Análise de Heurísticas de um Robocode Defensivo

Júlio Zinga S. Lopes
Thiago Gondim Ribeiro

Departamento de Sistemas e Computação – Universidade Federal de Campina Grande (UFCG) – Campina Grande – PB – Brasil – Disciplina: Inteligência Artificial: Professora: Joseana Macedo Fechine – Período 2007.1
{juliozinga, thiago.gondim} @gmail.com

Abstract *This paper describes the studying and implementation of heuristics to Robocode game. The project points out desired behaviors by miniature tank robots so they could achieve good performance in combats, proposing heuristics which these behaviors can be achieved through. The implementation was developed in Java language, using part of the structure provided by the Robocode project itself*

Resumo. *Este artigo descreve o estudo e a implementação de heurísticas para o jogo Robocode. O projeto aponta comportamentos desejáveis a um robô para que este tenha um bom desempenho nos combates, e propõe heurísticas que traduzam este comportamento. A implementação foi feita em Java, utilizando parte da estrutura fornecida pelo próprio Robocode.*

1. INTRODUÇÃO

Robocode é um jogo educacional de código aberto muito útil para o aprendizado da linguagem Java, Orientação a Objeto e Inteligência Artificial. O objetivo do corrente projeto foi encontrar heurísticas que tornem um robô inteligente o suficiente para sobreviver o máximo às batalhas e alcançar uma boa pontuação. O jogo é composto por robôs, que movem-se em um campo de batalha, e atacam-se através de seus canhões. Possuem percepção de sua própria posição em relação ao campo de batalha e através de um radar conseguem identificar demais robôs. Vence quem fizer mais pontos atingindo os inimigos.

Ainda como objetivo do projeto, o mesmo visou definir uma série de parâmetros – tais como movimentação, análise do campo de batalha, interação com os demais robôs e decisão de quando atacar ou manter-se recuado – de um comportamento que possibilite a um robô ter um sucesso satisfatório nos combates.

Este documento organiza-se sob a seguinte disposição: Introdução; Referencial teórico; Metodologia adotada; Resultados obtidos; Análise dos Resultados; Conclusões/Considerações Finais

2. REFERENCIAL TEÓRICO

2.1. Robocode

O Robocode trata-se de um jogo educacional de código aberto inicialmente desenvolvido pela IBM , tendo como idealizador e responsável Mathew Nelson. Atualmente encontra-se hospedado no SourceForge e recebe contribuições de várias pessoas de todo o Mundo.

O jogo tem como um de seus intuitos auxiliar pessoas no aprendizado da linguagem Java e Inteligência Artificial, através, entre outras coisas, do envolvimento que jogos virtuais despertam em pessoas adeptas a esta prática. Nele, competidores implementam tanques em miniatura que duelam com outros tanques identicamente estruturados, porém diferentemente programados. O duelo acontece em um campo de batalha. Robôs movem-se, atacam, batem na parede e reconhecem o inimigo por radar.

Embora a idéia deste jogo possa parecer simples, bons robôs podem ter milhares de linhas de código dedicadas às estratégias.

2.2. Estudo Comportamental

Para o desenvolvimento de heurísticas e, para que possibilite um bom rendimento ao robô, foi feito um estudo de comportamentos desejáveis, incluindo: movimentação, análise do campo de batalha, interação com os demais robôs e decisão de quando atacar ou manter-se recuado.

2.2.1. Movimentos

Um robô deve possuir uma boa movimentação, pois isso dificulta que outro robô o atinja com tiros. Uma movimentação retilínia e contínua é fácil de ser percebida e prevista por adversários. Assim sendo, é importante que a movimentação seja em formato de curvas.

Quanto mais imprevisível e inconstante for o trajeto do robô, mais protegido de tiros alheios estará. No jogo, não é possível detectar tiros que ainda não atingiram o robô, por isso quanto mais o robô se movimentar, mesmo sem identificar se algum tiro foi disparado, menor a probabilidade de ser atingido.

2.2.1. Detecção de Paredes

Observa-se que as colisões em paredes são prejudiciais porque diminuem a energia do robô, como forma de punição. Por isso um comportamento desejável seria a detecção da proximidade de uma das paredes e o reajuste do trajeto para evitar a colisão. Outro motivo para se evitar colisões em paredes é o fato de o robô perder velocidade ou parar a cada batida, tornando-o um alvo vulnerável.

