Hide and Seek

Agentes e Inteligência Artificial Distribuída 2019

Descrição do Problema

Este trabalho consiste em colocar duas equipas (uma de *hiders* e outra de *seekers*), a jogar ao jogo das escondidas. O jogo termina quando um *seeker* estiver à beira de um *hider*, ganhando a equipa dos *seekers*, ou quando chegar ao fim o número de passos de tempo predefinido, ganhando a equipa dos *hiders*.

Inicialmente, os *hiders* têm um determinado número de passos de tempo para se poderem movimentar e esconder (um período de aquecimento), sendo que os *seekers* estão parados. Quando este número expira, todos as equipas estão em jogo.

Agentes

Ambos os tipos de agentes em jogo, devido às suas semelhanças, herdam de GameAgent. Estes têm o conhecimento da sua posição (x,y) no mapa, da sua orientação (CIMA, BAIXO, DIREITA, ESQUERDA), das posições do mapa que constam no seu campo de visão, e dos possíveis movimentos que podem realizar no próximo passo.

SeekerAgent

HiderAgent

Agentes

O Game*MasterAgent* é responsável por controlar o fluxo de jogo. Conhece todos os outros agentes, e é o único que tem acesso ao mundo. Tem um conjunto de comportamentos que lhe permite comunicar com os agentes para, por exemplo, sinalizar uma jogada, ou sinalizar aos agentes que o período de aquecimento terminou.

Interação e Protocolos

Para permitir a comunicação entre os agentes, todos os eles têm um comportamento cíclico (Listener) para a receção de mensagens. Cada mensagem tem um cabeçalho específico que, depois de interpretado pelo agente que a recebe, invoca o comportamento apropriado para o seu tratamento.

As mensagens utilizadas contêm a seguinte estrutura:

- HEADER;[PARÂMETRO;]*

Os parâmetros utilizados tanto podem ser coordenadas (separadas por ‘,’), booleanos ou floats.

Estratégias dos Agentes

O objetivo dos agentes é muito semelhante, mas com espetros opostos: os *hiders* querem-se afastar dos *seekers*, e os seekers querem-se aproximar dos hiders. Portanto, os agentes usam também estratégias muito semelhantes: ao escolher um movimento, têm em conta os adversários que eles e os seus colegas de equipa conseguem ver (após comunicarem com estes) e, com base no tipo da sua equipa, tentam aproximar-se/afastar-se dessas posições conhecidas. Caso não conheçam nenhum adversário, movem-se aleatoriamente.

De notar que um movimento dá-se sempre para uma casa de distância do agente.

Outros mecanismos

De forma a ser possível conhecer outros agentes num determinado momento, é usado o mecanismo de páginas amarelas. Desta forma, todos os agentes se registam no serviço, e o *GameMasterAgent* pode procurar pelos restantes agentes, enquando que os *hiders* e os seekers podem procurar pelo *GameMasterAgent* de forma a permitir comunicação bidirecional. Cada *hider* e cada *seeker* procura também pelos elementos da mesma equipa.

Software Utilizado

O programa foi escrito na linguagem JAVA, com recurso ao JADE, de modo a permitir correr os agentes, e adicionar comunicação entre os mesmos. Todas as outras funcionalidades, como cálculo do campo de visão, foram feitas com recurso a algoritmos como Ray Tracing, mas sem qualquer *software* ou *package* adicional.

Experiências Realizadas

As experiências foram realizadas com recurso a mundos (representados em ficheiros de texto) com diferente complexidade, constando de espaço aberto e paredes. Em ambos os mapas, as equipas são de 2 elementos, e são usados 50 passos de tempo, dos quais 20 são de aquecimento.

	Mundo Aberto	Paredes
Experiência 1	Seekers - 21 pt	Seekers - 24 pt
Experiência 2	Seekers - 21 pt	Seekers - 25 pt
Experiência 3	Seekers - 22 pt	Seekers - 25 pt
Experiência 4	Seekers - 21 pt	Seekers - 27 pt
Experiência 5	Seekers - 21 pt	Seekers - 25 pt

Análise dos Resultados

Em todas as experiências realizadas, os seekers ganharam com relativa facilidade (menos de 10 passos de tempo após o aquecimento terminar), sendo que se repara numa diferença de aproximadamente 5 passos de tempo entre as experiências de cada mapa.

Tais resultados justificam-se pela pequena dimensão dos mapas em relação ao número de agentes. Além disso, os *hiders* carecem de algoritmos de inteligência artificial como aprendizagem por reforço, portanto não têm o conhecimento suficiente para se conseguirem esconder dos *seekers*.

Conclusões

Não foi possível implementar o jogo descrito na especificação na sua totalidade: nomeadamente presença e interação com blocos e rampas, visto que subestimamos a carga de implementar campo de visão e comunicação entre agentes.

