Sokoban

-Agente Inteligente

- Inteligência Artificial
- Licenciatura em Engenharia Informática 2020/2021

- Diogo Moreira 93127
- Fábio Carmelino 93406

universidade de aveiro

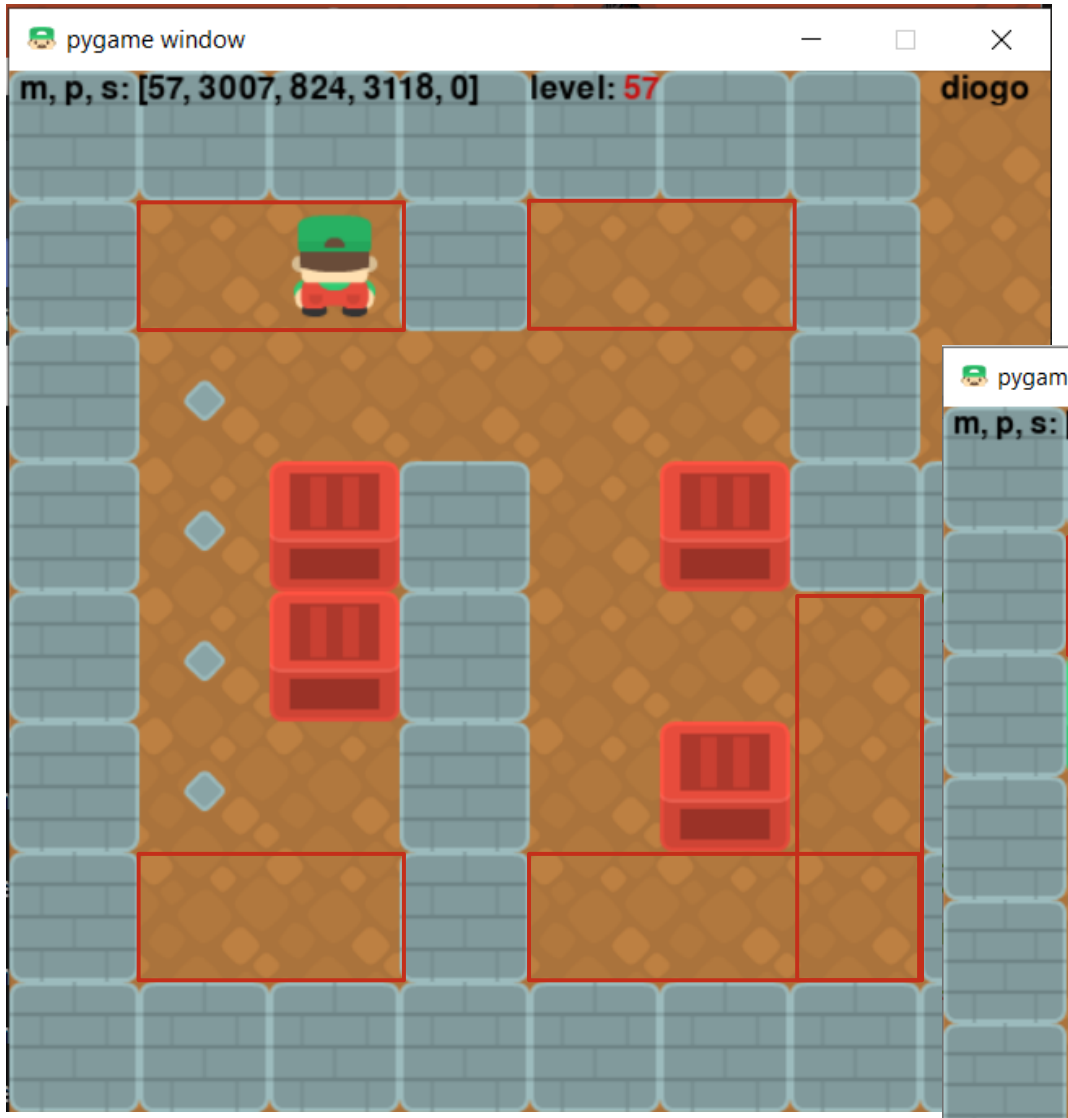
Algoritmo

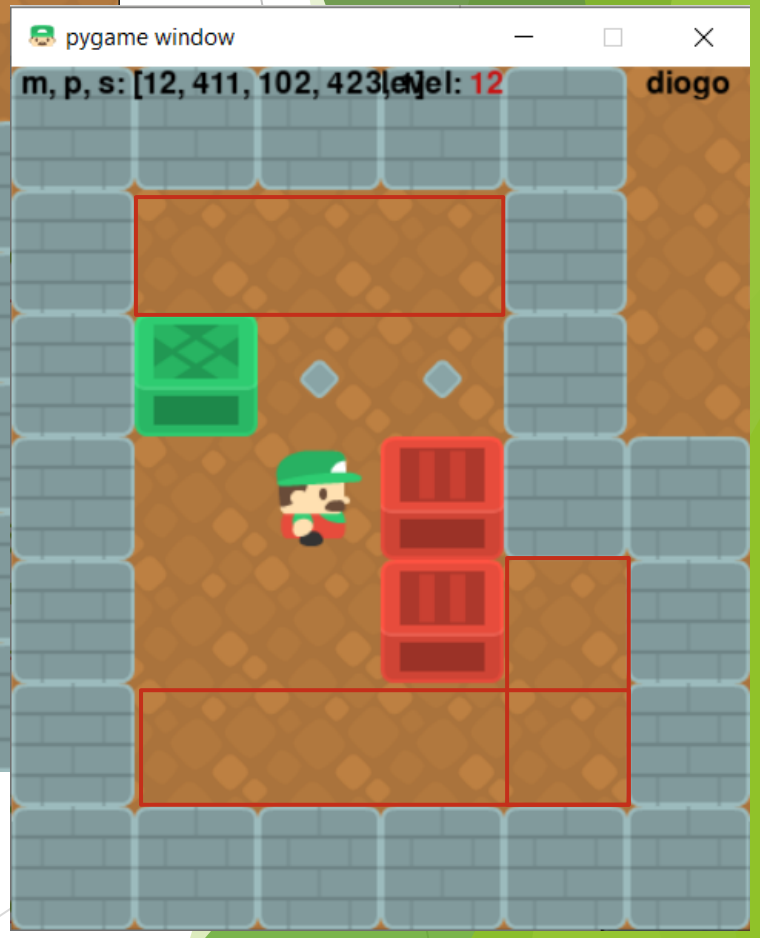
- Para este projeto, dividimos a pesquisa para as caixas e a pesquisa para o keeper com o objetivo de tornar a pesquisa mais rápida e a lógica do programa mais simples.
- Para as caixas é feita uma pesquisa na qual tentamos chegar ao resultado, ou seja, colocar todas as caixas nos objetivos, ao simular a movimentações de caixas, e excluindo as posições que levariam a um deadlock.
- Ao Keeper, para cada movimentação de caixa, fornecemos a posição onde ele está e a posição da caixa que quer empurrar realizando a pesquisa para confirmar se é possível ele empurrar a caixa.
- Para a heurística das caixas fazemos a soma das distâncias de Manhattan entre cada caixa que não esta num objetivo e os objetivos sem caixa.
- O custo das caixas é a distância de Manhattan entre a posição Keeper atual e a próxima.
- O custo do Keeper é 1 representando cada movimento dele e a sua heurística é a distância de Manhattan entre ele e o seu objetivo final.

Simple Deadlocks

Calculado no início do nível e representa os cantos e outras posições no mapa para as quais empurrar uma caixa causa um deadlock.

Todas as posições marcadas no mapa são consideradas como simple deadlock porque levam á impossibilidade de resolução do nível.

