

Development of an autonomous agent for the game Tetris



Diana Siso N^omec 98607
João Borges N^omec 98155
Rodrigo Lima N^omec 98475

A heurística

A heurística foi baseada na página seguinte:

<https://codemyroad.wordpress.com/2013/04/14/tetris-ai-the-near-perfect-player/>

Iremos ter as variáveis:

- Aggregate height, altura em conjunto, quanto menor, melhor.
- Complete lines, linhas preenchidas no tabuleiro, quanto mais melhor.
- Holes, sítios no tabuleiro cobertos com uma peça na posição imediatamente em cima, quanto menos melhor
- E bumpiness, que seria sítios no tabuleiro com grandes discrepâncias de altura, ou seja, uma coluna com altura de 0 rodeada com colunas de altura 5, é calculado através do somatório da diferença entre a altura de todas as colunas adjacentes. Quanto menor melhor.

Cada um destes valores é depois multiplicado por uma variável constante e são todos somados, obtendo o valor final. Esta função encontra-se no ficheiro newBotsFuncts.py com o nome de Score.

Como processar as seguintes peças - tree_search.py

Foi usado o tree search dado nas aulas; é criada uma nova tree no código do student.py, e depois será calculada a tree através do método search. O search percorre todas as posições possíveis da peça e verifica todos os seus scores de acordo com a heurística, no final, são criados nós para cada um destes movimentos, com várias informações, como por exemplo, o score desse movimento, os comandos para o mesmo, e o tabuleiro depois deste movimento. Seguidamente, o search irá criar novos nós que são os movimentos com a seguinte peça sobre os nós previamente calculados, ou seja, outra simulação que verificará qual o caminho com menor score, para que seja feita a melhor jogada possível tendo em conta as peças seguintes. Para não ocupar tanto tempo de processamento, a cada iteração do ciclo de profundidade é retirado os nós com pior custo, pois é extremamente improvável que a soma destes seja a melhor.

Outros detalhes

- Quando o tree search procura soluções para as três seguintes peças, o search é feito apenas uma vez, após isso é colocado uma flag que indica que não é necessário calcular novamente enquanto os comandos para essas três peças forem usados.
- A quantidade de peças calculadas à frente é reduzido quando a velocidade do jogo chega a um certo patamar.
- O tViewer.py devolve uma image em tempo real do que está a ocorrer no jogo através do terminal, ou seja, substitui o viewer.py.
- Infelizmente, o nosso bot consegue como resultado máximo 1708 pontos, tendo obtido muito poucas vezes valores acima de 1000 pontos, isto devido a pura sorte; acreditamos que o bot apenas necessita de uma resposta mais rápida, ou seja, um pequeno refactoring no código. Mesmo assim, o bot apresenta uma grande consistência, sendo que raramente faz menos de 100 pontos, e maior parte das vezes faz no mínimo 300 pontos, isto devido tanto à heurística como à previsão de jogadas com o tree search.
- O botFuncts.py foi a primeira implementação feita para a primeira entrega. A heurística foi mudada por completo, originando o newBotFuncts.py.