2.2.1. Interação com outros robôs

No Campo de Batalha, um robô precisa saber lidar com ações dos outros robôs (por exemplo, quando for atingido por uma bala ou se chocar com outro robô). Existe ainda a possibilidade de identificar a quantidade de robôs e agir de acordo com esta informação.

Ao ser atingido por um tiro, é possível identificar a direção de origem deste tiro e ajustar um novo trajeto para o robô ou apenas apontar o canhão para a origem do disparo. É

importante sair do local que recebeu o tiro, pois evita que outros tiros com o mesmo destino o atinjam.

Em relação ao número de oponentes, o Robocode oferece a possibilidade de termos a precisão da quantidade de oponentes presentes em qualquer momento do jogo. Foi possível observar o aumento das chances de ser atingido em virtude do aumento de robôs no campo de Batalha. Quanto mais robôs, mais ágil e irregular deve ser o trajeto, mesmo não diminuindo as chances de ficar preso entre os robôs. Em virtude disso, enquanto houverem mais de três robôs disputando, o robô deve permanecer afastado do centro de concentração, fazendo ataques de longe. É preciso saber quando atacar de perto e quando distanciar-se.

3. METODOLOGIA

O processo de desenvolvimento adotado deu-se através de encontros noturnos na casa de um dos integrantes. Onde discutia-se como melhor abordar o problema, trocando idéias sobre os comportamentos percebidos, ações e heurísticas e, onde testávamos estas idéias na prática, implementando e testando as mesmas contra outros robôs baixados pela internet. Assim, o desenvolvimento deu-se de forma centralizada, mesmo quando os integrantes não estavam presencialmente reunidos, utilizando-se mensageiro instantâneo e correio eletrônico para quaisquer pontos relevantes.

O único fator que pôde dificultar foi o fato de a dupla não ter tido muito tempo, tanto por demorarem na definição do que fariam quanto pela disponibilidade dos membros. No entanto, a dupla desenvolvedora conseguiu superar estas dificuldades e compensar o tempo ao trabalharem, também, durante as madrugadas e finais-de-semana.

4. RESULTADOS OBTIDOS

4.1. Heurísticas

A partir do estudo do comportamento desejável, foi desenvolvido algumas heurísticas que visam atingir o comportamento desejável.

4.1.1. Heurísticas de Percurso

Para se fazer um percurso difícil de ser previsto e que esteja sempre propício a desviar de muitas balas, foi implementado um deslocamento com contornos de 80, 180 e 360 graus, variando nos sentidos direita e esquerda.

Para cada batida na parede foi desenvolvido um método que verifica qual o melhor movimento a ser realizado pelo robô para se afastar da parede. Este método é chamado quando o robô bate na parede, como também tenta prever uma batida.

Figura 4.1 Mudança de trajetória após batida em uma parede

Listagem 4.1: verifica se o robô está perto de uma das paredes, caso seja positivo, chama o método para se afastar

```
verificaposição().
...
if (EstaPertoDaParede(getX(), getBattleFieldWidth())) {
        afastaDaParede("horizontal");
}
// verifica se MyRobot está perto de uma parede "superior" ou "inferior"
if (EstaPertoDaParede(getY(), getBattleFieldHeight())) {
        afastaDaParede("vertical");
```

O método verificaposição() analisa se o robô está a menos de 180 pixels de alguma parede. Caso isto se confirme, é chamado o método afastaDaParede(), que reorganiza a rota do robô em virtude do seu ângulo de inclinação e da posição da parede.

Listagem 4.2: define o novo trajeto para afastar o robô de uma parede, de acordo com a inclinação do robô

```
afastaDaParede()
        if (direcao.startsWith("h")) {
        if (getX() > getBattleFieldWidth() / 2) {
        if (heading > 0 && heading < 90) {
            setTurnLeft(45 + heading);

        } else if (heading > 90 && heading < 180) {
            setTurnRight(45 + heading);

        }
        if (heading > 0 && heading < 180) {
            setBack(DISTANCIA_DE_FUGA);
        } else
setAhead(DISTANCIA_DE_FUGA);
```

Como é possível observar no início do método, é verificado a posição do robô em relação ao campo de batalha, se o robô localiza-se na parte direita ou esquerda. Em seguida, é verificada a inclinação do robô, para só então indicar a rotação necessária para que o robô se afaste da parede.

4.1.2 Heurísticas de Interação com outros Robôs

Foram elaboradas três heurísticas que determinam o comportamento do robô em relação à interação com os demais robôs. Estas abordam a reação a batidas entre robôs, batidas por um tiro e comportamentos diferentes de acordo com a quantidade de robôs.