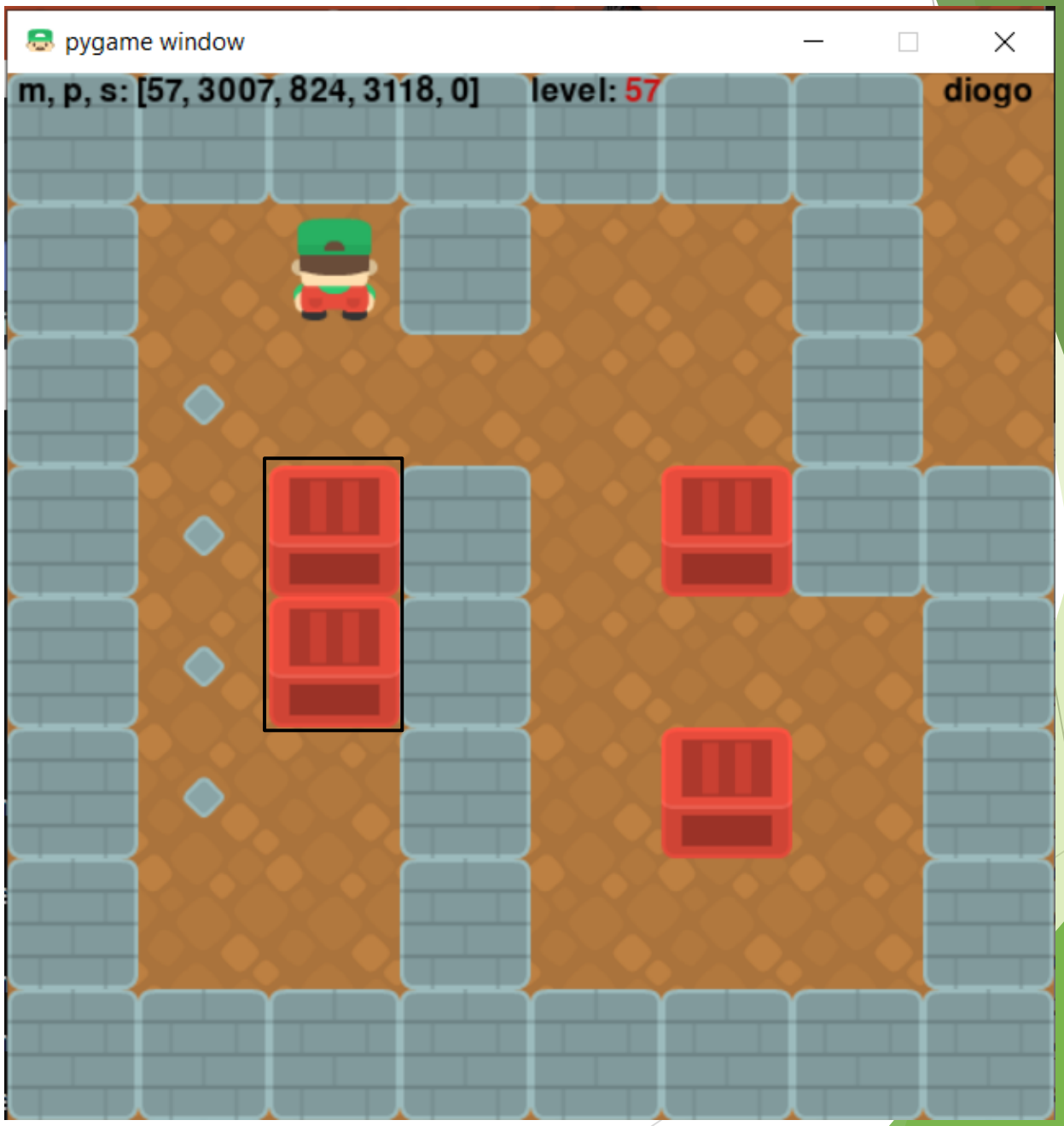




Freeze Deadlocks

Calculado a cada ação e acontece quando empurramos uma caixa e ficamos com uma ou mais caixas em deadlock.

As duas caixas marcadas no mapa estão num freeze deadlock porque as caixas bloqueiam-se uma a outra no eixo do y (vertical) e estão bloqueadas no eixo do x (horizontal) por uma parede sendo impossível resolver o nível.



Resultados

- Foi possível ver um progresso constante no agente sempre que eram implementadas novas formas de prevenção de deadlock ou quando tornávamos o código mais eficiente por exemplo ,ao trocar listas de uma dimensão([(x,y),(x2,y2),…]) para lista de duas dimensões ([x][y]=True) foi possível ver uma grande melhoria.
- Os maiores obstáculos encontrados foram nos níveis 21,57, 68 e 118 sendo que o último ainda se encontra presente apesar das várias tentativas de otimização.
- Para a resolução do nível 21 implementamos uma lista com os estados pelos quais já tínhamos passado.
- Para a resolução do nível 57 foi a implementação de simple deadlocks e otimização do código em geral
- Para o nível 68 (foi o maior obstáculo) tentamos implementar freeze deadlocks e várias melhorias de performance (arrays de duas dimensões e bisect para o sort) o que tornou o agente muito mais rápido na resolução dos níveis e finalmente possibilitou ultrapassar este nível.

Conclusão

Neste trabalho, ficamos satisfeitos com a grande parte do trabalho que realizámos, pois foi possível vermos um constante progresso no desempenho do nosso agente.

Apesar da evolução constante do nosso agente ainda ficou por fazer uma implementação que conseguisse passar o nível 118 (chegamos a uma solução mas não é rápido o suficiente para terminar o nível antes do tempo acabar).