Agentes e Sistemas Multiagente (1º ano do MEI)

Conceção e Implementação de Um Sistema Multiagente

Relatório de Desenvolvimento

João Pereira	Luís Vieira	Pedro Barbosa
PG47325	PG47430	PG47577

21 de maio de 2022

Resumo

O presente trabalho prático, realizado no âmbito da unidade curricular de Agentes e Sistemas Multiagente inserida no perfil de Sistemas Inteligentes, visa a conceção e implementação de um Sistema Multiagente, com agentes inteligentes, utilizando as técnicas abordadas ao longo do semestre.

Este irá começar com uma introdução e contextualização sobre o tema em questão. De seguida, o mesmo irá ser aprofundado através de um esclarecimento sobre os diferentes domínios tratados, quais os objetivos definidos e a forma como os iremos atingir. Seguidamente, será feita uma descrição do sistema multiagente desenvolvido assim como uma explicação da sua arquitetura e funcionamento. Finalmente, os resultados obtidos serão submetidos a uma análise crítica e serão apresentados um conjunto de sugestões e melhorias para o sistema desenvolvido.

Conteúdo

Capítulo 1

Introdução

Os Sistemas Multi-Agente (SMA) são sistemas compostos por múltiplos agentes, que exibem um comportamento autónomo, mas ao mesmo tempo interagem com os outros agentes presentes no sistema. Estes agentes exibem duas características fundamentais: serem capazes de agir de forma autónoma tomando decisões levando à satisfação dos seus objectivos; serem capazes de interagir com outros agentes utilizando protocolos de interacção social inspirados nos humanos e incluindo pelo menos algumas das seguintes funcionalidades: coordenação, cooperação, competição e negociação.

Assim, o presente trabalho tem como principal objetivo a elaboração de um conjunto de agentes inteligentes e autónomos que irão explorar um mundo de duas dimensões onde serão agrupados em duas equipas, cada uma com cinco elementos. No que concerne a cada agente, este deverá explorar, descobrir e localizar agentes oponentes e colaborar com agentes da sua equipa para perseguir e eliminar cada agente.

Deste modo, o grupo decidiu começar por fazer uma análise do enunciado e o levantamento dos requisitos necessários para a realização sucedida do projeto. Seguidamente, idealizamos um design e uma modelação de arquitetura distribuída baseada em agentes para a resolução do problema em mãos. Finalmente, através da utilização da biblioteca *JADE*, implementamos os diferentes agentes e o respetivo sistema multiagente onde os mesmos se encontram.

Posto isto, e tendo em consideração todos os tópicos acima abordados, este relatório visa ajudar a compreender e explicar todos os raciocínios e tomadas de decisão feitas de forma a conseguir realizar todas os objetivos para este projeto e, assim, atingir o resultado final desejado.

Capítulo 2

Domínio do Problema e Objetivos

2.1 Domínio

O domínio deste problema situa-se na área de Sistemas Inteligentes, com especificidade em Agentes e Sistemas Multiagente. No âmbito particular do mesmo, um conjunto de agentes inteligentes e autónomos num mundo bi-dimensional, dividido em linhas e colunas, que é habitado por duas equipas diferentes, equipa A e equipa B, que, por sua vez, são compostas por cinco agentes diferentes.

2.2 Objetivos

Os objetivos deste projeto são os diferentes agentes explorarem o sistema, descobrirem a localização de agentes oponentes e colaborarem com os companheiros da sua própria equipa para perseguir e eliminar cada adversário. Consequentemente, um agente é considerado eliminado se for cercado por elementos da equipa adversária em quatro direções, sejam elas na horizontal e na vertical ou nas diagonais, sendo os casos de borda do tabuleiro e cantos do mesmo sujeitos a condições especiais.

Por sua vez, cada agente apresenta um campo de visão de dimensões 7 por 7 e apenas conseguem comunicar com outros agentes da mesma equipa. Deste modo, torna-se necessário organizar os processos de comunicação e colaboração entre agentes da mesma equipa com o intuito de melhorar a eficácia dos mesmos no que concerne a planeamento e tomada de decisões. Em adição, é fulcral a conceção de estratégias de procura e posicionamento colaborativas de forma a melhorar a performance de cada equipa.