O método onHitRobot é acionado cada vez que uma batida entre robôs acontece. Este verifica se o robô oponente está no alvo. Caso estiver, atira com potência máxima.

Listagem 4.3: Se o oponente estiver na frente do canhão, atire nele

```
if (bearingFromGun > -10 && bearingFromGun < 10) {
        fire(3);
}
```

A reação do robô a um tiro é tratada pelo método onHitByBullet. Para atender ao comportamento desejado, ao ser atingido, o robô primeiramente identifica se o tiro é oriundo da parte dianteira ou traseira, e assim irá seguir na direção contrária. Caso o tiro tenha vindo da direita, o robô rotaciona para a direita, dependendo do ângulo de incidência do tiro, como é possível observar abaixo:

Listagem 4.4: Se o oponente estiver na frente do canhão, atire nele

```
if (inimigoBearing > 160 && inimigoBearing < -160) {
        setAhead(Double.POSITIVE_INFINITY);
        return;
}

if (inimigoBearing > 0) { // verifica se o inimigo está à direita
                setTurnRight(120 - event.getBearing());
}
```

4.1.2 Heurísticas de Ataque e Defesa

Cada vez que o radar localiza um robô, o número de robôs presentes no campo de batalha é verificado. Se existirem mais de três robôs no jogo, o robô se afasta, evitando assim o “fogo cruzado”.

Listagem 4.5: Parte do Método onScannedRobot: Se existirem mais que três oponentes, afasta-se do alvo que o radar detectar

```
if (getOthers() > 3 && !alvo.equals(alvoAntigo) ) {
        setTurnGunLeft(bearingFromGun);
        setBack(Double.POSITIVE_INFINITY);
        setTurnLeft(90);
        setTurnRight(90); }
```

Se o robô for atingido por um mesmo inimigo 3 vezes ou mais, uma função de reconhecimento é ativada no robo(isCritico()) e, o robô passa a atacar apenas este inimigo por um determinado período. Após este período, o robô volta a agir normalmente e a atacar qualquer robô que seja localizado pelo radar.

Listagem 4.6: Código que identifica um Inimigo Crítico, chamado quando um adversário atinge o robô mais de três vezes

```
if (inimigo.isCritico()) {
        alvoCritico = nomeInimigo;
        setTurnRight(e.getBearing());}
```

5. ANÁLISE DOS RESULTADOS

Durante a implementação das heurísticas apresentadas acima, foram capturados os resultados das batalhas. Apresentamos a seguir as telas capturadas.

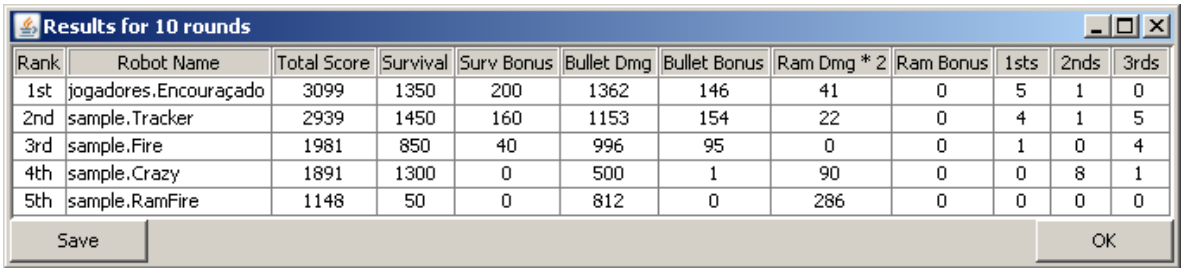
Results for 10 rounds

Rank	Robot Name	Total Score	Survival	Surv Bonus	Bullet Dmg	Bullet Bonus	Ram Dmg * 2	Ram Bonus	1sts	2nds	3rds
1st	jogadores.Encouraçado	3232	1300	200	1508	173	50	0	5	1	1
2nd	sample.Tracker	2537	1000	80	1231	159	68	0	2	1	3
3rd	sample.Crazy	2040	1450	80	418	12	80	0	2	5	3
4th	sample.RamFire	1813	600	0	827	33	252	101	0	3	1
5th	sample.Fire	1334	650	40	595	46	2	0	1	0	2

Save OK

Figura 5.1 Tabela de pontuação de robô, sem as heurísticas estudadas

A Figura 5.1 representa o *ranking* de 10 partidas seguidas entre 5 robôs. Este foi o primeiro quadro registrado. Nele estão implementados apenas as heurísticas de percurso e a identificação de alvos com um radar perseguidor. O robô principal é o de nome Encouraçado, em homenagem ao barco russo Encouraçado Potekin. Nesta tabela é possível observar que, embora o robô Encouraçado tenha uma ampla vantagem sobre o segundo colocado, observa-se que os demais tiveram um desempenho razoável.



