Universidade do Minho

Mestrado Integrado em Engenharia Informática - 4º ano

Sistemas Inteligentes

2016/2017

Sistemas Autónomos

Programação de Robôs - *Robocode*

Grupo 1

Carlos Abelha
David Moreira
Tiago Azevedo

6 de Junho de 2017

Conteúdo

1 Introdução

No âmbito da Unidade Curricular Sistemas Autónomos do perfil de Sistemas Inteligentes, foi proposto o desenvolvimento e programação de um sistemas autónomos, com particular ênfase na construção de processos de simulação de comportamentos individuais, de grupo e sociais, isto com recurso ao ambiente de programação *Robocode*.

1.1 *Robocode*

O ambiente de programação *Robocode* tem como principal objetivo e motivação a disponibilização de um cenário de simulação para o desenvolvimento de robôs virtuais num panorama competitivo e colaborativo.

Este ambiente de programação é de utilização fácil e consistente, para facultar conceitos básicos da linguagem de programação *JAVA*. Para além disto fornece uma abordagem bastante flexível e evoluída que permite a utilização de estratégias, que demonstram Inteligência Artificial, para desenvolver robôs individuais ou conjuntos de robôs (equipas), onde se pode encontrar técnicas de cooperação e processos de tomada de decisão.

1.2 Descrição do problema

Com este trabalho surge a necessidade de desenvolver um conjunto de robôs, neste caso 4 robôs individuais e uma equipa de 3 elementos (1 *Team Leader* e 2 *Droids*), sendo que as suas políticas e procedimentos de atuação sejam de escolha livre.

Cada robô individual necessita de evidenciar uma estratégia e arquitetura de controlo, assim como a equipa, sendo que esta precisa de ser capaz de comunicar entre si de forma a que seja possível a partilha de informação, e alcançar objetivos comuns.

1.3 Objetivos

Os objetivos principais para o desenvolvimento deste projeto, tanto para os robôs individuais, como para a equipa de robôs, são os seguintes:

- Construir robôs a partir de diferentes classes, *Robot*, *Droid* e *TeamRobot*(*AdvancedRobot*).
- Desenvolver sistemas de controlo implementando as diferentes estratégias de controlo, *Openloop*, *Feedforward* e *Feedbackward*.
- Optar por diferentes arquiteturas de controlo, *Reativas*, *Deliberativas* e *Híbridas*.
- Planeamento de trajetórias, desvio de obstáculos e arquiteturas de navegação.
- Desenvolver um robô com comportamento caracterizado pelos modelos emocionais, *PAD Space*, *OCC Model* e *OCEAN Model*.

2 Desenvolvimento

Ao longo deste capítulo irá ser descrito cada robô desenvolvido, passando a explicar cada fase de implementação dos mesmos, e decisões tomadas.

2.1 Robô *Reativo*

Este robô foi desenvolvido com recurso a uma arquitetura reativa e estratégias de controlo do tipo *open loop* e *feedfoward*.

A implementação de uma arquitetura reativa num robô faz com que não possua capacidade de planeamento, nem conhecimento do ambiente onde se encontra e não usufrua de formas de raciocínio complexas, comportando-se como um autómato inserido no meio ambiente onde se encontra.

Procura lidar com a perceção que tem do mundo através da receção de itens de informação que lhe são passadas através dos sensores à sua disposição. Todo seu comportamento pode ser visto como resultado de invocação de regras do tipo condição/ação, quando associadas a certas formas de comportamento.

As principais vantagens deste tipo de arquitetura é a rapidez da resposta e a fácil implementação.

As estratégias de controlo do tipo *open loop* e *feedfoward* foram utilizadas para o processo de tomada de decisão sobre a utilização de informação do ambiente para controlar o robô. Embora dependa dos sensores, estes não são utilizados para fornecer, nem armazenar informação do ambiente em que o robô está inserido.

Figura 1: Estratégia de Controlo Reativa

Para desenvolver este robô foram tomadas decisões e algumas implementações importantes, tais como:

- Controlo de intensidade de disparo, consoante a sua própria energia.
- *ScannedRobot*, é utilizado como sensor de movimento, apenas para detetar o inimigo e posteriormente atacar.
- *HitRobot*, para atacar o inimigo imediatamente após chocar com ele.
- *HitWall*, uma vez que não tem noção do ambiente envolvente, nem dos limites do campo de batalha, e aproveita para para mudar de direção.