Tendo estas informações em mente, o grupo propôs que para atingi-los iria começar por definir o design e a arquitetura geral do sistema tendo em conta o que achava ser necessário. Seguidamente, iria iniciar a implementação dos agentes no sistema criado e, conforme fosse necessário, apontar as alterações necessárias em relação ao planeamento inicial. Depois, iria tratar de conseguir fazer um relatório de jogo com algumas estatísticas que considerou serem importantes. Por fim, iria refazer os modelos inicialmente criados consoante as mudanças anteriormente mencionadas.

Capítulo 3

Descrição da solução final

Relativamente à descrição final do sistema, este foi implementado num tabuleiro de dimensões 35 por 35 de acordo a cumprir com todos os objetivos delineados pelos docentes. Desta forma, o sistema criou-se tendo em conta três tipos diferentes de agentes.

- **Manager -** Responsável por gerir o tabuleiro, dar prioridade à equipa que deve jogar na ronda e mover os *players* para as posições escolhidas pelo respetivo *coach*.
- **Coach -** Responsável por decidir a tática e tomar as decisões necessárias com o intuito de ganhar o jogo. Assim, escolhe o que cada um dos seus *players* deve fazer na respetiva ronda.
- **Player -** É criado e recebe ordens do *coach* sendo movido pelo *manager*. Aquando da sua morte, este mata-se e sai do sistema.

Capítulo 4

Arquitetura e funcionamento

Neste capítulo será apresentado o design do sistema final, a modelação da arquitetura (baseada em agentes) proposta pela equipa, utilizando para tal as metodologias UML. Esta primeira fase permitiu desenhar uma solução para o problema anteriormente descrito, bem como a idealização e caracterização de todos os componentes necessários para alcançar os objetivos do sistema.

Relativamente a todos os componentes construídos, definimos também as interações que estes têm para poderem colaborar, trocando mensagens. Só desta forma é que conseguem cooperar para derrotar a equipa adversária.

Além do design e modelação será também descrito o funcionamento do sistema, considerando detalhes mais específicos do mesmo, como por exemplo os aspetos relativos à codificação, que foi implementada sempre em concordância com a arquitetura. É de notar que todos os diagramas dizem respeito a uma arquitetura **centralizada**, em que todas as decisões táticas tomadas são decididas pelo treinador da equipa, e portanto, este age por todos os seus jogadores.

4.1 Diagrama de classes

O primeiro diagrama que decidimos criar foi o diagrama de classes, visto que nos permitiu desenhar todas as interfaces, agentes e classes de apoio para a construção do sistema.

Como se pode observar na figura 4.1 o nosso sistema é composto por 3 agentes: o **Player**, o **Manager** e o **Coach**.

O Player precisa de conhecer a sua equipa, quando é inicializado pelo *MainContainer* e deve armazenar o seu identificador no formato de String. A sua função é notificar o coach da sua existência, indicando que será membro da equipa e que está pronto para jogar. Outro comportamento que tem é "matar-se", quando recebe ordem para tal, proveniente do Manager.

O Manager tem conhecimento do seguinte:

- Qual o tamanho do tabuleiro;
- Qual o máximo de rondas que se podem jogar por jogo;
- Se o jogo já começou;
- Qual é a ronda atual;