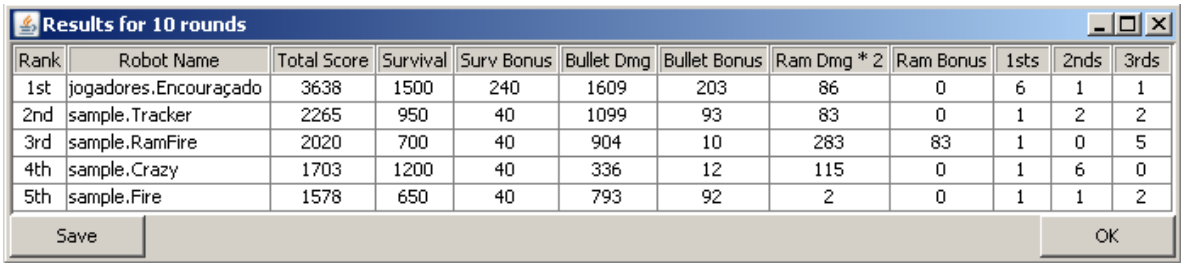
Results for 10 rounds

Rank	Robot Name	Total Score	Survival	Surv Bonus	Bullet Dmg	Bullet Bonus	Ram Dmg * 2	Ram Bonus	1sts	2nds	3rds
1st	jogadores.Encouraçado	3099	1350	200	1362	146	41	0	5	1	0
2nd	sample.Tracker	2939	1450	160	1153	154	22	0	4	1	5
3rd	sample.Fire	1981	850	40	996	95	0	0	1	0	4
4th	sample.Crazy	1891	1300	0	500	1	90	0	0	8	1
5th	sample.RamFire	1148	50	0	812	0	286	0	0	0	0

Save OK

Figura 5.2 Tabela de pontuação de robô, com a heurística de detectar paredes

Na figura 5.2 foi implementado a heurística de detectar paredes e saber delas se afastar. Observa-se uma diminuição do rendimento da pontuação do robô. Todavia, é possível observar que poucos foram os robôs que venceram as partidas: Encouraçado foi campeão 5 vezes e ficou em segundo, enquanto o *Tracker* foi primeiro 4 vezes, na tabela anterior esta distribuição foi mais eqüitativa.



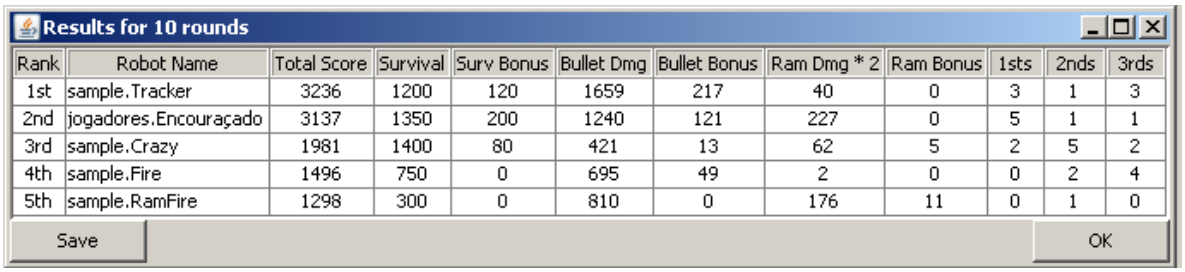
Results for 10 rounds

Rank	Robot Name	Total Score	Survival	Surv Bonus	Bullet Dmg	Bullet Bonus	Ram Dmg * 2	Ram Bonus	1sts	2nds	3rds
1st	jogadores.Encouraçado	3638	1500	240	1609	203	86	0	6	1	1
2nd	sample.Tracker	2265	950	40	1099	93	83	0	1	2	2
3rd	sample.RamFire	2020	700	40	904	10	283	83	1	0	5
4th	sample.Crazy	1703	1200	40	336	12	115	0	1	6	0
5th	sample.Fire	1578	650	40	793	92	2	0	1	1	2

Save OK

Figura 5.3 Tabela de pontuação de robô, com a heurística de reagir a um tiro recebido

Na tabela acima temos o resultado depois de implementado o método onHitBybullet, o qual identifica um tiro recebido e muda o trajeto do robô. Percebe-se melhoria na pontuação e aumento no número de vitórias em 1º lugar.



