RL - BOMBERMAN

Intelligent Systems II

2023

Daniela Dias, 98039
Filipe Gonçalves, 98083
Gonçalo Machado, 98359
João Borges, 98155
Miguel Beirão, 98157

TABLE OF CONTENTS

ARTIFICIAL INTELLIGENCE (AI)

01.

INTRODUCTION

Brief contextualization and introduction

INTRODUCTION

Bomberman is a classic multiplayer arcade game where players navigate maze-like arenas, strategically placing bombs to eliminate opponents and obstacles while avoiding being caught in the explosions, aiming to be the last one standing.

The proposed work consists in the **development of a reinforcement learning agent** that based on rewards and punishments tries to make the most points possible.

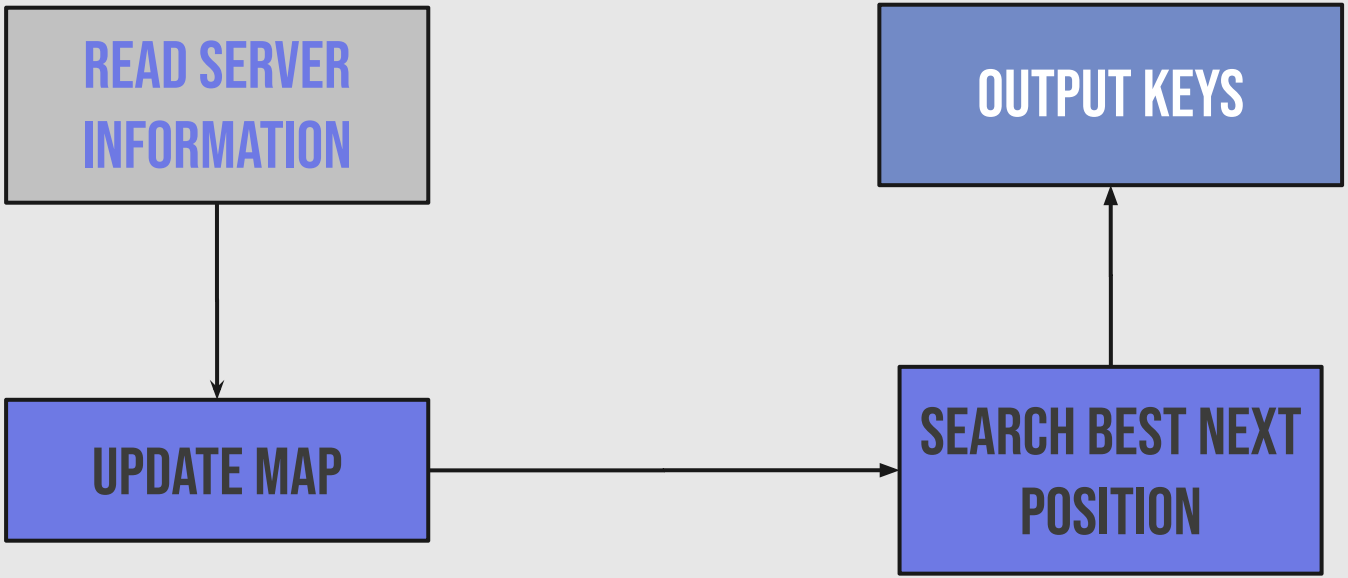
/[AI]

02.

