IA TPG
DigDug

Projeto final

universidade
de aveiro

Rodrigo Aguiar Nº 108969
Tomás Matos Nº 108624
Diogo Almeida Nº 108902

Implementação

- O nosso agente é maioritariamente reativo, visto termos achado que era a melhor forma, sendo que os inimigos e o próprio digdug se movem frame a frame.

- A cada frame, o "pensamento" do agente é o seguinte:
  - Move-se diretamente para o inimigo mais próximo
  - Se for seguro disparar e conseguir disparar, dispara contra o inimigo
  - Se não conseguir disparar, decide se deve fugir ou não do inimigo
  - Caso decida fugir, escolhe um bloco à volta dele seguro e dirige-se para lá

```
next_to_kill = sorted(
    state["enemies"], key=lambda k: distance_from2(k["pos"], digdug_pos)
)[0]

next_to_kill_pos = (next_to_kill["pos"][0], next_to_kill["pos"][1])
go_to = next_to_kill_pos
```

```
if safe_to_shoot(distance_to_enemy, next_to_kill, close_enemies):
    if can_shoot(
        digdug_pos, next_to_kill_pos, dig_dir, danger_zone, rocks
    ):
        shoot = True
```

```
if not shoot:
    if should_run(
        go_to, close_enemies, next_to_kill, digdug_pos, danger_zone
    ):
        go_to = find_closest_safe_block(
            digdug_pos, close_enemies, danger_zone, rocks
        )
```

Forma de Pesquisa

- Como forma de pesquisa continuamos a utilizar uma pesquisa A*, alterando os custos a cada frame que o dig dug realiza consoante a nova posição dos inimigos, das pedras e dos blocos de terra que estão escavados que vão sendo maiores . A cada frame vamos calcular o shortest path para o inimigo mais próximo, considerando uma heurística que consiste em somar o custo da posição onde estamos com o custo da posição de destino, que revelou achar os caminhos mais curtos não escavando blocos novos desnecessariamente.

Como base para esta pesquisa usámos os seguintes custos:

- 1 para qualquer posição já escavada
-
- custo imenso para posições com pedra, para não correr contra elas
-
- custo bastante grande de correr para a frente de fygar's (ou para trás também se estiverem a correr contra uma parede visto poderem virar-se e mandar fogo)
-
- custo bastante grande de correr diretamente para cima de um pooka
-
- 3 para qualquer bloco restante

```
for position in danger_zone:
    mapa.update_cost(position, 1)

for rock in rocks:
    mapa.update_cost(rock, 1000000000)

for fygar in fygars:
    mapa.update_cost((fygar[0], fygar[1]), 10000)
    for pos in range(1, 5):
        if (
            (fygar[2] == 1)
            or ((fygar[0] - 1, fygar[1]) not in danger_zone)
        ) and (fygar[0] + pos < size[0]):
            mapa.update_cost((fygar[0] + pos, fygar[1]), 10000)
        if (
            (fygar[2] == 3)
            or ((fygar[0] + 1, fygar[1]) not in danger_zone)
        ) and (fygar[0] - pos >= 0):
            mapa.update_cost((fygar[0] - pos, fygar[1]), 10000)

for pooka in pookas:
    mapa.update_cost((pooka[0], pooka[1]), 10000)
```

Melhorias

Com a nossa primeira implementação, o dig dug já lidava eficientemente com Pookas.

Sendo assim focamo-nos na 2ª entrega a melhorar o comportamento em relação aos Fygar's.
Implementámos as seguintes melhorias:

- Não deixamos que o dig dug se mova para trás de Fygar's que estejam a correr contra uma parede aumentando bastante o custo destes movimentos, visto que se podem virar para trás e disparar no mesmo frame, não havendo forma de reagir contra este comportamento.

- Como comportamento normal, o dig dug agora move-se diretamente para cima (ou baixo, caso não seja possível ir para cima) de um Fygar, para ser mais fácil de o matar.

```
for fygar in fygars:
    mapa.update_cost((fygar[0], fygar[1]), 10000)
    for pos in range(1, 5):
        if (
            (fygar[2] == 1)
            or ((fygar[0] - 1, fygar[1]) not in danger_zone)
        ) and (fygar[0] + pos < size[0]):
            mapa.update_cost((fygar[0] + pos, fygar[1]), 10000)
        if (
            (fygar[2] == 3)
            or ((fygar[0] + 1, fygar[1]) not in danger_zone)
        ) and (fygar[0] - pos >= 0):
            mapa.update_cost((fygar[0] - pos, fygar[1]), 10000)
```

```
if next_to_kill["name"] == "Fygar":
    if checkinbounds((next_to_kill_pos[0] + 1, next_to_kill_pos[1])):
        go_to = (next_to_kill_pos[0] + 1, next_to_kill_pos[1])
    else:
        go_to = (next_to_kill_pos[0] - 1, next_to_kill_pos[1])
else:
    go_to = next_to_kill_pos
```

Resultados

Os resultados das últimas 20 runs foram as seguintes:

Fizemos 50 runs de testes para ver os resultados do nosso agente.
Os resultados foram os seguintes:

- 63 mil pontos de média;
- Nível médio = Nível 12;
- Máximo de 113645 pontos;

O nosso agente conseguia correr com game speed 200 sem problemas, pelo que se mostra ser eficiente e rápido.

PS : Testado em computador com um I7-9750h e 16GB ram

	id	timestamp	player	level	score
	Search column...	Search column...	Search column...	Search column...	Search column...
1	1	2023-12-16 21:32:27	108969	11	57258
2	2	2023-12-16 21:33:48	108969	10	45665
3	3	2023-12-16 21:35:08	108969	12	62147
4	4	2023-12-16 21:36:45	108969	14	66573
5	5	2023-12-16 21:37:35	108969	7	31879
6	6	2023-12-16 21:39:22	108969	14	78290
7	7	2023-12-16 21:40:18	108969	10	40289
8	8	2023-12-16 21:41:44	108969	13	68723
9	9	2023-12-16 21:42:54	108969	11	48266
10	10	2023-12-16 21:43:55	108969	10	50915
11	11	2023-12-16 21:46:11	108969	14	71835
12	12	2023-12-16 21:47:22	108969	5	14605
13	13	2023-12-16 21:48:33	108969	11	47818
14	14	2023-12-16 21:50:10	108969	11	51588
15	15	2023-12-16 21:51:36	108969	10	44406
16	16	2023-12-16 21:53:08	108969	14	75275
17	17	2023-12-16 21:54:01	108969	11	61814
18	18	2023-12-16 21:55:14	108969	10	45142
19	19	2023-12-16 21:55:55	108969	9	42833
20	20	2023-12-16 21:56:43	108969	11	48720

Conclusão

Com este trabalho conseguimos aprender a construir elementos reactivos e deliberativos. Tivemos de conseguir fazer a conjunção dos dois tipos de forma a otimizar o nosso agente ao máximo.

Também pudemos consolidar os nossos conhecimentos sobre árvores de pesquisa de forma a levar o nosso agente a atingir um determinado objetivo com a maior eficiência.

Tivemos algumas dificuldades em lidar com o fogo dos Fygar's e a tentar ganhar pontos a partir do uso de pedras mas percebemos que esta não seria uma solução muito proveitosa e encaramos o problema fazendo com que o nosso agente fosse um dig dug "Assassino", uma vez que procura sempre atacar o inimigo mais próximo.