向系：Robocode中不管机器人在哪个方向都是以静态战场屏幕为参照的绝对角度（也即大家说的Heading）,正上方为0度角。也即不管是Robot,Gun,Radar向北为0，向东为90，向南为180，向西为270。
* 相对方向系：相对方向是Robot,Gun,Radar以机器人的动态heading角度为参照的角度差不再以整个静态屏幕为参照了，叫它相对因为机器人的heading是随着机器人移动而不停的在改变，heaing只是个相对物体。
* 顺时针和逆时针是看另一机器人是在你的Heading角度的(0,180)还是（-180,0）之间。

　　再次提醒：Heading是个静态角度，正上方总为0.不管是取Heading，还是取方向。Bearing是个角度差值，是由参照的Heading和发现时的Heading的差值。方向的问题就说到这，欢迎大家讨论。

我看了Robocode的基础知识，自己写了个bot，放到BattleField上却是屡战屡败……伤心ing。

　　Bot对于周围环境的了解非常有限。它可以知道其它机器人的距离、方位、方向、速度和能量等级。但是，它看不到子弹。怎么才可以有效的躲避对方的子弹呢？

　　Bot虽然看不到子弹，但是对方的能量等级还是可以scan到了。对方只要发射子弹就会耗损能量，并且耗损的能量介于0和3之间。根据这些线索，如何发现其它机器人正向它开炮对于“笨笨”的Bot不就易如反掌了？ ^\_^

　　当Bot检测到对方发射子弹的信息时，向左或向右移动一小步，嘿嘿，子弹就打不到咯～并且大多数Bot的瞄准方法是要么直接向目标开炮，要么试着根据Bot的速度和方向来推算位置。如果我的Bot不移动，两种算法都会正好冲着这个Bot的当前位置开炮。哈哈哈，这时我的Bot再移动，不就全部都打不到啦。（是不是颇有武侠小说里以静制动的高手味道？^\_^）

　　下面是部分代码和注释：

double previousEnergy = 100; //初始状态对方能量为100  
int movementDirection = 1; //移动方向  
int gunDirection = 1; //炮管方向

/\*\*  
\* 当检测到对方Bot，触发事件  
\* @param e  
\*/  
public void onScannedRobot(ScannedRobotEvent e) {  
//调整自己和对方之间的角度  
setTurnRight(e.getBearing()+90-30\*movementDirection);

//如果对方的能量损耗一定值，进行躲避动作  
double changeInEnergy = previousEnergy – e.getEnergy();  
if (changeInEnergy>0 && changeInEnergy<=3) {  
//躲避!  
movementDirection = -movementDirection; //和上次的躲避方向相反  
setAhead((e.getDistance()/4+25)\*movementDirection);  
}  
//将炮管指向对方当前位置  
gunDirection = -gunDirection;  
setTurnGunRight(99999\*gunDirection);

//射击  
fire(1);

//重新设置对方能量  
previousEnergy = e.getEnergy();  
}

　　是不是很简单？这个技巧还存在问题。子弹一发射，我的Bot就移动，所以它最终可能会移回炮弹轨迹之内。最好是在估计子弹要到达时再移动。

　　我有个更大胆的假设：因为现在我的Bot命中率还不高，那么如果我的Bot一直不开火，只是躲避对方的子弹的话，能不能拖到对方的能量为0呢？确实存在一点问题。对方子弹一发射，我的Bot就移动，并且这个移动是规律的来回移动。如果移动距离短了，就可能在回来的时候撞到对方的子弹；如果移动距离长了，就等于做一个直线运动，对方很容易计算得到Bot的运动轨迹。还有一个问题，躲避的时候很有可能撞到墙上……（撞墙是要减energy的:~(）

　　针对以上的问题，我另写了一个Bot。代码如下：

import robocode.\*;

public class HanicBot extends AdvancedRobot{  
private double eDist; //对方的距离  
private double move; //移动的距离  
private double radarMove = 45; //雷达移动的角度  
private double dFirePower; //火力