METHODS

Explanation of the approaches used

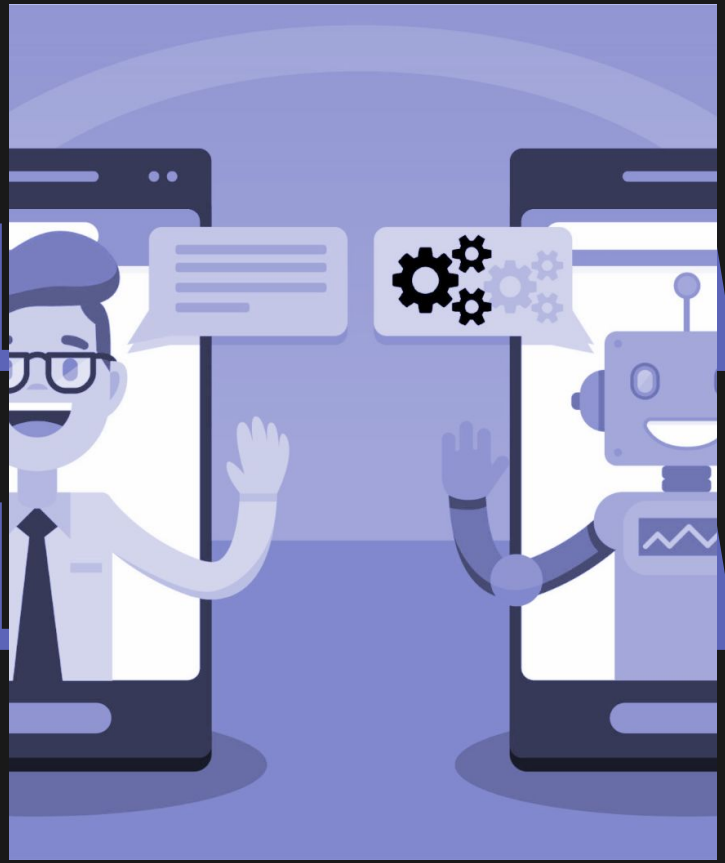
BASE METHOD


READ SERVER INFORMATION
OUTPUT KEYS
UPDATE MAP
SEARCH BEST NEXT POSITION

METHOD 1

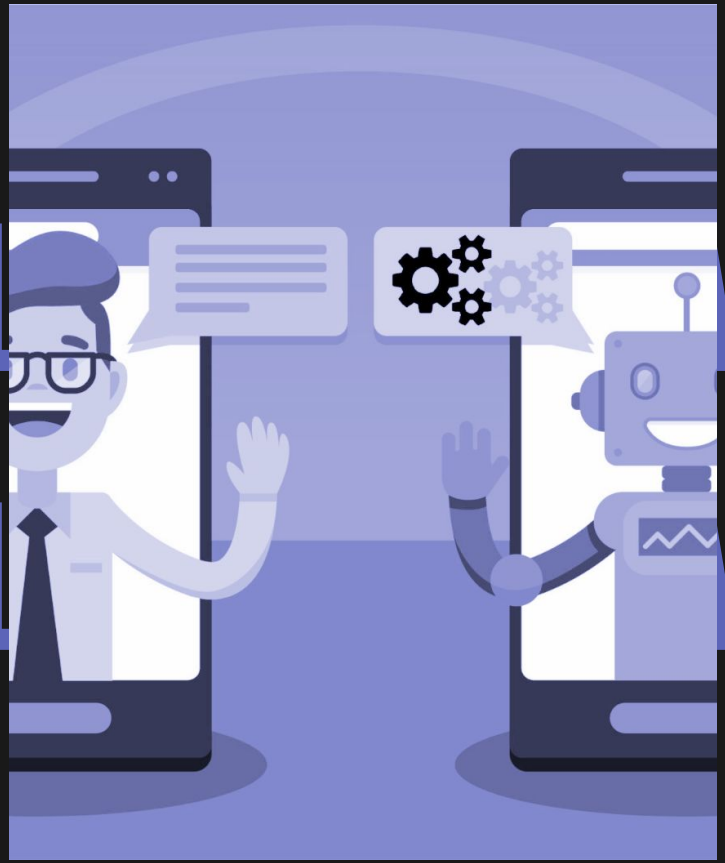
MAIN FEATURES

- Reward System
- Map Updater
- Enemy Best Position to Kill Calculator
- Dynamic Exit
- Greedy Search

