universidade de aveiro

Dig Dug AI Agent

MIGUEL AIDO MIRAGAIA Nº108317

GONÇALO RAFAEL CORREIA MOREIRA LOPES Nº107572

Inteligência Artificial – 23/24

Agent

O nosso agente não implementa algoritmo, a opção de implementação de "tree-search" foi considerada mas não foi executada pois após uma conversa com o professor Luís Seabra este não ter considerado estritamente necessário para a forma como iriamos desenvolver o jogo. Também ajudou o facto de termos um agente mais reativo e com um mecanismo de defesa mais fixo já implementado quando surgiu a opção de implementação.

Esse mecanismo baseia-se em manter sempre uma distância de 2 blocos para com os inimigos para prevenir ataques inesperados ou de alta probabilidade de morte. Apesar de grande importância para a longevidade do DigDug surgiram dificuldades no ataque em certos casos que irão ser referidos a seguir. Para resolver este problema desenvolvemos código para lidar com cada uma dessas situações da melhor forma (segura e inteligente).

Funções e ações

Existem algumas funções e ações que consideramos de alta importância para toda a logica de ação do agente. Sendo elas as funções **convert_direction_to_key_avoid_Pooka**, **orient_towards_enemy** e a ação **double jump**.

Convert_direction_to_key_avoid_Pooka: Função que permite um movimento defensivo em relação aos inimigos, DigDug movimenta-se em direção oposta a do inimigo e desta forma previne que se mantenham a uma distancia inferior a 2 blocos. Esta distância de segurança permite ter ações mais ponderadas e cuidadosas sem tomar os riscos de morrer inesperadamente por falta de opções/ações.

Orient_towards_enemy: Função de permissão de reorientação do DigDug para estar corretamente virado para o inimigo no momento do disparo, baseado no ultimo movimento do DigDug guardado na variável prev_move. Organiza as direções por distancia ao inimigo quando executadas e escolhe a que maximiza a sua distancia para não existir a mínima hipótese de se reorientar demasiado próximo do inimigo.

Ação double jump: Esta ação consiste num duplo movimento do DigDug, acontece sempre que 'e reconhecido que est'a um bloco acima de um inimigo e desta forma permite a sua reorientacao em seguranca. Com este duplo movimento tem espaco para se orientar corretamente para o inimigo e ainda assim manter uma distancia de seguranca.

Loops

Pela decisão de ter um agente mais reativo e defensivo levou a que algumas das suas ações levem a loops contínuos e por vezes infinitos. Para colmatar estes exceções foi desenvolvido código que permite lidar com elas.

Exemplos desses loops é o caso em que o DigDug mantem uma distancia de segurança de 2 blocos e por isso na tentativa de ataque na vertical fica preso num loop continuo. Esta situação foi lidada de maneiras diferentes para cada inimigo. Para o fygar foi implementada a função **move_carefuly__to_fygar**, para o Pooka continuamos dependentes da sua habilidade de traverse para sair desta situação, nada ideal no entanto usar a logica da função para o Fygar levaria a mortes demasiado prévias e consistentes, preferimos assim esta opção, menos logica, mais segura e com falhas menos consistentes (exceder o numero de steps). Sabendo do impacto que esta decisão tem na nossa pontuação criamos uma logica de ataque na horizontal ao invés da situação na vertical, demonstrou-se eficaz porem levava a outras excessões que por falta de tempo não foram possíveis resolver e portanto mantivemos a decisão de depender do Pooka.

- **move_carefully__to_fygar:** em suma esta função é usada para prevenir o loop, a sua execução acontece quando o DigDug apresenta a orientação correta para o fygar e tanto este como o digdug apresentam a mesma posição x e tem uma distancia que os separa por 2 blocos no eixo y. Se esta condição se verificar a variável "hold" é ativa e o DigDug espera uns steps até que o Fygar execute um movimento e desta maneira seja possível se aproximar dele sem correr o risco de estar numa situação insegura e causadora da sua morte.

Loops (continuação)

Quando um Fygar fica preso na horizontal entre uma rocha e um bloco não escavado, antigamente o DigDug, por conta do nosso metodo defensivo de ficar á espera que o fygar se afastasse para nos podermos aproximar, esperava infinitamente que este se movesse. Por forma a corrigir o loop, guardamos as ultimas 100 posições de o inimigo mais proximo num array e verificamos se todas as posições guardadas são iguais. Caso se verifique, ou seja, o fygar esteve sempre no mesmo sitio, movimentamo-nos na sua direção para o matar (sabendo que corremos o risco de morrer ao faze-lo).

Existe um loop em que o fygar se movimenta nas mesma duas posições infinitamente(porque percebe que estamos perto dele). Para resolver esse loop, implementamos algo parecido com o loop de cima, mas neste caso guardamos as ultimas 200 posições e vemos se só existem 2 posições diferentes. Este loop só se verifica em 2 situações, a primeira é quando o DigDug está em cima do Fygar e a outra é quando está por baixo. Caso se verifique o loop mandamos o DigDug ir para a direção contrária à do Fygar(ou para cima ou para baixo) para sair do loop e se redirecionar para matar o Fygar.