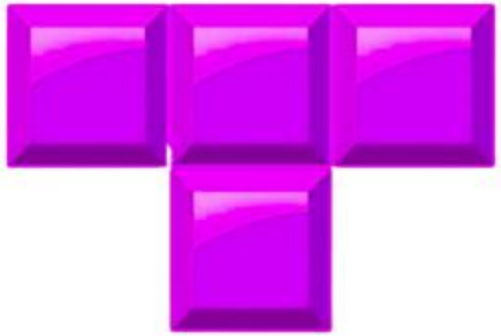
universidade de aveiro

Desenvolvimento de um agente autónomo para o jogo **tetris**



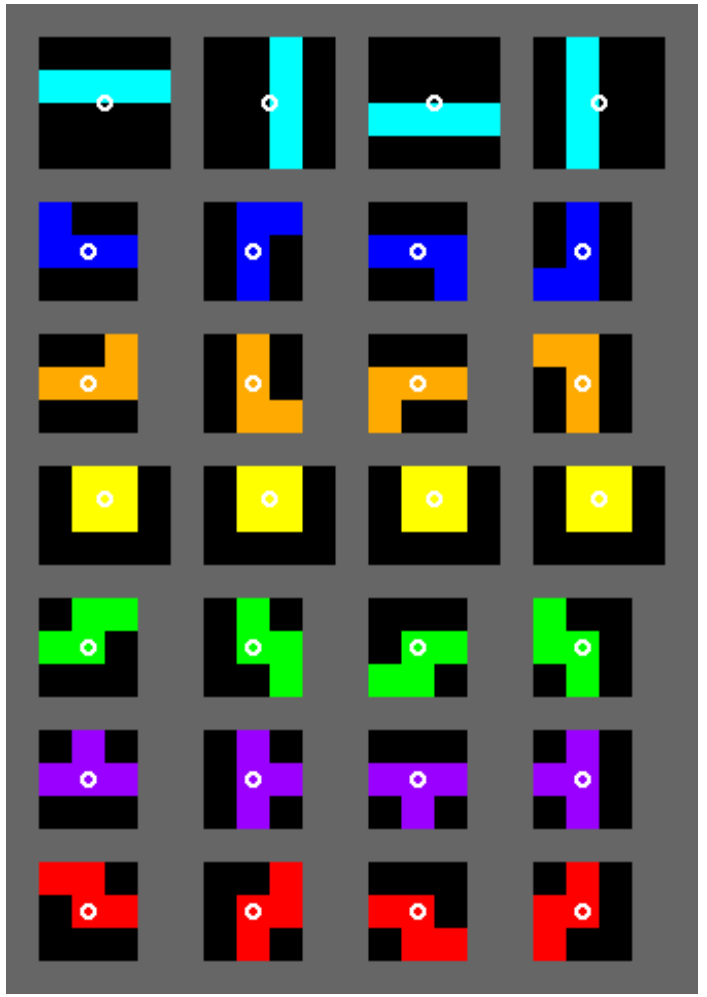
Inteligência Artificial

Bruno Lemos – 98221 João Amaral - 98373

Engenharia de Computadores e Informática

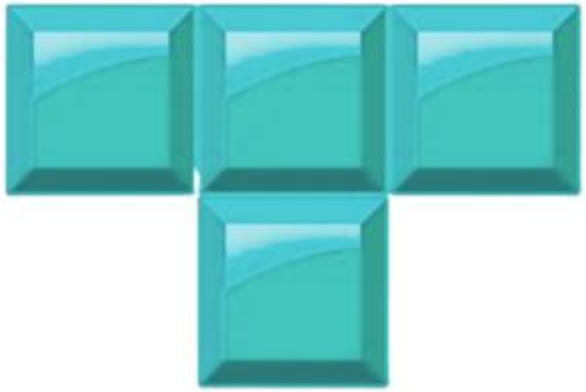
Possibilidades para uma peça

- Desenvolvemos uma função genérica que trata da identificação da peça atual, incluindo as suas possíveis rotações
- A partir do estado atual do mapa, calculamos todas as possibilidades de colocar a peça no mapa (simulando a sua queda para cada *x*) e guardamos todas essas opções numa lista.
- Por fim, submetemos todas essas possibilidades a várias funções de avaliação tendo em conta fatores de penalização e favorecimento