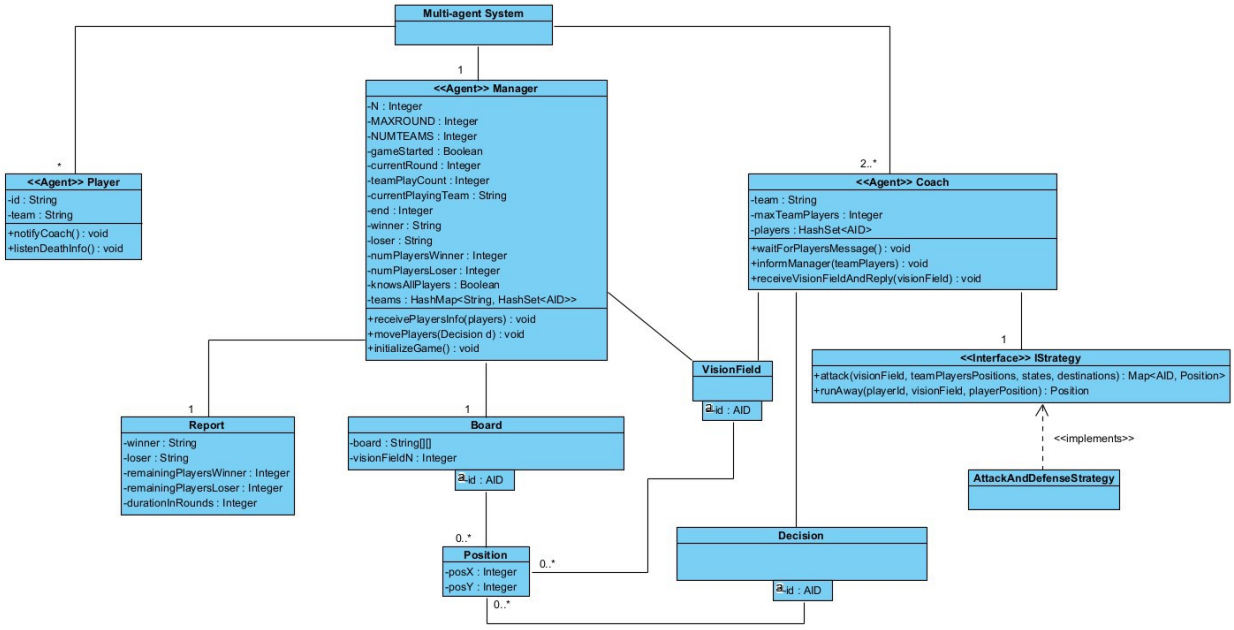

Figura 4.1: Diagrama de classes

- Quantas equipas já jogaram na ronda atual;
- Qual a equipa que está a jogar;
- Qual o vencedor e o perdedor;
- Quantos jogadores permaneceram vivos na equipa vencedora e na equipa perdedora;
- Se já conhece todos os jogadores de cada equipa (utilizado no processo de inicialização);
- A constituição das equipas.

O comportamento deste agente resume-se a receber a constituição das equipas, na inicialização, mover jogadores, inicializar o jogo e enviar os campos de visão à próxima equipa a jogar. De acordo com as suas características este tem acesso a várias classes: **Report** para criar um relatório do jogo quando este acaba; **Board** para fazer a gestão do tabuleiro e **VisionField** para transportar os campos de visão entre este e o Coach.

Por fim, o Coach tem conhecimento da sua equipa, dos seus jogadores e do número máximo de jogadores que podem constituir a equipa. Este é responsável por esperar que todos os jogadores anunciem a sua presença na equipa, informar o Manager de quem constitui a sua equipa e receber os campos de visão a cada ronda, bem como tomar as decisões de acordo com uma estratégia específica. Assim, este possui acesso às seguintes classes: **VisionField** para receber os campos de visão; **Decision** para transportar as suas decisões para o Manager. Adicionalmente tem acesso a uma interface que representa a estratégia de jogo, neste caso implementamos uma estratégia denominada **AttackAndDefenseStrategy**, mas facilmente se podiam incluir mais classes para esta interface, com estartégias diferentes de ataque e defesa.

4.2 Diagrama de colaboração

Em seguida desenvolvemos um diagrama de comunicação para indicar todas as comunicações entre os agentes, tendo em conta a ordem.