- *HitByBullet*, quando é atingido direciona se para o inimigo que atacou, dispara e por fim desvia a trajetória
- *BulletHit*, assim que atinge um inimigo volta a disparar na mesma direção em que o disparo teve sucesso.

2.2 Robô *Deliberativo*

Este robô foi desenvolvido com recurso a uma arquitetura deliberativa e estratégia de controlo do tipo *feedfoward*. A implementação de uma arquitetura deliberativa permite ao robô a capacidade de imitar o processo de planeamento e tomada de decisão para a execução de uma tarefa. Ou seja o robô utiliza o conhecimento que possui sobre o ambiente onde esta inserido para planear as suas ações durante a execução.

As principais vantagens deste tipo de arquitetura é a possibilidade de executar ações de elevado nível, visto que possui a capacidade de perceção do ambiente e planeamento de tarefas, e demonstra uma maior utilidade num ambiente estático/previsível.

A estratégia de controlo do tipo *feedfoward* foram utilizadas para o processo de tomada de decisão sobre a adoção de estratégias para a utilização de informação do ambiente para controlar o robô, sendo que os sensores são utilizados, apenas para receber informação do ambiente, sendo que esta informação é utilizada para atualizar as variáveis que modelam o ambiente.

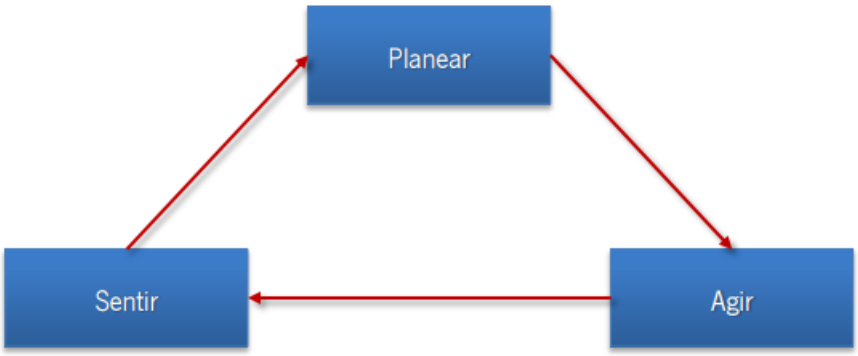

Figura 2: Estratégia de Controlo Deliberativa

Para desenvolver este robô foram tomadas decisões e algumas implementações importantes, tais como:

- Controlo de intensidade de disparo, consoante o número de inimigos, distância e energia dos mesmos.
- *ScannedRobot*, é utilizado reconhecer todo o ambiente e manter as variáveis de controlo atualizadas.
- *RobotDeath*, assim que um inimigo é abatido, a sua morte é registada.

2.3 Robô *Híbrido*

Este robô foi desenvolvido com recurso a uma arquitetura híbrida e estratégia de controlo do tipo *feedback*.

Uma arquitetura híbrida combina as abordagens deliberativa e reativa, ou seja, permite ao robô a capacidade de planeamento de ações a partir de informação proveniente do meio ambiente sendo também possível reagir de forma rápida a situações não planeadas.

A ideia principal passa por categorizar as funcionalidades do robô, isto é, uma maior prioridade onde é atribuída as funcionalidades reativas sobre as deliberativas, de modo a usufruir das características mais importantes, capacidade de reação mais rápida a eventos detetados no ambiente.

Este tipo de arquitetura apresenta significativos ganhos em termos de adaptabilidade, robustez e desempenho.

A estratégia de controlo do tipo *feedback* foram utilizadas para o processo de tomada de decisão sobre a utilização de informação do ambiente para controlar o robô, sendo a utilizados sensores para monitorizar constantemente o ambiente, ajustando as ações conforme a interpretação.

