U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

André Sousa (up202005277@fe.up.pt)

Gonçalo Pinto (up202004907@fe.up.pt)

Pedro Fonseca (up202008307@fe.up.pt)

Block Escape

AI First Project

Software used and related work

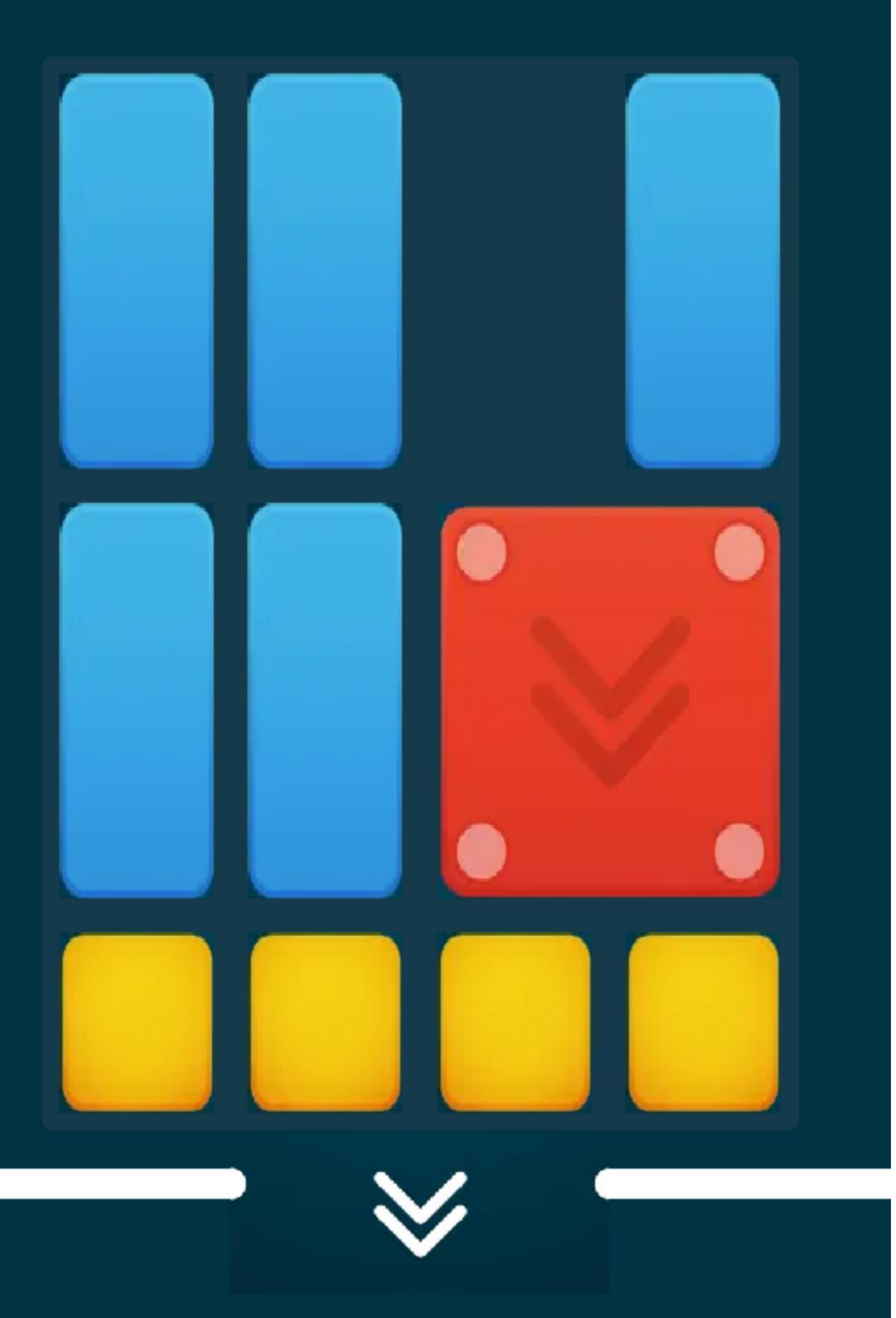
- Stuart Russell, Peter Norvig; Artificial intelligence. ISBN: 978-0-13-207148-2
- Richard S. Sutton; Reinforcement learning. ISBN: 978-0-262-03924-6
- Stuart Russel, Peter Norvig; Artificial Intelligence: A modern Approach.
- https://www.transum.org/Maths/Investigation/CarPark/
- PyCharm Professional
- Python 3.9
- Git

Specification of the work

- **Goal:** take the red block to the exit, while navigating around obstacles on a limited grid.