Futuramente, acreditamos que seria bastante interessante a implementação de algoritmos de aprendizagem por reforço. O grande obstáculo que se coloca é o objetivo dos agentes ser dinâmico: os *seekers* querem procurar os *hiders*, que podem estar em movimento, e os *hiders* querem-se esconder dos *seekers*, que também podem estar em movimento.

Informação Adicional

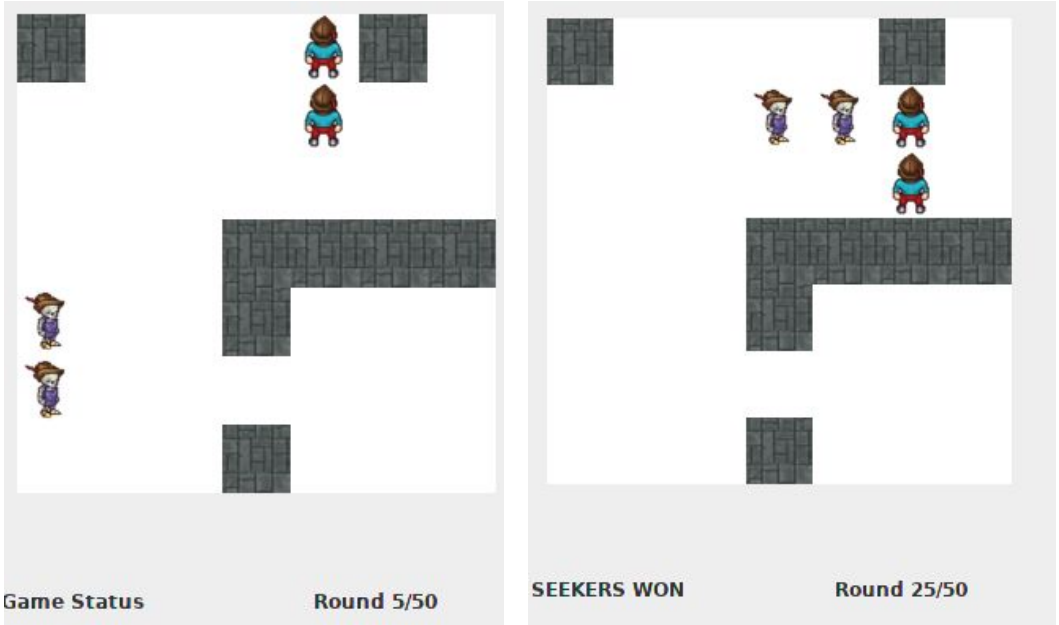
Exemplo de Execução

Como compilar e executar:

$ cd src

$ javac -cp ../lib/jade.jar:. *.java

$ java -cp ../lib/jade.jar:. HideNSeek ../res/worlds/<mundo>.txt



Exemplos da interface gráfica durante o decorrer de um jogo

Classes Implementadas

- HideNSeek - inicializa o programa e executa os agentes
- HideNSeekWorld - lê o mapa de um ficheiro e guarda as posições dos agentes
- GameAgent - representa um agente do jogo
- GameMasterAgent - responsável por controlar o fluxo do jogo
- HiderAgent - representa um hider
- SeekerAgent - representa um seeker
- FieldOfView - calcular campo de visão de um agente com *Ray Tracing*
- Ray - calcula a última célula vista numa determinada direção
- Position - posição (x,y) de um agente

Classes Implementadas

- GUI - interface gráfica (visualização da evolução do mapa)
- GameView - lê as imagens e guarda os gráficos da GUI
- Logger - guarda logs das comunicações para ficheiros de texto

Outras observações

No decorrer e final de cada execução, é possível consultar o diretório “logs”, onde constam 3 ficheiros:

- master_<timestamp>.txt - comunicação entre agentes e *master*
- hiders_<timestamp>.txt - comunicação entre *hiders*
- seekers_<timestamp>.txt - comunicação entre *seekers*

Outras observações

De forma a calcular o campo de visão dos agentes, uma das funcionalidades mais importantes do programa, foi implementada uma adaptação do algoritmo "Ray Tracing". Para cada agente, são gerados um número de raios numa amplitude de aproximadamente 140º (afinal, ninguém tem uma visão de 360º), tendo em conta a sua orientação. E para cada raio, são calculadas as células que o agente consegue ver, ou seja, aquelas que não são tapadas por paredes.

Assim, apesar da limitação de termos um mapa na forma de *array* bidimensional, conseguimos assegurar que o agente pode não só ver células na horizontal/vertical, como também diagonal.

Grupo 30

- Francisco Filipe - up201604601@fe.up.pt
- João Miguel - up201604241@fe.up.pt
- Pedro Fernandes - up201603846@fe.up.pt