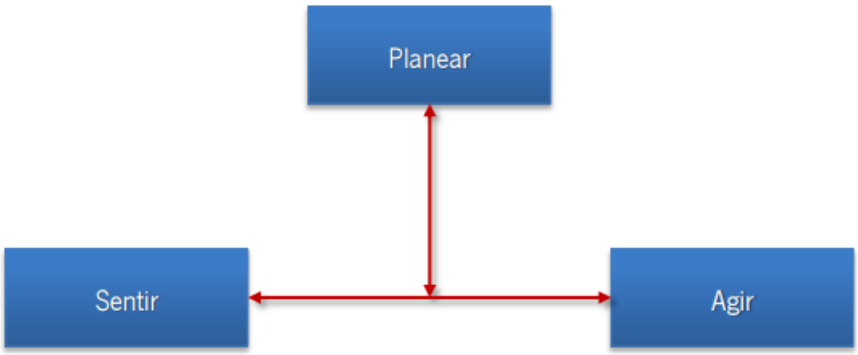

Figura 3: Estratégia de Controlo Híbrida

Para desenvolver este robô foram tomadas decisões e algumas implementações importantes, tais como:

- Controlo de intensidade de disparo, consoante o número de inimigos, distância, sua energia e dos inimigos.
- *ScannedRobot*, é utilizado reconhecer todo o ambiente e manter as variáveis de controlo atualizadas. É também utilizado para posteriormente atacar o inimigo, recorrendo a uma previsão linear do disparo.
- *HitRobot*, para atacar o inimigo imediatamente após chocar com ele.
- *HitWall*, uma vez que não tem noção do ambiente envolvente, nem dos limites do campo de batalha, e aproveita para para mudar de direção.

- *HitByBullet*, quando é atingido direciona se para o inimigo que atacou, dispara e por fim desvia a trajetória
- *BulletHit*, assim que atinge um inimigo volta a disparar na mesma direção em que o disparo teve sucesso.
- *RobotDeath*, assim que um inimigo é abatido, a sua morte é registada.

2.4 Robô *Emotivo*

Para este robô foi utilizado como base o robô *Reativo*, desenvolvido anteriormente. Além disso, este robô foi desenvolvido tendo em conta o *OCC Model* (*Ortony*, *Clore* e *Collins*), sendo um modelo que ficou estabelecido como modelo standard para a geração de estados emocionais em agentes.

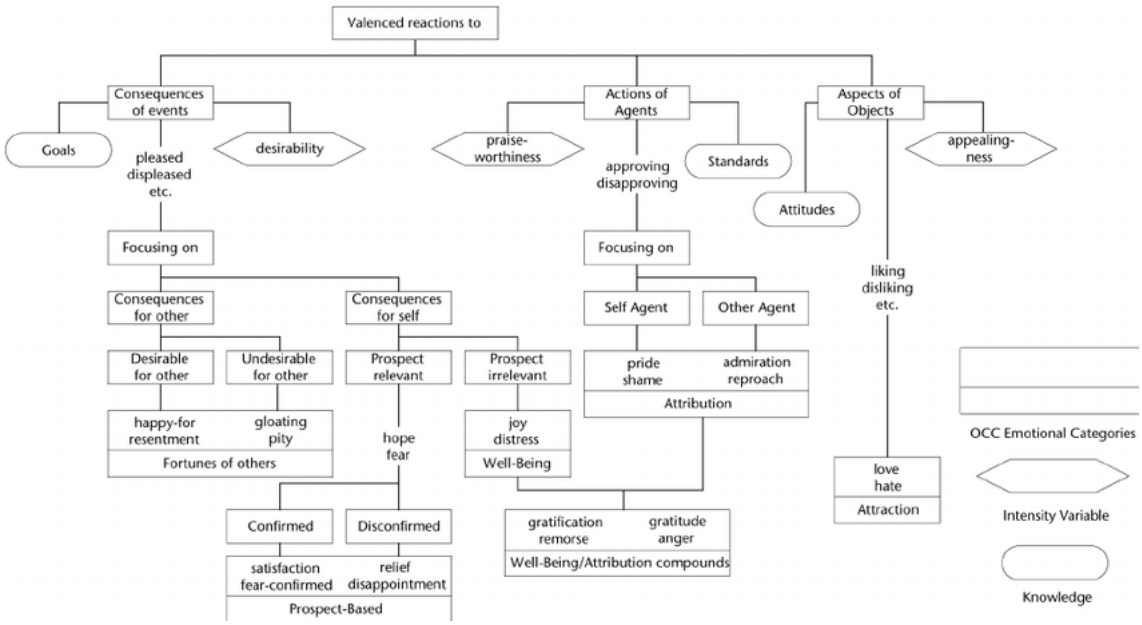

Figura 4: *OCC Model*

A aplicação deste modelo permite aos robôs expressar emoção, sendo necessário avaliar todas as situações em que o robô se possa deparar com tal emoção, devolvendo assim uma ação que represente de alguma forma essa mesma emoção.