/\*\*  
\* main func run()  
\*/  
public void run() {  
eDist = 300;  
while(true){  
//每过一个周期，运动随机的距离  
double period = 4\*((int)(eDist/80)); //周期；敌人越接近，周期越短，移动越频繁  
//周期开始，则移动  
if(getTime()%period == 0){  
move = (Math.random()\*2-1)\*(period\*8 – 25);  
setAhead(move + ((move >= 0) ? 25: -25));  
}  
//避免撞墙  
double heading = getHeadingRadians(); //取得bot方向的弧度数  
double x = getX() + move\*Math.sin(heading); //移动move后将要达到的x坐标  
double y = getY() + move\*Math.cos(heading); //移动move后将要达到的y坐标  
double dWidth = getBattleFieldWidth(); //战场的宽度  
double dHeight = getBattleFieldHeight(); //战场的长度  
//当(x,y)超过指定的范围，则反向移动move  
if(x < 30 || x > dWidth-30 || y < 30 || y > dHeight-30){  
setBack(move);  
}  
turnRadarLeft(radarMove); //转动雷达  
}  
}//end run()

/\*\*  
\* 当检测到对方Bot，触发事件  
\* @param e  
\*/  
public void onScannedRobot(ScannedRobotEvent e) {  
eDist = e.getDistance(); //取得对方距离  
radarMove = -radarMove; //设置雷达  
double eBearing = e.getBearingRadians(); //取得和对方相对角度的弧度数  
//将bot转动相对的角度，以后bot的运动将是以对方为圆心的圆周运动  
setTurnLeftRadians(Math.PI/2 – eBearing);  
//转动炮管指向对方  
setTurnGunRightRadians(robocode.util.Utils.normalRelativeAngle(  
getHeadingRadians() + eBearing – getGunHeadingRadians()));  
//根据对方距离射击  
dFirePower = 400/eDist;  
if (dFirePower > 3){  
dFirePower = 3;  
}  
fire(dFirePower);  
}  
}

　　首先，为了迷惑对方，不让对方容易的得到Bot的移动规律，Bot就要在一定的时间内做出随机的运动，这个很容易办到。并且，我给Bot的运动改变时间规定了周期。这个周期随离对方的距离改变，敌人越接近，周期越短，移动越频繁。

double period = 4\*((int)(eDist/80));  
if(getTime()%period == 0){  
move = (Math.random()\*2-1)\*(period\*8 – 25);  
setAhead(move + ((move >= 0) ? 25: -25));  
}

　　其次，Bot的运动不是呈直线的。而是以对方为圆心的圆周运动。

setTurnGunRightRadians(robocode.util.Utils.normalRelativeAngle(  
getHeadingRadians() + eBearing – getGunHeadingRadians()));

　　最后是如何避免撞墙。这里要用到点三角函数-\_-!! 原理就是，计算Bot一次运动后将要达到的坐标是不是位于规定的危险区域。如果是，则立即反方向运动。

double heading = getHeadingRadians();  
double x = getX() + move\*Math.sin(heading);  
double y = getY() + move\*Math.cos(heading);  
double dWidth = getBattleFieldWidth();  
double dHeight = getBattleFieldHeight();  
if(x < 30 || x > dWidth-30 || y < 30 || y > dHeight-30){  
setBack(move);  
}

这个Bot的威力如何？呵呵，我去测试一下先～

好了，就说到这里了，欢迎各大高手来踩……

**关于其它的一些”编程游戏”**   
有许多软件是基于这种思想的，Robocode它自己就是来源于机器人大战Robot Battle(<http://www.robotbattle.com/>)这款软件。其它的编程游戏还包括：   
· AI Fleet Commander   
· AI Wars   
· AT-Robots   
· Bolo   
· BotWarz   
· C-Robots   
· Cadaver   
· CodedWombat   
· Colobot   
· Corewars   
· CybWar   
· GRobots   
· DroidBattles   
· Karel the Robot   
· Mindrover   
· IntelliBots   
· Omega   
· RealTimeBattle   
· Robot Wars   
· RoboWar   
· SRobots   
· VBRobots   
就我所看过的”编程游戏”，Robocode是最简单上手的。

· 它非常容易上手，是特别为教学而设计的   
· 它具有平滑且吸引人的图形   
· 它完全地将编辑器，编译器和运行环境集成在了一起。   
· 它是由JAVA编写的，且JAVA非常适合当作初学语言