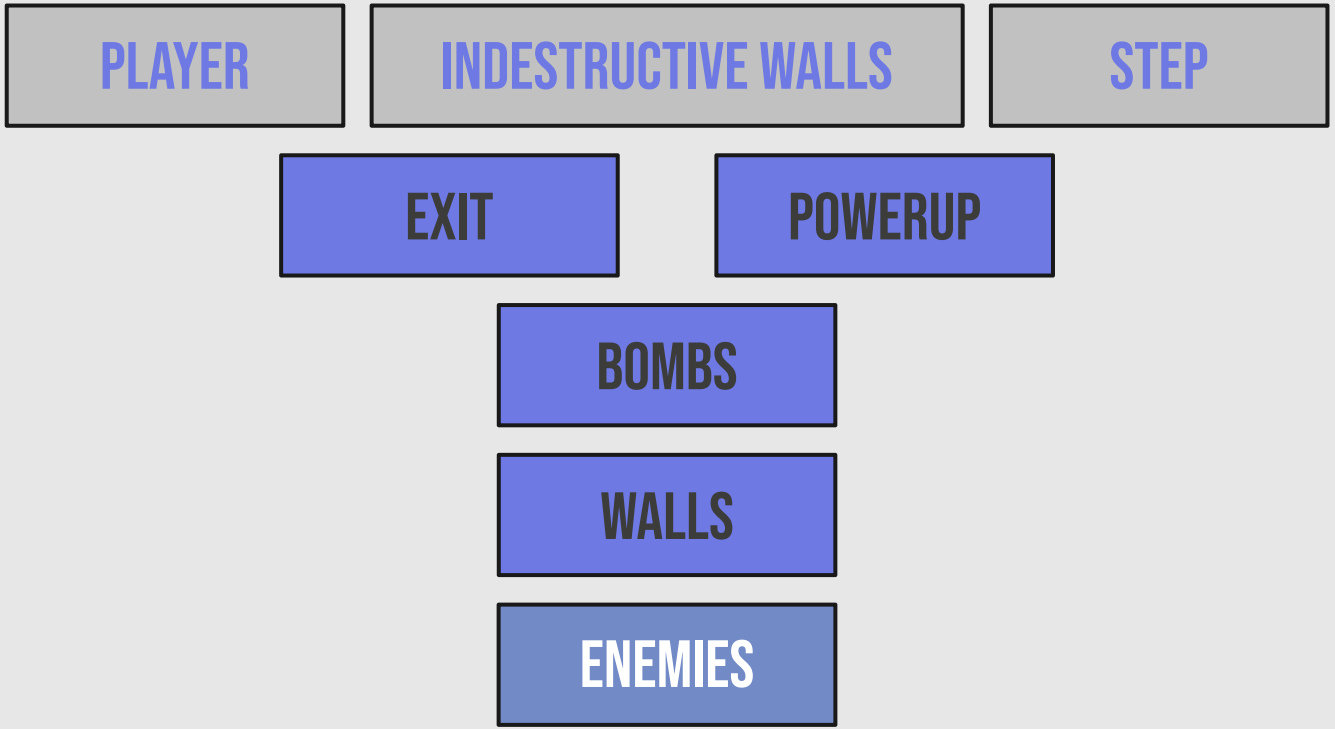


METHOD 1 - REWARDS

- **Indestructive Wall** = -100
- **Destructive Wall** = 6
- **Destructive Wall Exit** = 4.5
- **Enemy Area** = -8
- **Enemy Position** = -20
- **Enemy Bomb** = 20
- **Bomb Drop Position** = -10
- **Bomb Damage** = -10
- **Step** = -0.01
- **Player** = -0.02
- **Powerup** = 10
- **Exit** = 50



METHOD 1 - MAP UPDATER



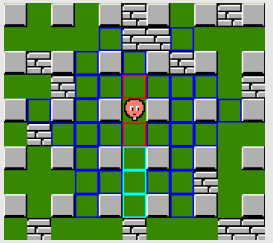
METHOD 1 - ENEMY

DIRECTION

- Calculate direction based on previous movement
- Place best bomb position in front of next movement

RADIUS

- Create circumference with the enemy position as center
- Place descending values the far away it is from the center



ARTIFICIAL INTELLIGENCE (AI)

METHOD 1 - DYNAMIC EXIT

METHOD 1 - GREEDY SEARCH

ALGORITHM

- Search tries to find the best node until the current node opened has depth equal or higher than sixteen
- If the reward of the current node is positive, then the search is concluded
- If the position resulting of the action was already on any ancestor of the node, then the node is not opened
- If the position resulting of the action was already in any nodes already opened but had less reward, then the node is not opened
- If the position resulting of the action is not a valid position, then the node is not opened
- If the total reward, based on the current opened node reward and the next action reward is very low, then the node is not opened
- If after the loop there was no good node found, then the next Bomberman position will be the node with the best reward

METHOD 1 - LIMITATIONS

LIMITED SEARCH

- Might get stuck in a loop due to rapid direction variation of enemies
- Can get stuck if it doesn't find any walls or enemies far away

