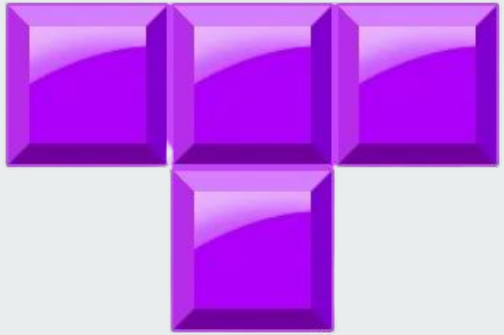
universidade de aveiro

Tetris AI

Projeto de Inteligência Artificial
Departamento de Telecomunicações e Informática



João Fernandes (93460)
João Morais (93288)

Algoritmo

- Criámos um algoritmo análogo ao *treesearch* apresentado na aula, mas adaptado ao Tetris.
- Pesquisámos a heurística mais adequada ao Tetris.
- Para mover a peça para ser analisada pela heurística usámos o ficheiro *shapes.py* que nos foi facultado..
- Quando é escolhida a melhor rotação e a melhor posição usamos as *keys* para mexer a peça até ao seu destino.

Heurística

- As pesquisas efetuadas possibilitaram encontrar a heurística que melhor se adequa ao nosso projeto. Para a heurística baseámos-nos neste site: https://codemyroad.wordpress.com/2013/04/14/tetris-ai-the-near-perfect-player/
- Optámos por uma heurística com os seguintes fatores:
  - **Altura agregada** (soma das alturas de cada coluna)
  - **Número de buracos** (sendo buraco um espaço vazio que tem um espaço ocupado em cima)
  - **Linhas completas** (linhas que podem ficar preenchidas quando a peça é colocada)
  - **Bumpiness** (subtração da altura da coluna atual com a coluna seguinte)
  - Colocámos ainda outro fator (**Perto da parede**) que valoriza se a peça estiver perto de uma parede

- A heurística é calculada da seguinte maneira:

$$score = -0.510066 * AlturaAgregada + 0.760666 * Linhascompletas - 0.45 * Numerodeburacos - 0.184483 * Bumpiness + 0.09 * Pertodaparede$$

Resultados obtidos

- Usando esta implementação, conseguimos com que o AI alcançasse o máximo 1000 pontos. Chegámos a um ponto mais estável por volta dos 600/700 pontos onde o AI atingia com bastante frequência.
- Tivemos de valorizar o facto de a peça estar junto às paredes porque o algoritmo começa a amontoar peças no centro do tabuleiro

Conclusão

- No início tentámos implementar a *treesearch* fornecida na aula mas esta mostrou-se ser ineficiente. Acabamos por criar o nosso próprio algoritmo sem recorrer ao treesearch.
- Para melhorar o algoritmo era preciso fazer com que este avaliasse eficientemente a próxima peça sem que isso afetasse o tempo de execução. Ao tentar avaliar a próxima peça revelou-se ser um processo que afetava drasticamente o tempo de execução pois cada peça passava a ter significativamente mais situações para serem avaliadas.