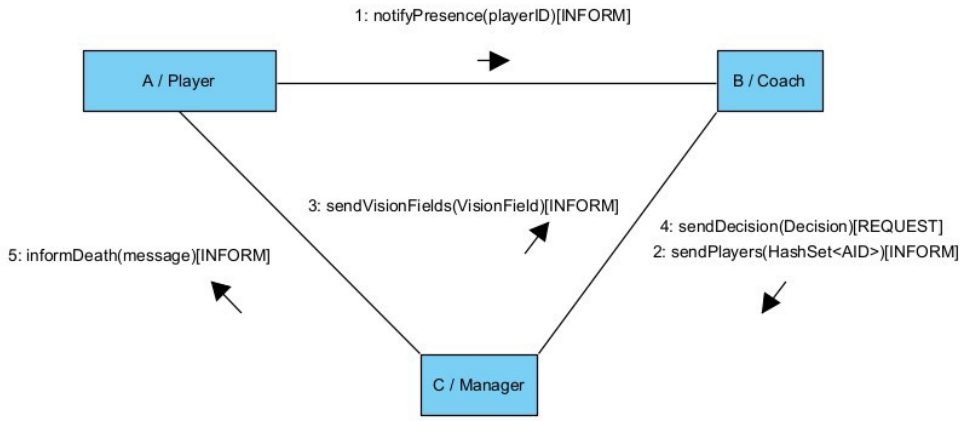

Figura 4.2: Diagrama de colaboração

De acordo com a figura 4.2 verificamos que é possível haver 5 tipos de comunicação: o Player notifica o Coach da sua presença; cada Coach envia os seus jogadores ao Manager; este envia os campos de visão dos jogadores ao Coach em cada ronda; cada Coach toma a sua decisão, quando for a sua vez de jogar; o Manager comunica com o Player se este eventualmente morrer.

4.3 Diagrama de atividade

Este diagrama também se revelou essencial para entender o funcionamento geral do sistema, tendo em conta todas as atividades que este deverá ter, de modo a cumprir os objetivos do problema. Incluir de uma forma um pouco mais detalhada, as interações entre os agentes, bem como as tarefas de cada um deles.

Na figura 4.3 é apresentado o fluxo de tarefas presente no nosso sistema multiagente. O Manager é o agente que possui maior parte das tarefas, visto que tem dois papéis bastante importantes, exercer a função de gestor do próprio sistema, como também árbitro do jogo. A recursividade descrita no diagrama diz respeito ao processo repetitivo que o Manager tem de fazer para cada ronda do jogo.

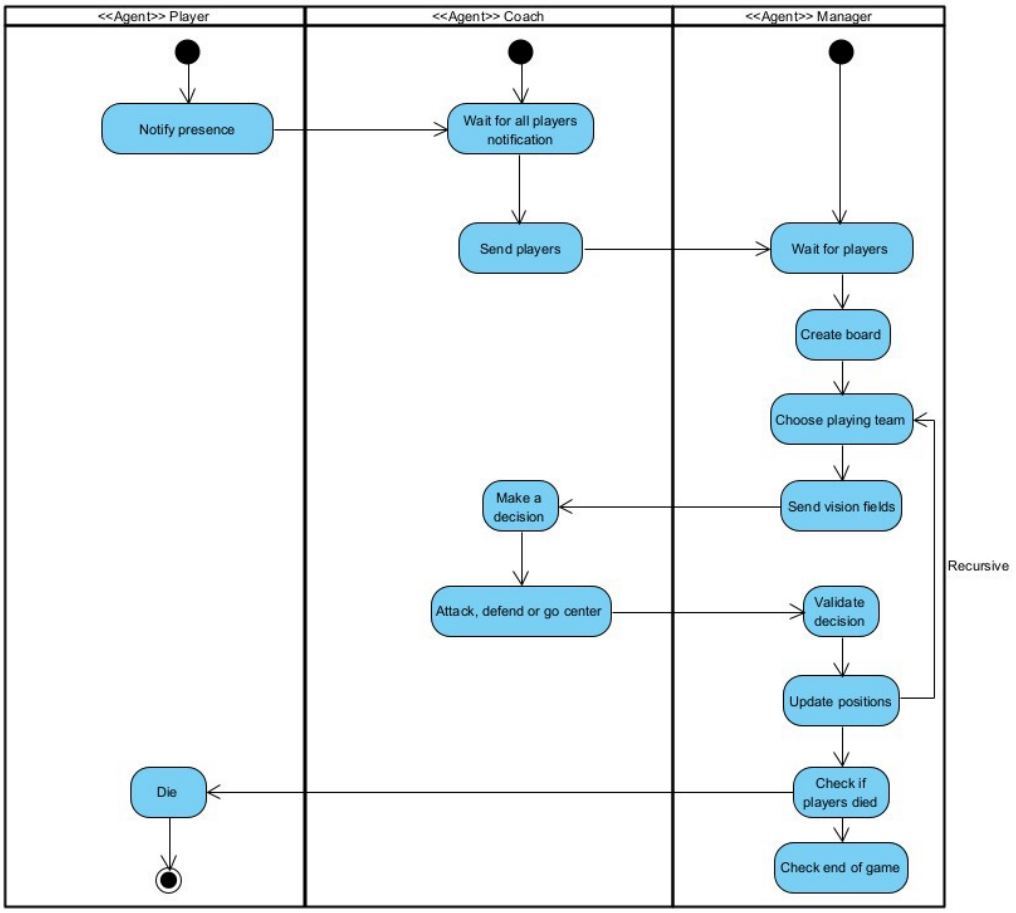

Figura 4.3: Diagrama de atividade

4.4 Diagrama de máquina de estados

Por fim, incluímos também um diagrama representativo dos possíveis estados de cada jogador da equipa, na fase de tomada de decisão por parte do treinador. Para tal este verifica o caso específico de cada jogador antes de agir. Dependendo do campo de visão de cada jogador, estes podem tomar estados diferentes, que depois serão usados para completar a estratégia.

