deti universidade de aveiro
departamento de eletrónica, telecomunicações e informática

IA- Inteligência Artificial

TPG - Dig Dug

João Luís Nº107403, LEI
José Gameiro Nº108840. LEI
Diogo Falcão Nº108712, LEI

17 de dezembro de 2023 - P5

Domínio do Problema e organização do código

- Para este projeto de grupo, usamos programação imperativa - um algoritmo constituído por pesquisa - e programação declarativa.
- O agente criado, um agente reativo simples, reage de acordo com a situação apresentada pelo mundo atual com ações pré-definidas, como andar para cima, para baixo, para a direita, para a esquerda ou disparar.

- O código que nós desenvolvemos está organizado em 3 ficheiros: student.py, tree_search.py e complementary_funcions.py.
- No student.py lemos todas as informações necessárias para implementar a pesquisa do agente através do state.
- Criámos funções complementares em complementary_functions.py de modo a simplificar o código no ficheiro principal. Este ficheiro inclui as funções "`runAwayFromFygarFlame`", "`checkIfPookaIsGhost`", "`rockNearBy`", entre outras.
- Por último, no ficheiro tree_search.py, usámos um algoritmo de pesquisa que iremos especificar mais adiante.

Algoritmo de pesquisa utilizado

Para distância > 3:

- Utilizámos a pesquisa A* em árvore para calcular o caminho para o inimigo mais próximo e deslocarmo-nos até ele.

- Usámos a noção de distância de Manhattan para isso, tentando evitar que o nosso agente se deslocasse muitas vezes na diagonal uma vez que concluímos que *Pookas* em modo *ghost* nos traziam muitos problemas, quando entravam num túnel escavado neste género.

- Para o cálculo do custo, associamos um custo muito alto a coordenadas que tinham pedras, para evitar que o agente ficasse preso, e também aumentámos um pouco o custo para a posição imediatamente abaixo da posição da pedra, para evitar em níveis mais avançados que o nosso agente se matasse a ele próprio com a pedra, já que o número de pedras vai aumentando conforme o nível.

Algoritmo de pesquisa utilizado

Para distância <= 3:

- A nossa árvore de pesquisa passa do modo A* para condições (*if's*), quando o inimigo que o nosso agente identificou como mais próximo se encontra a uma distância menor do que 3 da posição atual do Dig dug, visto que a partir daqui, o nosso agente ele pode colocar-se em situação de perigo.

- Estas condições têm em consideração a posição e direção atuais do nosso agente, do inimigo, a distância entre o Dig Dug e o inimigo mais próximo e, por último, o estado em que se encontra o mapa.

- O mapa é atualizado com a ação do agente, antes de esta ser enviada para o servidor.

Conclusão e considerações finais

- Neste momento o nosso agente acaba por ficar muitas vezes preso no nível 7, correspondente ao aumento da inteligência dos inimigos.

- Para os próximos passos, umas das formas a melhorar a nossa solução seria implementar apenas pesquisa, diminuindo drasticamente o uso de if cases e generalizar a solução em vez de tentar solucionar cada problema caso a caso.

- Se implementássemos um algoritmo de pesquisa, iríamos analisar todas as situações possíveis e as suas consequências, posteriormente atribuiríamos a cada solução encontrada a soma entre o custo e a heurística e escolheríamos a solução que apresentasse um menor valor.