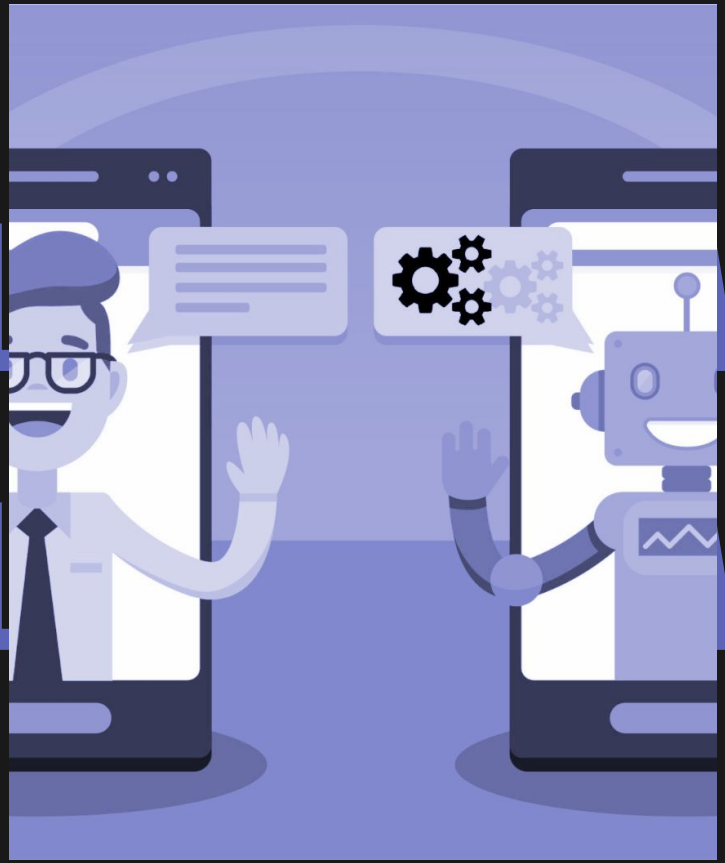
LACK OF PREDICTION

- Might corner itself between a bomb and an enemy

METHOD 2

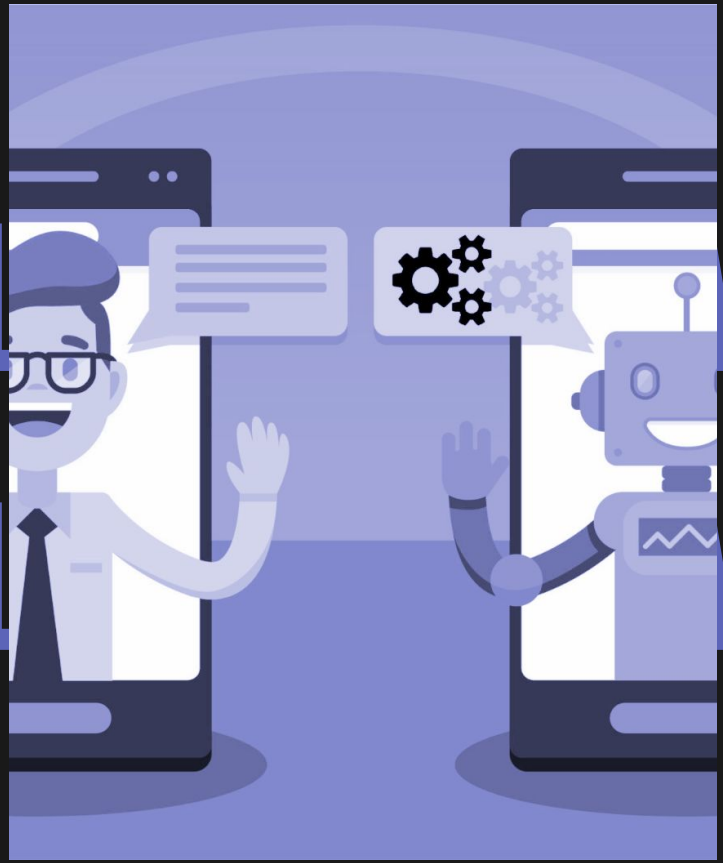
MAIN FEATURES

- Reward System
- Dynamic Reward
- Enemy Best Position to Kill Calculator
- Adaptive Reward for Enemy Types
- Breadth-First Search



METHOD 2 - REWARDS

- **Valid Move** = -0.01
- **Indestructive Wall** = -100
- **Damage** = -55
- **Reward** = 30
- **Powerup** = 20
- **Exit** = 50
- **Player** = -10



METHOD 2 - WALL REWARDS

For each destructible wall, we retrieve its position and update the game map with the negative reward **Indestructible Wall**, indicating the presence of a wall that can't be crossed.