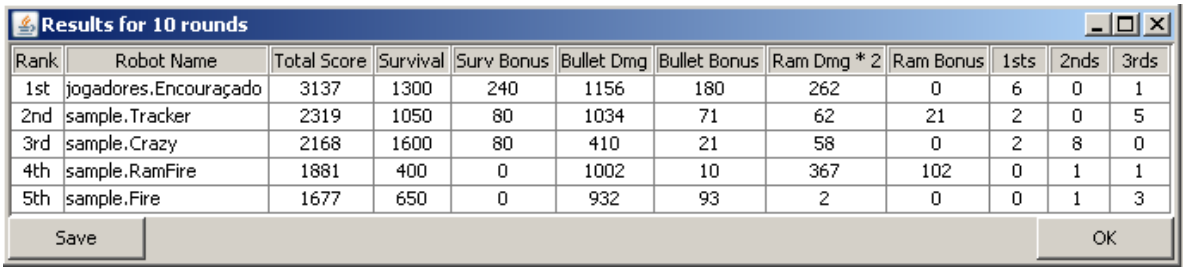
Results for 10 rounds

Rank	Robot Name	Total Score	Survival	Surv Bonus	Bullet Dmg	Bullet Bonus	Ram Dmg * 2	Ram Bonus	1sts	2nds	3rds
1st	sample.Tracker	3236	1200	120	1659	217	40	0	3	1	3
2nd	jogadores.Encouraçado	3137	1350	200	1240	121	227	0	5	1	1
3rd	sample.Crazy	1981	1400	80	421	13	62	5	2	5	2
4th	sample.Fire	1496	750	0	695	49	2	0	0	2	4
5th	sample.RamFire	1298	300	0	810	0	176	11	0	1	0

Save OK

Figura 5.4 Tabela de pontuação de robô, com a heurística de reagir a batida a outro robô, porém sem reagir aos tiros recebidos

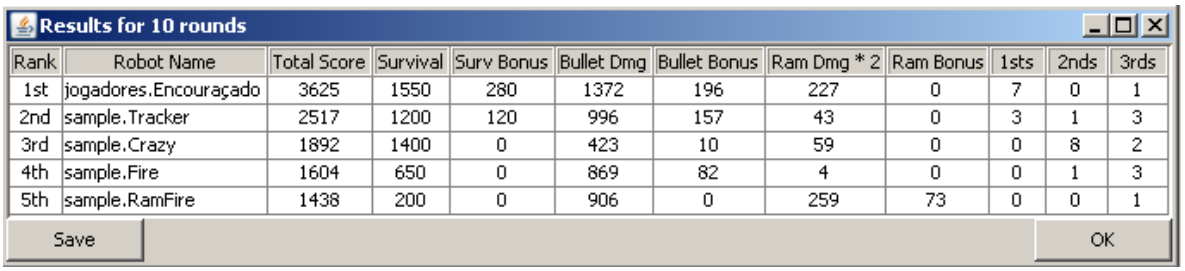
A tabela 5.4 é referente à implementação do método onHitRobot, porém sem a implementação do onHitByBullet. Nele vemos um grande decréscimo do rendimento da pontuação, embora o robô Encouraçado tenha vencido o maior número de partidas. Isto é justificável pelo fato de o robô estar mais exposto e não reagir quando recebe tiros.



Rank	Robot Name	Total Score	Survival	Surv Bonus	Bullet Dmg	Bullet Bonus	Ram Dmg * 2	Ram Bonus	1sts	2nds	3rds
1st	jogadores.Encouraçado	3137	1300	240	1156	180	262	0	6	0	1
2nd	sample.Tracker	2319	1050	80	1034	71	62	21	2	0	5
3rd	sample.Crazy	2168	1600	80	410	21	58	0	2	8	0
4th	sample.RamFire	1881	400	0	1002	10	367	102	0	1	1
5th	sample.Fire	1677	650	0	932	93	2	0	0	1	3

Figura 5.5 Tabela de pontuação de robô, com a heurística de reagir a batida a outro robô e quando a tiros recebidos