De uma forma geral, sempre que um jogador possui vantagem sobre um inimigo que está no seu campo de visão, ou então está num estado de igualdade (por exemplo 2 da equipa A contra 2 da equipa B), o treinador marca esse jogador com o estado de ataque. Se um jogador está numa situação de desvantagem (por exemplo 2 contra 1), é marcado com o estado de fugir (defesa). O outro estado possível é o de ir em direção ao centro do tabuleiro (se no seu campo de visão não encontrar inimigos). Este último estado poderá ser posteriormente alterado para ataque. Mais tarde será descrito em detalhe o processo de ataque para explicar esta situação. A mudança de estado está então representada na figura 4.4

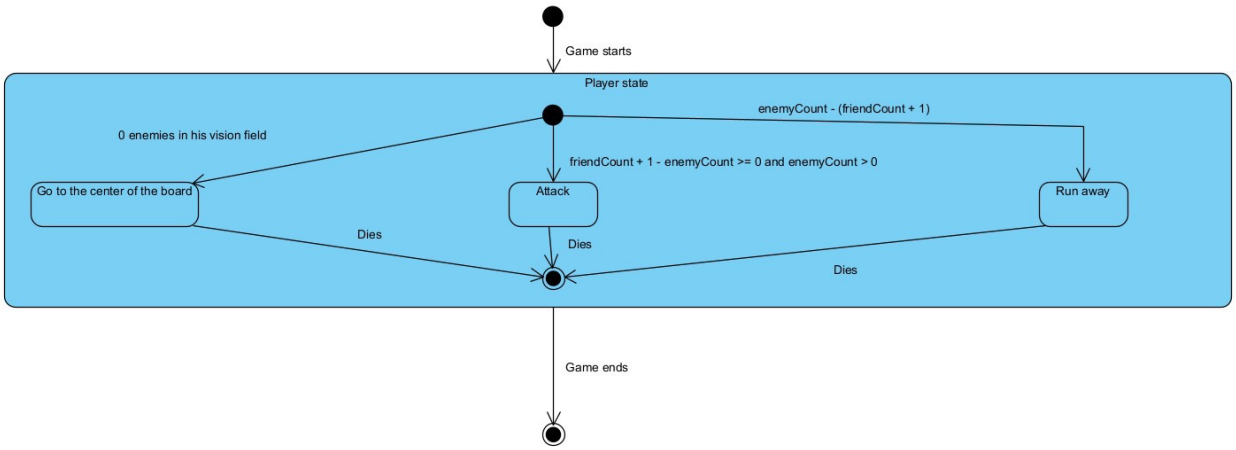

Figura 4.4: Diagrama de máquina de estados

4.5 Casos mais específicos do funcionamento do sistema

4.5.1 Inicialização dos agentes

Na inicialização dos agentes, incluída no *MainContainer*, podem ser fornecidos argumentos aos diferentes agentes. Em primeiro lugar, é criado o Manager, que não recebe argumentos, depois são criados os treinadores, coachA e coachB, e estes recebem a sua equipa bem como o número máximo de jogadores que podem integrar na mesma. Por fim, são criados os players que recebem o seu id (A1, B1, ...).

4.5.2 Envio dos jogadores para o Manager

O coach só envia os ids dos seus jogadores ao manager quando tem a sua equipa completa.