If we don't need to destroy more walls (i.e. **less than 8 walls and the exit was found**), we don't update the area around the destructible wall.

Next, **we check if the wall will be destroyed by a bomb placed by the Bomberman**. For this, we take into account the wall coordinates and the list with the information of all bombs.

- If the wall is targeted by a bomb, we continue to the next wall without updating the area around it.

METHOD 2 - WALL REWARDS

REWARD AROUND THE WALL

If the wall isn't targeted by a bomb, we update the area around the destructible wall.

- We consider a **Propagate Range (= 20)** and calculate all possible positions based on the current range that aren't an indestructible wall, updating the game map array.
- **We consider a decay factor**, which starts at 1 and increases by 1 for each position that can't be updated due to an indestructible wall. This ensures that the reward diminishes as we find more indestructible walls between the Bomberman and the destructible wall.
- The reward value assigned to each position depends on the distance from the wall and is inversely proportional to the square of the distance (**Reward / current_range**2) / decay_factor**).

METHOD 2 - ENEMY REWARDS

For each enemy, we retrieve its position and name. We set the value for **speed and wall pass attributes** based on the enemy's name, which determine the enemy's movement behavior.

- We update the game map at the enemy’s position with the negative reward **Damage**.
- If the enemy is currently moving, we calculate the enemy's movement direction.
- If there is no information about the enemy's direction or the enemy is not moving, we iterate over all possible positions around the enemy and update the game map array with **Damage** (if the position isn’t an indestructible wall).

METHOD 2 - ENEMY REWARDS

MOVEMENT SIMULATION

- With the enemy's movement direction, we consider the enemy's speed, calculate the next positions (number of positions = speed), and update the game map with negative reward **Damage**.

- If one of the next positions is an indestructible wall and the enemy doesn’t have wall pass, we don’t consider the following positions.

METHOD 2 - ENEMY REWARDS

BOMB SIMULATION

- Next, we simulate the presence of a bomb in the direction of the enemy. We calculate possible bomb positions **based on the bomb's radius and timeout**, and we update the game map with a reward value if the position isn't an indestructible wall.
- The reward value depends on the distance from the enemy and is inversely proportional to it (**Reward / distance**).
- If the position is an indestructible wall and the enemy doesn't have wall pass, we only add **Reward / distance** to the next position (if it isn't an indestructible wall) and stop updating the game map.

METHOD 2 - ENEMY REWARDS

PROPAGATION OF REWARD

- Finally, we propagate the enemy's influence in the game map with some reward. This was motivated by **our limited search which can't cover the full map** and may struggle to find enemies that are too far away.
- **We propagate a slight reward in the enemy's movement direction and opposite direction**. We also consider the positions adjacent to the enemy's movement direction and update the game map array with a slight reward value.
- These rewards diminish as the distance from the enemy increases.

METHOD 2 - BOMB REWARDS

BOMB EXPLOSION

For each bomb, we retrieve its position, timeout, and radius. We update the game map at the bomb's position with the negative reward **Damage**, indicating the presence of the bomb.

For all relative positions around the bomb, we range from 1 to the bomb's radius and we update the game map at each position.

- The value assigned to each position depends on the bomb's countdown and is proportional to the distance from the bomb. **The lesser the countdown**, the worse the reward. **The closer the position is to the bomb**, the worse the reward.

  (4 / bomb_countdown) * **Damage** / (current_range + 1))

METHOD 2 - BOMB REWARDS

BOMB SAFE POSITIONS

We also consider **safe positions around the bomb**, in other words, the positions adjacent to the positions covered by the explosion.

If the safe position isn't occupied by the indestructible wall, we update the game map with a reward value equal to **Reward / 2**.

- This encourages the agent to choose safe positions around the bombs.

METHOD 2 - BREADTH-FIRST SEARCH

ALGORITHM

- Search tries to find the best node (i.e. best reward) by using different actions until:
  - The current node opened has depth higher than 12.
  - We have found an ideal reward (close reward or great reward).