- **Project Objectives:**
  - Implement search algorithms to efficiently calculate an efficient path to the exit.
  - Test and analyze various algorithms and heuristics to determine the best approach for the game.

- **Algorithms:** breadth-first search, depth-first search, iterative deepening, greedy search, A* algorithm and weighted A*



Formulation of the problem as a search problem

Objective Test: The red block is in the exit position.

State Representation: An object Puzzle with defined dimensions and a list of blocks.

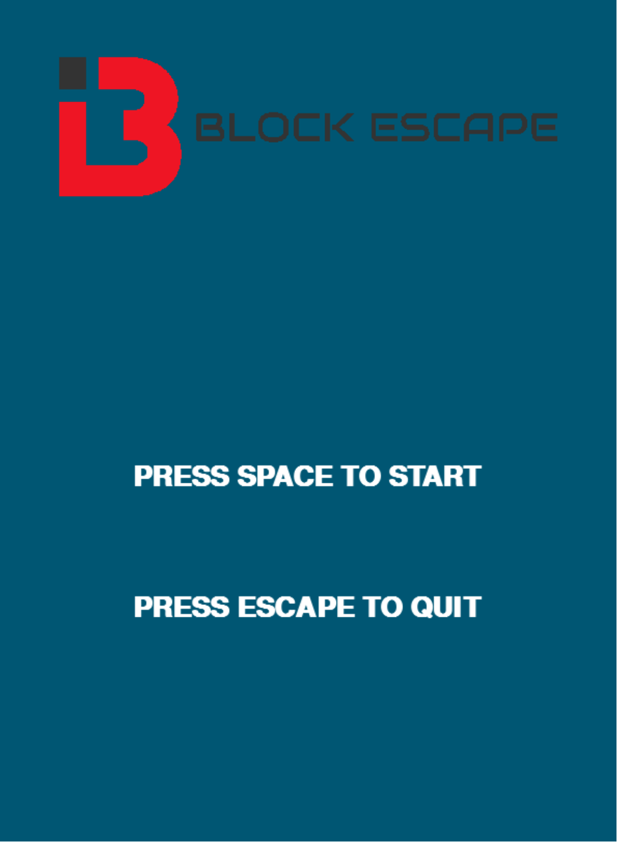
Initial State: A board filled with blocks and the red block is blocked.

Operators: move: Move a block to an empty space in the game board.

- Preconditions: Enough space adjacent to the current position and the coordinate is valid on the game board.
- Cost: 1 move.
- Effect: Update the position of the block with the new position.

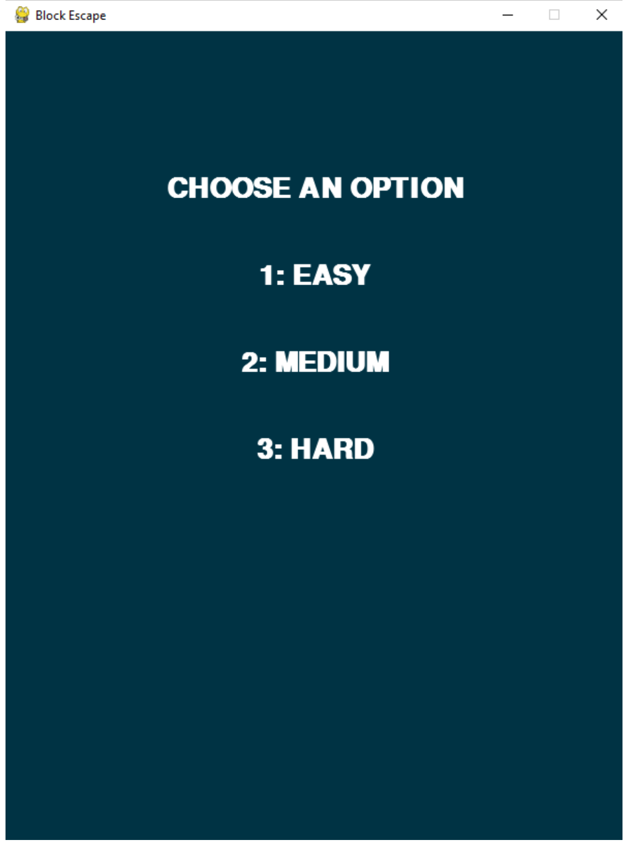
Heuristics:

- H1: Distance from the red block to the exit.
- H2: Weighted sum of the number of obstacles between the red block and the exit.
- H3: The largest contiguous empty space near the red block.
- H4: Prioritize moves that keep the red block close to the edges of the game board.
- H5: Always ensure that the red block has at least one valid move option available.
- H6: Prioritize fitting pieces along the edges of the game board from largest to smallest.
- H7: Prioritize moving the largest pieces on the game board first.
- H8: Prioritize moving the smallest pieces on the game board first.


BLOCK ESCAPE

PRESS SPACE TO START

PRESS ESCAPE TO QUIT

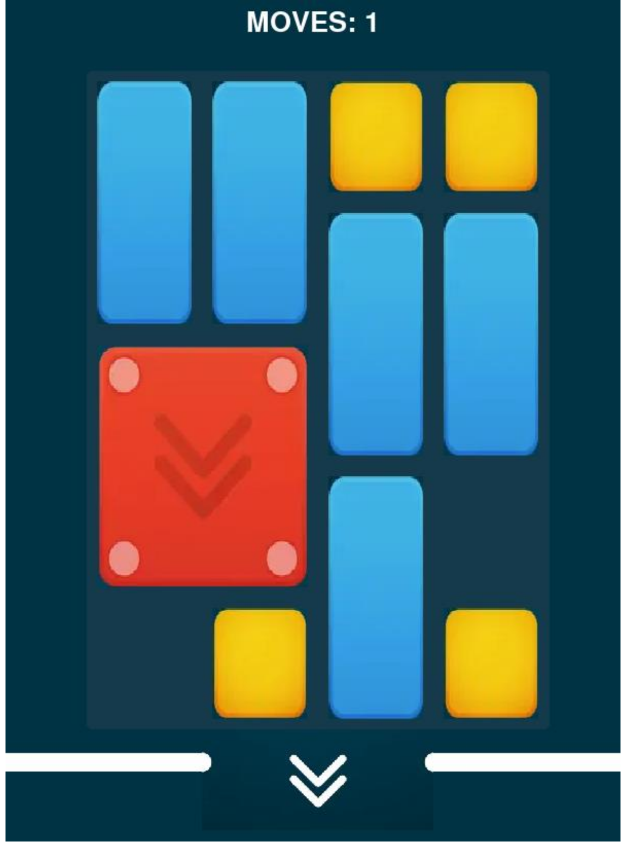

Block Escape

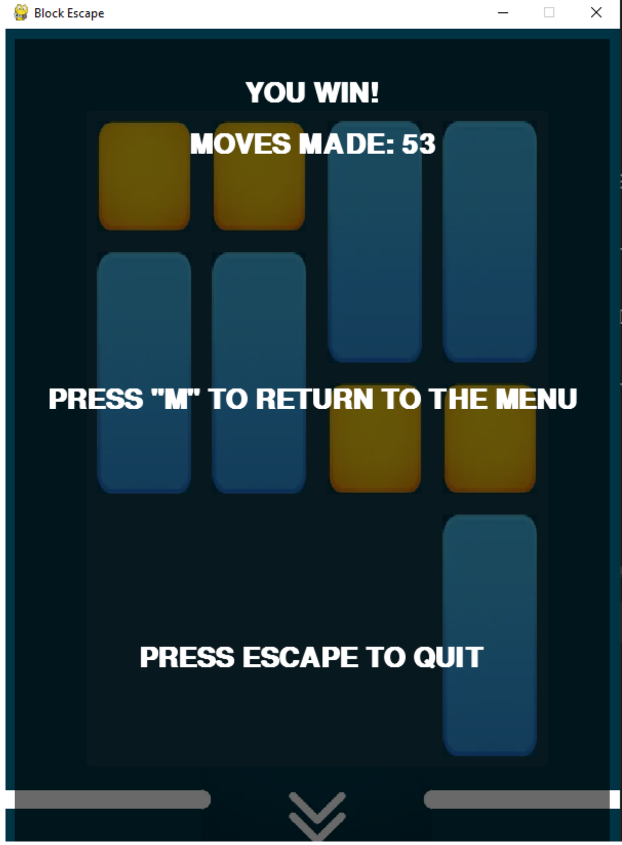
CHOOSE AN OPTION

1: EASY

2: MEDIUM

3: HARD


MOVES: 1


Block Escape

YOU WIN!

MOVES MADE: 53

PRESS "M" TO RETURN TO THE MENU

PRESS ESCAPE TO QUIT

Implementation work already carried out

- Menu that allows users to start or quit the game
- Difficulty level selection after starting the game
- Graphical User Interface (GUI) for piece movements