Figura 5: Comportamento Emocional

Para desenvolver este robô foram tomadas decisões e algumas implementações importantes, tais como:

- *Win*, de forma a expressar "alegria"na ronda seguinte da batalha, através da alteração de cor.
- *Death*, de forma a expressar "tristeza"na ronda seguinte da batalha, através da alteração de cor.
- *BulletHit*, assim que o robô atinge uma taxa de 100% de acerto nos últimos 5 disparos, altera a cor do corpo para manifestar "orgulho". Passa ainda a ignora qualquer restrição no controlo de disparo.
- *BulletMissed*, caso o robô atinge uma taxa de 100% de falha nos últimos 5 disparos, altera a cor do corpo para manifestar "vergonha". Passa ainda a disparar apenas em metade das vezes que deteta um inimigo.

2.5 Equipa Robôs

De forma a facilitar a carga computacional da equipa e permitir uma divisão mais eficiente das tarefas, decidimos criar dois tipos de robôs para esta equipa, onde temos um *Team Leader* e dois *Droids*, sendo que cada um é responsável por diferentes funções.

O robô *Team Leader* assume as funções de liderar a equipa, mantendo o estado do mapa, ou seja , toda a informação relativa aos robôs adversários, aliados e neutros (robôs estáticos a simular obstáculos). O *Team Leader* comunica com os *Droids* passando-lhes a informação necessária para que estes entrem em ação. Os *Droids* não possuem radar para a identificação do ambiente, sendo a informação proveniente do *Team Leader* essencial para iniciar os mecanismos e ação.

Tática de jogo

Com a estrutura da equipa concluída, foram analisadas as melhores abordagens para a tomada de decisão, que todos os robôs devem possuir.

- O *Team Leader* circula em torno do mapa evitando paredes e desviando-se de obstáculos estáticos, isto enquanto regista e atualiza toda a informação dos inimigos e aliados.
- O *Team Leader* envia informação dos inimigos e obstáculos existentes dentro do ambiente para os *Droids*.
- Os *Droids* posicionam-se de forma a evitar colisões entre elementos da mesma equipa.
- Ao longo da ronda , os *Droids* recebem informações de inimigos e atuam conforme o *Team Leader* visa necessário.
- Em caso de embate verificam se o robô é inimigo, obstáculo ou *teammate*, sendo que em caso inimigo ele dispara na sua direção.

Estratégia e Arquitetura de Controlo

De forma a modelar e atuar sobre o ambiente, decidimos optar por uma estratégia de controlo *feedback* em conjunto com uma arquitetura híbrida, na qual os sensores recebem informações do ambiente, manter atualizada a mesma, para posteriormente ser usada pelo *Team Leader* tomar ações reativas em relação ao ambiente.

As estruturas que são utilizadas, essencialmente, para manter grande parte da informação do ambiente, são 3 estruturas que armazenam toda a informarão dos inimigos, obstáculos e equipa.

- private HashMap<String, RobotInfo> *teammates*;
- private HashMap<String, RobotInfo> *enemies*;
- private HashMap<String, Point2D.Double> *boxes*;

Atribuição de Obstáculos

O *Team Leader* têm como função atribuir aos *Droids* informação relativa a obstáculos, isto para ser possível evita-los.

O *Team Leader* para além de fornecer esta informação aos *Droids*, caso este obstáculo esteja presente na sua rota este vai contornar através das coordenadas associadas ao obstáculo.

Os *Droids* para além das instruções facultadas pelo *Team Leader*, não considera inimigos como obstáculos, logo caso exista choque ele vai disparar na sua direção, até o *Team Leader* dar ordem diferente.

Controlo de disparo

Como os *Droids* não possuem radares, o *Team Leader* está responsável por enviar as coordenadas dos robôs oponentes para que estes possam disparar. Associado a esse disparo esta o facto de na frente da linha se localiza um aliado ou obstáculo. Os posicionamentos atribuídos e as ordens enviadas pelo *Team Leader*, visam minimizar este mesmo problema.