  Close Reward:
  total_reward > 0 and reward >= **Reward** - 2 and next_node.depth <= 2

  Great Reward:
  total_reward >= **Reward** + 10

METHOD 2 - BREADTH-FIRST SEARCH

ALGORITHM

- A node isn't opened if:
    - The position resulting of the action is already on any ancestor of the node.
    - The position resulting of the action is already in the opened nodes but had less reward.
    - The position resulting of the action is a indestructible wall.
    - **The total reward is positive but the current reward is negative**.
- If the reward of the current node is ideal, then the search is concluded.
- If after the loop there isn't an ideal node, then the next Bomberman position will be the node with the best reward.

METHOD 2 - LIMITATIONS

LIMITED SEARCH

- Might loop due to new rewards being found at the max depth.
- Might not find enemies when the distance is too great.

LACK OF PREDICTION

- Might corner itself between a bomb and an enemy.

LACK OF DISTINCTION

- Enemies and walls are given the same reward.

METHOD 2 - COUNTER MEASURES

LOOP PROBLEMS

- Decrease the reward at the previous 7 Bomberman's positions.
- **reward += Reinforcement.Player * (7 - prev_bomber_pos.index(next_position))**

- Propagate a slight reward in the enemy's movement direction and opposite direction.
- Propagate a slight reward in the positions adjacent to the enemy's direction.

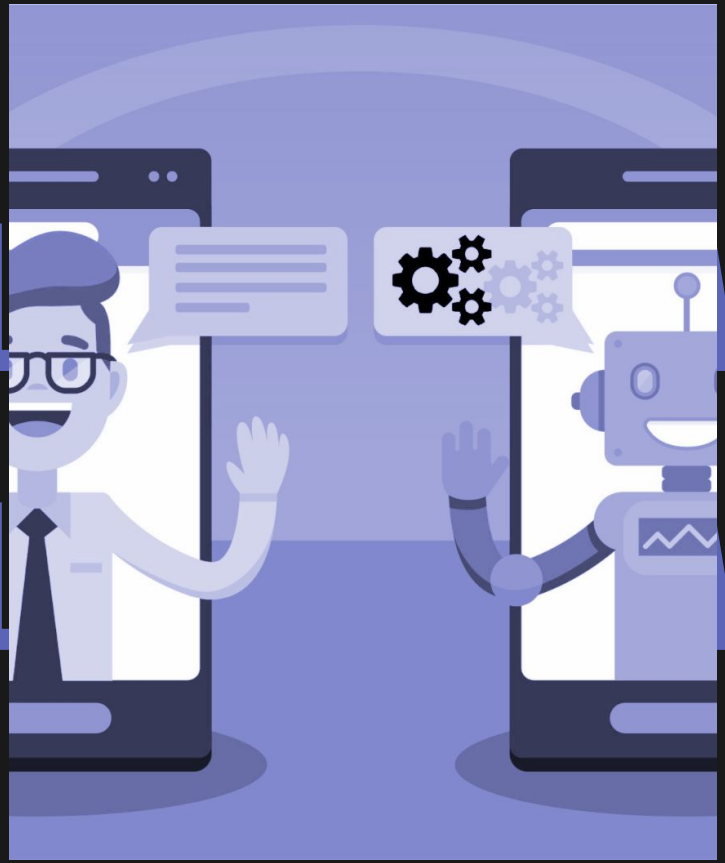
ENEMY IS FAR AWAY

ARTIFICIAL INTELLIGENCE (AI)