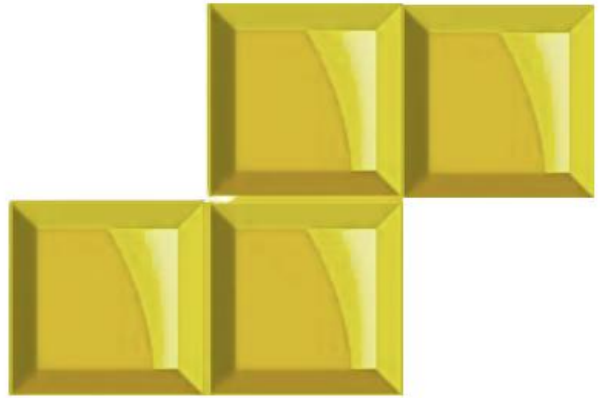


Possibilidades para a peça seguinte

- Inicialmente calculamos usamos o algoritmo de calculo de todas as possibilidades para a peça atual.
- De todas essas possibilidades, guardamos apenas metade das melhores jogadas para efeitos de otimização de tempo
- Com base nesses novos mapas que incluem a peça atual, recalculamos as possibilidades de jogar a peça seguinte em todos os seus possiveis estados.
- No fim, avalia todas essas possibilidades com as duas peças, escolhendo a melhor jogada através das mesmas avaliaçoes, ficando no fim apenas com a posiçao ideal para jogar a peça atual do mapa.
- Este algoritmo é novamente calculado quando acabamos de jogar a peça atual e aparecem novas, dado que verificamos que é mais eficiente do que jogar de duas em duas peças.

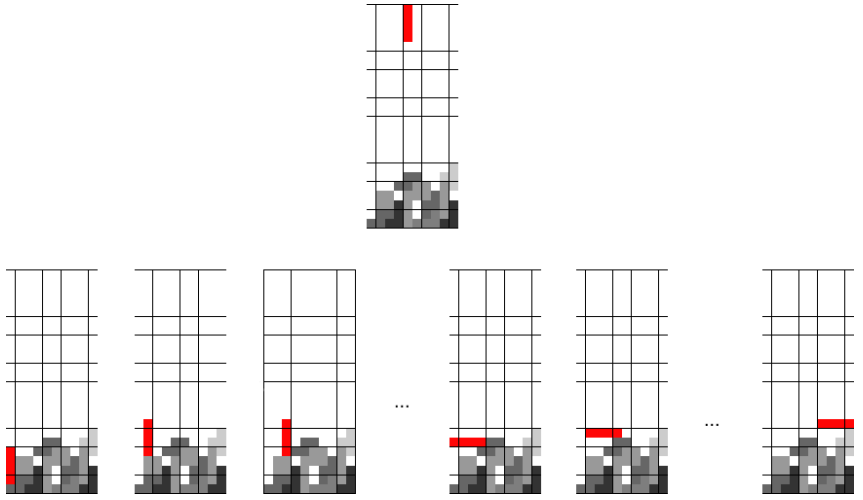


Current piece



Next piece

Avaliações das jogadas



- Dado um mapa, reunimos um conjunto de fatores que favorecem e penalizam uma dada jogada.
- **Altura total e máxima do mapa**: favorece jogadas onde a peça é colocada mais abaixo no mapa;
- **Relevo**: favorece jogadas que não tornem o mapa muito irregular em termos de altos e baixos;
- **Número de buracos**: penaliza jogadas que criem buracos;
- **Número de linhas completas**: favorece jogadas que façam um numero maior de linhas completas;
- **Número de colunas vazias**: penaliza jogadas que criem colunas altas de peças;
- Após submetermos um mapa a estas avaliações, escolhemos o mapa com o menor valor total, indicando que possivelmente se trata da melhor jogada.

Otimização do algoritmo

- Para que o nosso agente faça o maior numero de pontos possivel, é necessário que este calcule as possibilidades todas no menor intervalo de tempo para conseguir tomar decisões e jogar as peças. Para isso, implementamos solucões mais otimizadas para diminuir esse tempo de calculo.
- Uso de listas bidimensionais binárias (matriz) para determinar a ocupação de uma determinada coordenada do mapa de jogo.
- O cálculo das avaliações foi otimizado para esta nova representação matricial, reduzindo o numero de ciclos e operações de pesquisa.
- Uso de lookup tables para aceder a informação estática regularmente.
- Reduzimos o cálculo dos mapas possíveis para a metade quando o algortimo é de duas peças, em vez de calcular todas as possibilidades para duas peças.
- Alteração do comportamento do agente com base na velocidade atual e a altura do mapa, alternando entre o cálculo das possibilidades apenas para uma peça e para duas peças. Isto é, quando o jogo vai a alta velocidade e a altura do mapa não deixa concluir o calculo das possibilidades da peça atual e seguinte, alteramos apenas para calcular para a peça atual, o que é signficativamente mais rápido, até conseguir diminuir novamente a altura do mapa.