Planeamento de trajetória

De modo a que o *Team Leader* se consiga mover ao longo da borda do campo de jogo, para evitar a parede, qualquer obstáculo (não inimigo) que se encontre na zona será contornado. Assim que o *Team Leader* comece a analisar o terreno de jogo, é feito um mapeamento do percurso de forma a que os *Droids* possam auxiliar o *Team Leader* e de forma a poderem evitar obstáculos.

Inicialmente O *Team Leader* apenas contêm 4 pontos nos seus ciclos que depois podem sofrer ligeiras alterações devida a existência de obstáculos.

Com estes quatro pontos e a parede que se situa a uma certa distancia, caso exista um obstáculo na sua rota, este vai contorna-lo voltando mais uma vez à sua rota.

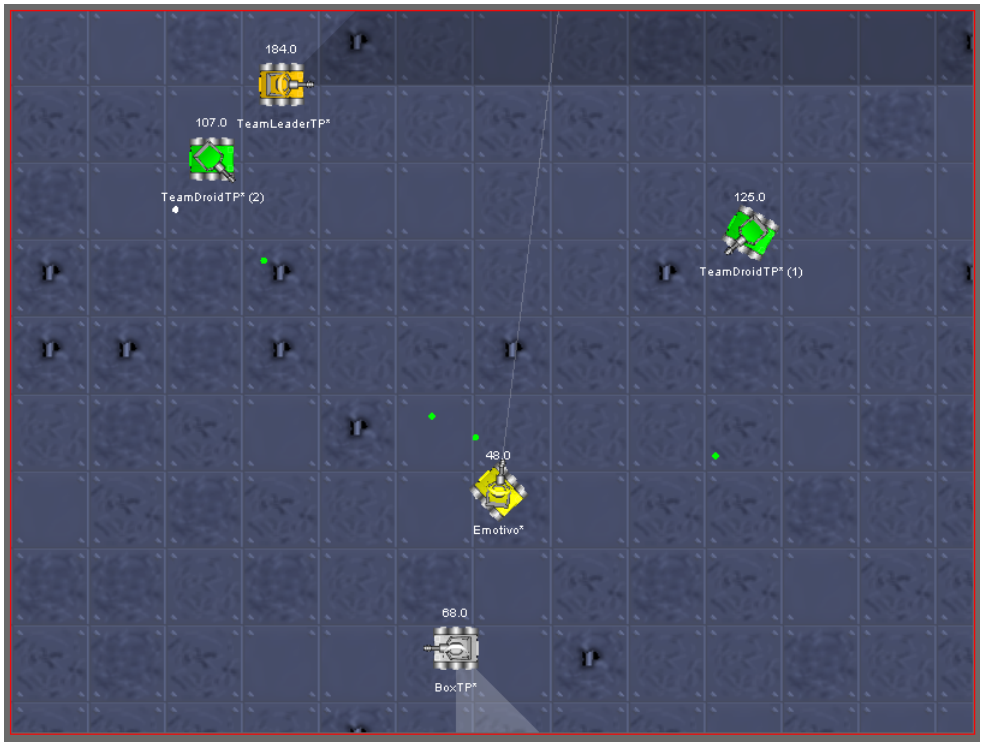

Figura 6: Equipa Robôs

3 Conclusões e Trabalho Futuro

Durante a elaboração deste projeto foi possível, não só o uso geral da programação no ambiente *Robocode*, como também permitiu ampliar os conhecimentos obtidos na Unidade Curricular de Agentes Inteligentes, do semestre anterior.

Uma das principais dificuldades encontradas durante a construção deste projeto foi o facto de tentar otimizar algoritmos criadas para situações especificas, por exemplo, durante a execução de planeamento de trajeto e contorno de obstáculos nas equipas foi necessário continuadamente alterar a estratégia, isto porque não seria viável para a maioria das situações.

Uma próxima fase de desenvolvimento dos robôs seria no sentido de melhorar a política de disparo dos *Droids* e aperfeiçoamento das "emoções"do robô *Emotivo*, visto que o *OCC Model* foi simplificado.