METHOD 3

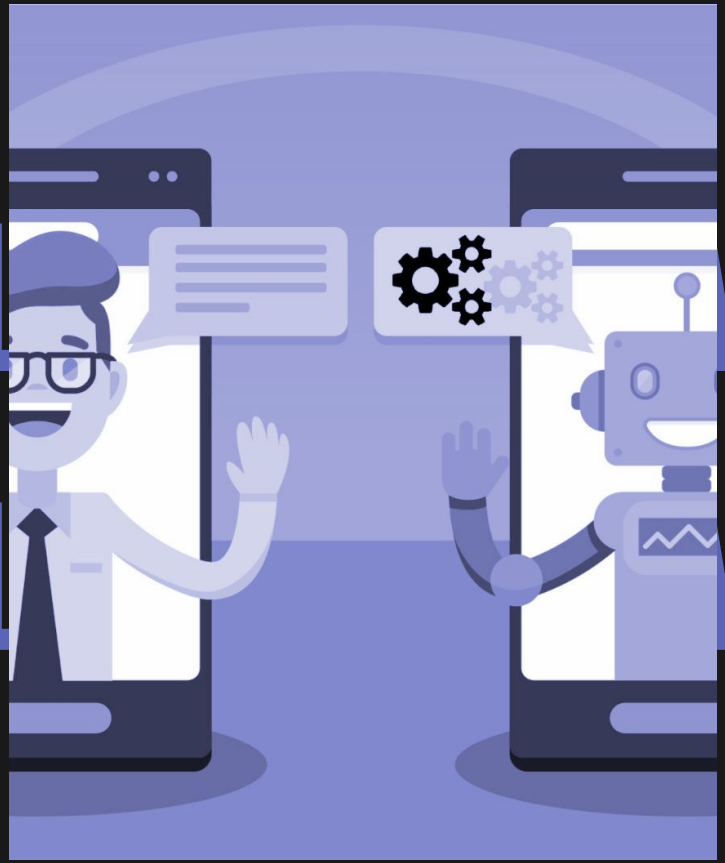
MAIN FEATURES

- Reward System
- Dynamic Rewards
- Map Updater
- Enemy Best Position to Kill Calculator
- Breadth-first Search



METHOD 3 - REWARDS

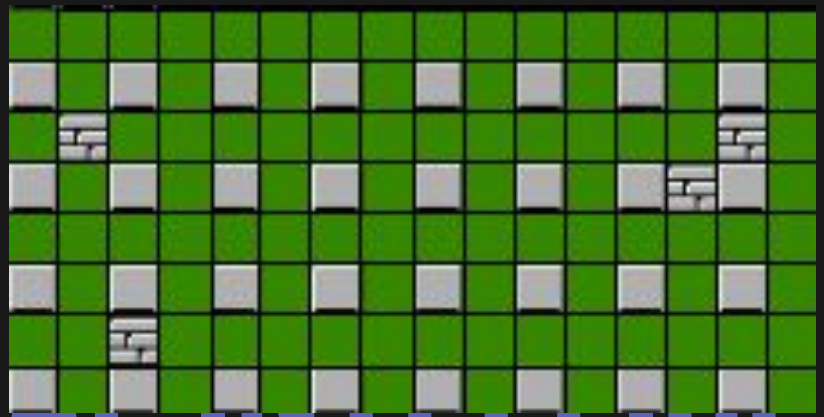
- **Indestructive Wall** = -100
- **Destructive Wall** = 4
- **Enemy Area** = -15
- **Enemy Position** = -20
- **Enemy Bomb** = 10
- **Bomb Drop Position** = -10
- **Bomb Damage** = 5
- **Step** = -0.01
- **Player** = -0.02
- **Powerup** = 5
- **Exit** = 50