4.5.3 Tomadas de decisão para mover os jogadores

Nesta função específica, o coach analisa o campo de visão de cada um dos seus jogadores, um de cada vez. Em primeiro lugar analisa quantos *teammates* e quantos inimigos o jogador tem no seu campo de visão. Consoante essa contagem entra num dos estados descritos anteriormente no diagrama, de acordo com as condições também lá estabelecidas. Se o estado correspondente for o de defesa, ou seja, fugir, a ação fica imediatamente confirmada e o jogador irá fugir. Se o estado for de ataque, então esse jogador irá fazer parte de um ataque, mas pode recorrer à ajuda de outros jogadores nesse ataque. Desta forma, os jogadores que provisoriamente estão no estado de ir para o centro do tabuleiro, podem ainda mudar para o estado de ataque.

Portanto, o coach verifica se é preciso atacar de acordo com o ataque implementado pela classe representativa da estratégia. Nós utilizamos apenas uma estratégia de teste, e portanto no nosso caso o ataque é feito da seguinte forma: o treinador irá escolher um campo de visão em que o seu jogador está no estado de ataque. Se existirem vários jogadores nesta situação escolherá o campo com a melhor condição de vantagem, ou seja, em que a diferença entre o número de jogadores da equipa e o número de inimigos seja maior. Depois de selecionado o campo, é escolhido um inimigo aleatório dentro desse campo, e todos os jogadores com o estado de ataque, ou com o estado de ir para o centro, ficarão com o estado de ataque e serão utilizados para ir atrás desse inimigo.

4.5.4 Modo de funcionamento da fuga, do ataque e da ida ao centro do tabuleiro

Para **fugir**, é feita uma recolha de todas as posições para onde o jogador pode ir, e depois é verificado se a posição destino já contém algum jogador do seu campo, e se ficará na linha ou na coluna de um inimigo. Qualquer posição que não esteja nesta situação será escolhida.

Para **ir em direção a um inimigo ou ir em direção ao centro**, é utilizada uma abordagem que funciona nos dois casos: são recolhidas as posições destino válidas do jogador, e depois é escolhida aquela que tiver menor distância em relação ao centro ou em relação ao jogador alvo. A distância utilizada é a distância euclidiana bidimensional segundo a fórmula:

$$d = \sqrt{(coluna2 - coluna1) * (coluna2 - coluna1) + (linha2 - linha1) * (linha2 - linha1)} \quad (4.1)$$

4.5.5 Atualização das posições dos jogadores por parte do Manager

Antes de atualizar o tabuleiro, o Manager verifica sempre se a posição destino é válida, verificando se existe lá algum jogador. Caso seja inválida não move o jogador.

4.5.6 Casos em que um jogador morre

No nosso sistema permitimos os jogadores moverem-se nas diagonais, portanto definimos que os casos em que um jogador morre são divididos em 3 partes (meio, bordas ou cantos), como se apresenta na figura 4.5.

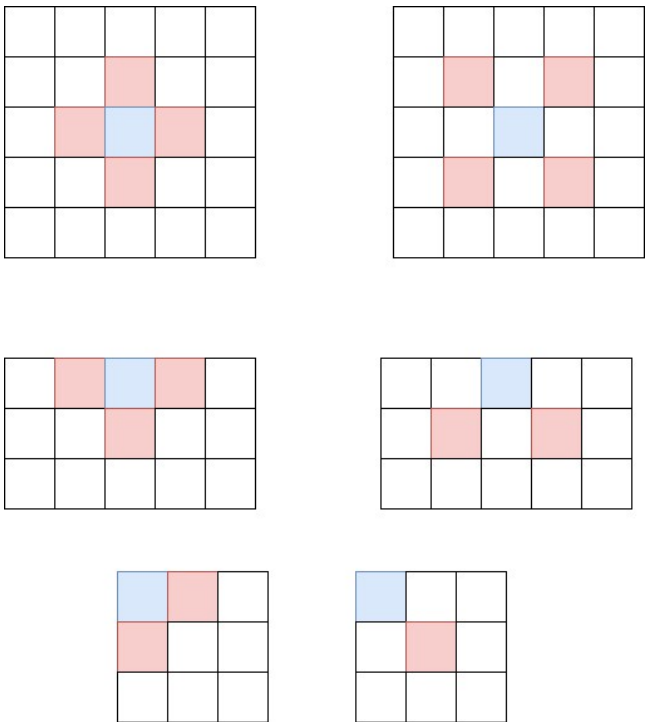

Figura 4.5: Casos de morte do jogador (Jogador a azul e inimigos a vermelho)

4.5.7 Condições de paragem do jogo

Assumimos que o jogo acaba depois de 150 rondas, quando uma das equipas fica sem jogadores ou quando permanece apenas um jogador numa equipa mas não tem posições livres para fugir.