- Game over detection to determine when the game ends
- Total moves counter to keep track of the number of moves made

Heuristics and operators

Heuristics:

- **H1:** Euclidean distance between the red block and the exit
- **H2:** Weighted sum of the obstacles between the red block and the exit
- **H3:** Maximize the amount of contiguous empty spaces near the red block
- **H4:** Prioritize moves that keep the red block close to the edges
- **H5:** Prioritize movements in which the red block has at least one possible movement
- **H6:** Prioritize fitting blocks along the edges of the game board from largest to smallest
- **H7:** Prioritize moving the largest blocks first
- **H8:** Prioritize moving the smallest blocks first

Operators:

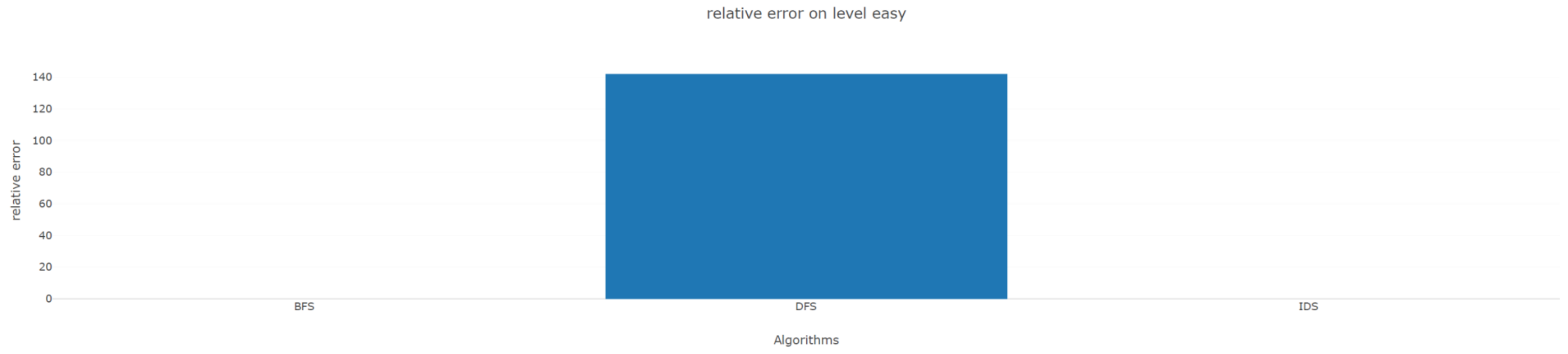
- move: Move a block (up, down, right or left) to an adjacent empty space in the game board.
  - Preconditions: Enough space adjacent to the current position and the coordinates are valid on the game board.
  - Cost: 1 move.
  - Effect: Update the position of the block.