METHOD 3 - DYNAMIC REWARD

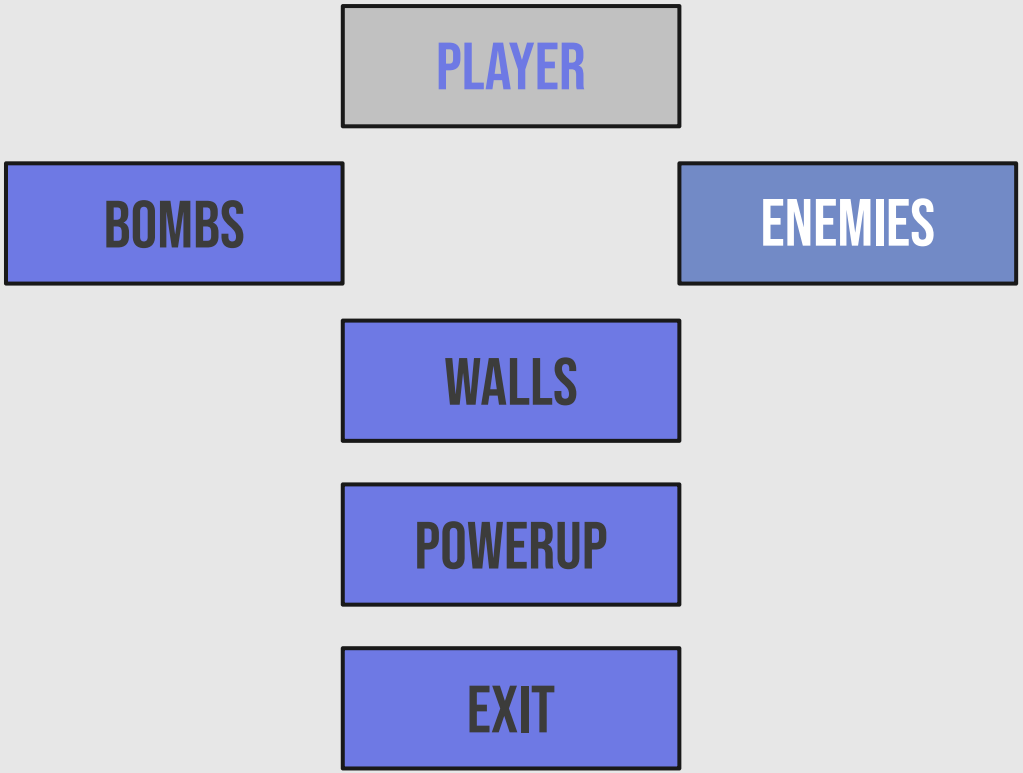
REWARD

- Enemies and walls have areas of value
- Overlapping areas are better by small amount



ARTIFICIAL INTELLIGENCE (AI)

METHOD 3 - MAP UPDATER



METHOD 3 - BREADTH-FIRST SEARCH

ALGORITHM

- Search tries to find the node with the best local reward and max depth of seven
- If current has a better local reward than the best node, than it is now the best node
- If the position resulting of the action was already on any ancestor of the node, then the node is not opened
- If the position resulting of the action is not a valid position or local reward is less then -15, then the node is not opened
- If the total reward of the node is below -30, then the node is not opened
- If after the loop there was no good node found, then the next Bomberman position will be the node with the best total reward

03.

COMPARISON

Comparison between approaches
Final results

RESULTS

	10 Runs			Highest Ever Score
	Lowest Score	Average Score	Highest Score	
Method 1	400	800	1500	2700
Method 2	500	700	1100	1500
Method 3	200	400	700	900

04.

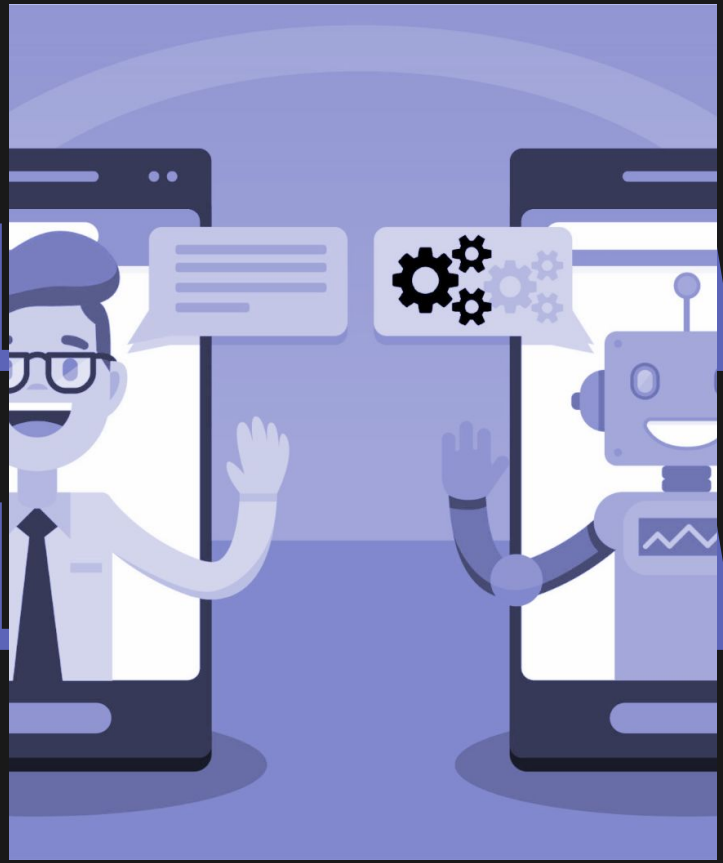
CONCLUSION

Best solution and future work

BEST SOLUTION

After testing for a little bit of time, we can say that the best method is the Approach 1

However, with more tests and more runs, it might not be the case



FUTURE WORK

FUTURE WORK

To improve our best approach we should:

- End possible loops with a simple loop detection
- Improve the reward system
- Improve the map updater to be more rough and precise
- Add cumulatively reward values for bomb position from both enemies and walls