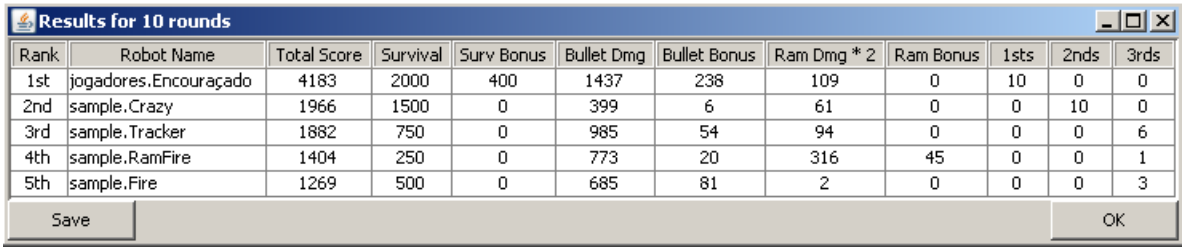
Na tabela 5.5, temos os dois métodos onHitRobot e onHitByBullet. Observa-se que o robô Encouraçado retoma a superioridade, inclusive com muitos bônus por ter atacado e destruído muitos robôs.



Rank	Robot Name	Total Score	Survival	Surv Bonus	Bullet Dmg	Bullet Bonus	Ram Dmg * 2	Ram Bonus	1sts	2nds	3rds
1st	jogadores.Encouraçado	3625	1550	280	1372	196	227	0	7	0	1
2nd	sample.Tracker	2517	1200	120	996	157	43	0	3	1	3
3rd	sample.Crazy	1892	1400	0	423	10	59	0	0	8	2
4th	sample.Fire	1604	650	0	869	82	4	0	0	1	3
5th	sample.RamFire	1438	200	0	906	0	259	73	0	0	1

Figura 5.6 Tabela de pontuação de robô, com a heurística de agir de acordo com a quantidade de robôs opositores

Nesta outra tabela, temos o robocode agindo de forma defensiva quando temos mais que 3 outros robôs competindo. O seu rendimento melhora ainda mais.



Rank	Robot Name	Total Score	Survival	Surv Bonus	Bullet Dmg	Bullet Bonus	Ram Dmg * 2	Ram Bonus	1sts	2nds	3rds
1st	jogadores.Encouraçado	4183	2000	400	1437	238	109	0	10	0	0
2nd	sample.Crazy	1966	1500	0	399	6	61	0	0	10	0
3rd	sample.Tracker	1882	750	0	985	54	94	0	0	0	6
4th	sample.RamFire	1404	250	0	773	20	316	45	0	0	1
5th	sample.Fire	1269	500	0	685	81	2	0	0	0	3

Figura 5.7 Tabela de pontuação de robô, com todas as heurística estudadas

Por final, temos a tabela do robô totalmente implementado. A ultima implementação foi a detecção de um inimigo crítico, aquele que acerta mais de 3 tiros no robocode. Observamos um desempenho excepcional, com 100% de aproveitamento nas vitórias.

6. CONCLUSÃO

Apesar do jogo Robocode fornecer facilidades no manuseio e compreensão da implementação do robô, construir um robô vencedor exige muita atenção e perspicácia. Os estudos desenvolvidos e as soluções propostas permitirão a futuros estudiosos do Robocode um bom início de partida.

Após a implementação verificou-se que o objetivo de evitar bater na parede foi o único que não teve êxito. Isto é justificado pelo fato de o trajeto do robô ser extremamente irregular, o que é positivo, uma vez que isso permite a ele desviar de "rajadas de tiros". O resultado conseguido foi muito satisfatório pois foi nítido o comportamento do robô como o planejado, além dos resultados das batalhas terem conseguido uma significativa melhoria.

Como sugestão para outras equipes, fica o desafio de explorar como implementar os comportamentos aqui indicados de formas diferentes e com outras abordagens. Ficou visível a possibilidade de se trabalhar com redes neurais, utilizando como parâmetros de entrada: velocidade inimiga, distância, energia prórpria, tempo decorrido. O Robocode oferece muitas variáveis para alimentar a rede.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] MACEDO, Joseana Fechine. (2007) "*Disciplina:Inteligência Artificial I*" On-line. Disponível em: http://www.dsc.ufcg.edu.br/~joseana/IA1-20071.html

[2] http://robocode.sourceforge.net/

[3] http://*www.wikipedia.org*