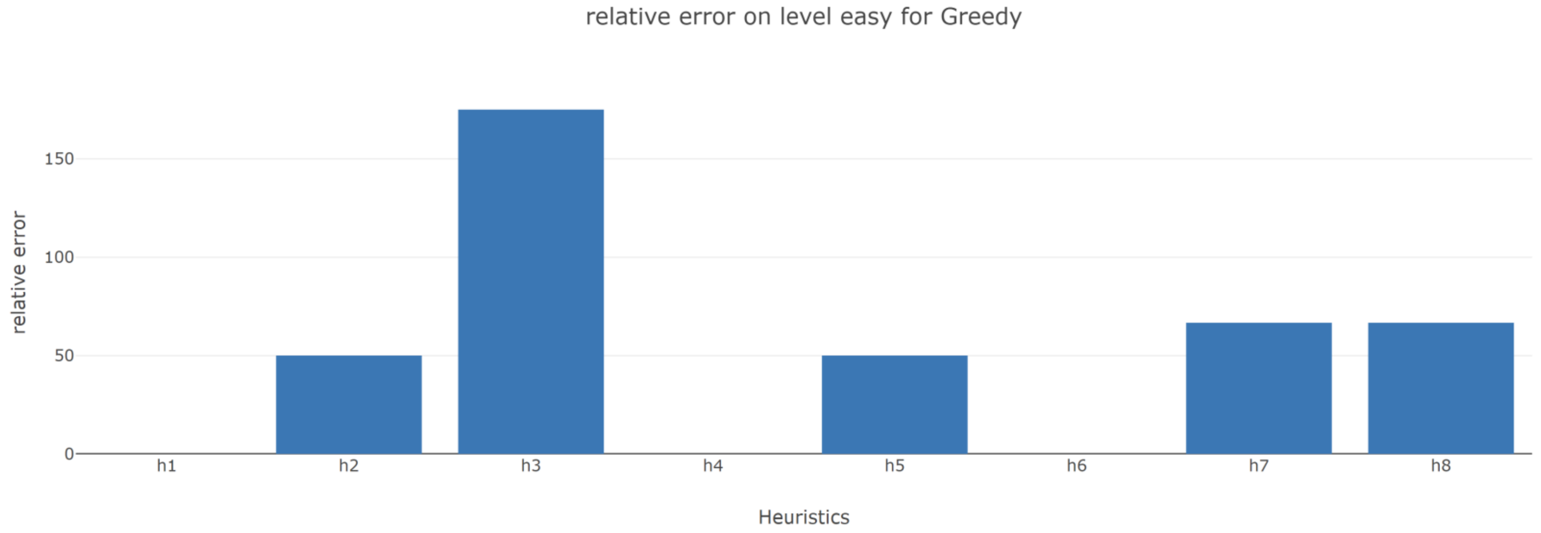
Implemented algorithms

- Breadth First Search
- Depth First Search
- Depth Limited Search
- Iterative Deepening Search
- Greedy Search
- A* search
- Weighted A* search

Experimental Results

Full report here





Conclusions

- In terms of speed, BFS was found to be slower compared to the other algorithms.
- Greedy search, on the other hand, performed differently due to its lack of a cost function.
- Prioritizing moves that keep the red block close to the edges of the board was found to be the best heuristic overall.
- All the heuristics provided an optimal solution in the A* algorithm.
- IDS was slower than DFS but used fewer nodes, making it a good choice when memory is a concern.
- The performance of search-based games is heavily influenced by the choice of algorithm and heuristic used.
- Our results suggest that BFS is not a good choice for this game.
- Greedy search was interesting but did not perform as well as expected.
- Prioritizing moves that keep the red block close to the edges of the board provided the best results overall.
- All heuristics provided optimal solutions in the A* algorithm.
- Overall, our findings highlight the importance of carefully selecting the appropriate search algorithm and heuristic for a given search-based game.