4.5.8 Começo do jogo

A seleção da equipa que começa a jogar é feita aleatoriamente, bem como as posições iniciais de cada jogador.

Capítulo 5

Resultados obtidos e análise crítica

Depois de implementar o nosso sistema multiagente, realizamos uma simulação manual de 20 jogos para extrair informações relevantes sobre os mesmos. Calculamos algumas estatísticas relevantes para o problema, apresentadas na seguintes tabelas:

Casos/Equipa	A	B
Vitórias	55%	40%
Vitórias eliminando todos os inimigos	30%	15%
Começou a jogar	75%	25%
Número de inimigos eliminados (em média)	2.15	1.45
Número de jogadores vivos (em média)	3.55	2.85

Tabela 5.1: Estatísticas resultantes da simulação de 20 jogos

Número de rondas (em média)	121.5
Número de empates	1
Começou a equipa A, e ganhou a equipa B, ou vice-versa	35%
Ganhou a mesma equipa que começou	60%

Tabela 5.2: Outras estatísticas relativas à mesma simulação

Depois de analisar os resultados obtidos, consideramos que a estratégia **AttackAndDefenseStrategy** adotada é uma boa estratégia, visto que conseguimos encontrar casos em que o jogo termina antes das 150 rondas máximas. Algumas das vezes até é possível eliminar a equipa adversária inteira e portanto, consideramos que o ataque se encontra bom. No que diz respeito à defesa, também consideramos que é boa sendo que conseguimos encontrar um caso de empate, em que ambas as equipas ficam com todos os jogadores vivos após 150 rondas.

Apesar disso, muitas vezes o jogo pode ser condicionado pela aleatoriedade na inicialização. Há jogos em que os jogadores vão começar todos muito juntos e este acabará mais rápido, enquanto que há jogos em que começam todos muito afastados e nunca chegam a eliminar ninguém, devido às grandes dimensões do mapa.

Capítulo 6

Sugestões e recomendações

O sistema multiagente desenvolvido parece cumprir os objetivos principais propostos no enunciado deste projeto, de acordo com os resultados obtidos. É possível finalizar o jogo com sucesso e a estratégia revelou-se eficiente.

No entanto, gostaríamos de sugerir possíveis melhorias no nosso sistema para completar num trabalho futuro. Em primeiro lugar deveríamos ter implementado uma nova classe com uma estratégia diferente da utilizada nos testes, para conseguirmos comparar as duas abordagens. Apesar de não o termos feito, implementamos a interface da estratégia para tornar o trabalho mais modular e permitir a inclusão de diferentes estratégias sem condicionar o funcionamento dos restantes componentes deste sistema.

Outra sugestão seria adicionar um modelo de *Reinforcement Learning*, tal como foi sugerido no enunciado. Este seria incluído nos treinadores das equipas para que pudessem aprender com as suas decisões, sendo recompensados pelas decisões mais eficientes.

Outra possível melhoria seria experimentar uma abordagem mais descentralizada, em que cada jogador poderia comunicar com o seu parceiro para eliminar os adversários.

Capítulo 7

Conclusão

Dado por concluido este projeto prático, o grupo considera ter realizado um bom trabalho. Para além de pôr em prática aquilo que foi aprendido nas aulas ao longo do semestre, fazendo dessa forma uma melhor consolidação da matéria, achamos que conseguimos cumprir o objetivo principal de conceber um sistema multiagente com diversos agentes que nele habituam e atuam e pô-los a funcionar de acordo com a comunicação que exercem uns com os outros.

Deste modo, e de forma a concluir o trabalho realizado, consolidamos todo o conhecimento desde o levantamento inicial de requisitos que achamos serem necessários para o projeto assim como um esboço inicial de como a comunicação entre os diferentes agentes deveria ser feita até à implementação física de toda essa preparação. Ao longo do caminho, deparamo-nos com alguns obstáculos, alguns mais difíceis que outros, mas que conseguimos sempre ir superando através do trabalho em grupo.

Em jeito de conclusão, o grupo acha que realizou um bom trabalho e que os objetivos a que se propôs com o mesmo foram bem delineados e alcançados com sucesso.