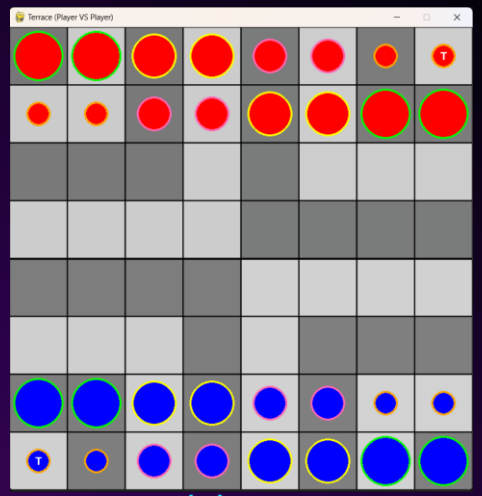

TERRACE
THE PUREST STRATEGY GAME
STARSHIP
U.S.S. ENTERPRISE
NCC-1701
UNITED FEDERATION OF PLANETS
CLASS : CONSTITUTION
SERVICE : 2245-2285
MAX. SAFE SPEED : WARP 8
POWER : MATTER/ANTIMATTER WARP CORE
ARMAMENTS : TORPEDO LAUNCHERS / PHASER BANKS
DEFENSES : DEFLECTOR SHIELDS

GAME DEFINITION

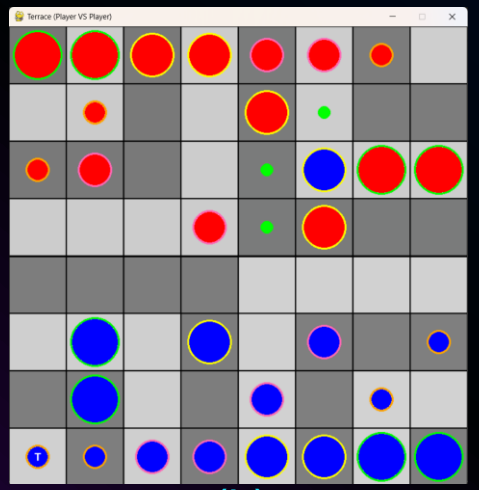
Terrace is an abstract board game played by two players on a multi-leveled board. Each player has 16 pieces with different sizes and the most important one is the “Terrace”. The main goal is to eliminate the opponent’s “T” piece or with your “T” piece reach the opposite corner of the board.

FORMULATION OF THE PROBLEM

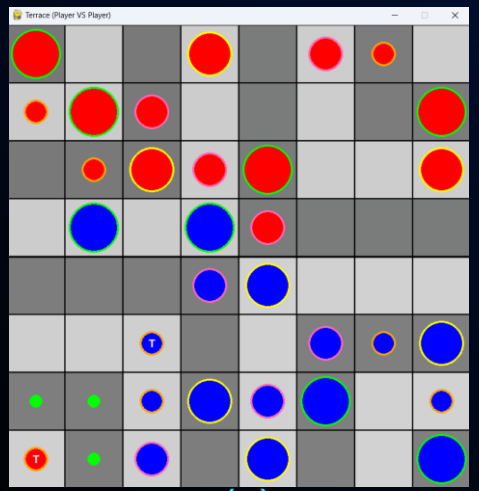
- State Representation: every piece can move anywhere, several representations are possible.
- Initial State: represented in picture (a)
- Objective State: picture (b) “T” piece is captured, or picture (c) “T” piece reached the opponent's corner.



(a)



(b)



(c)

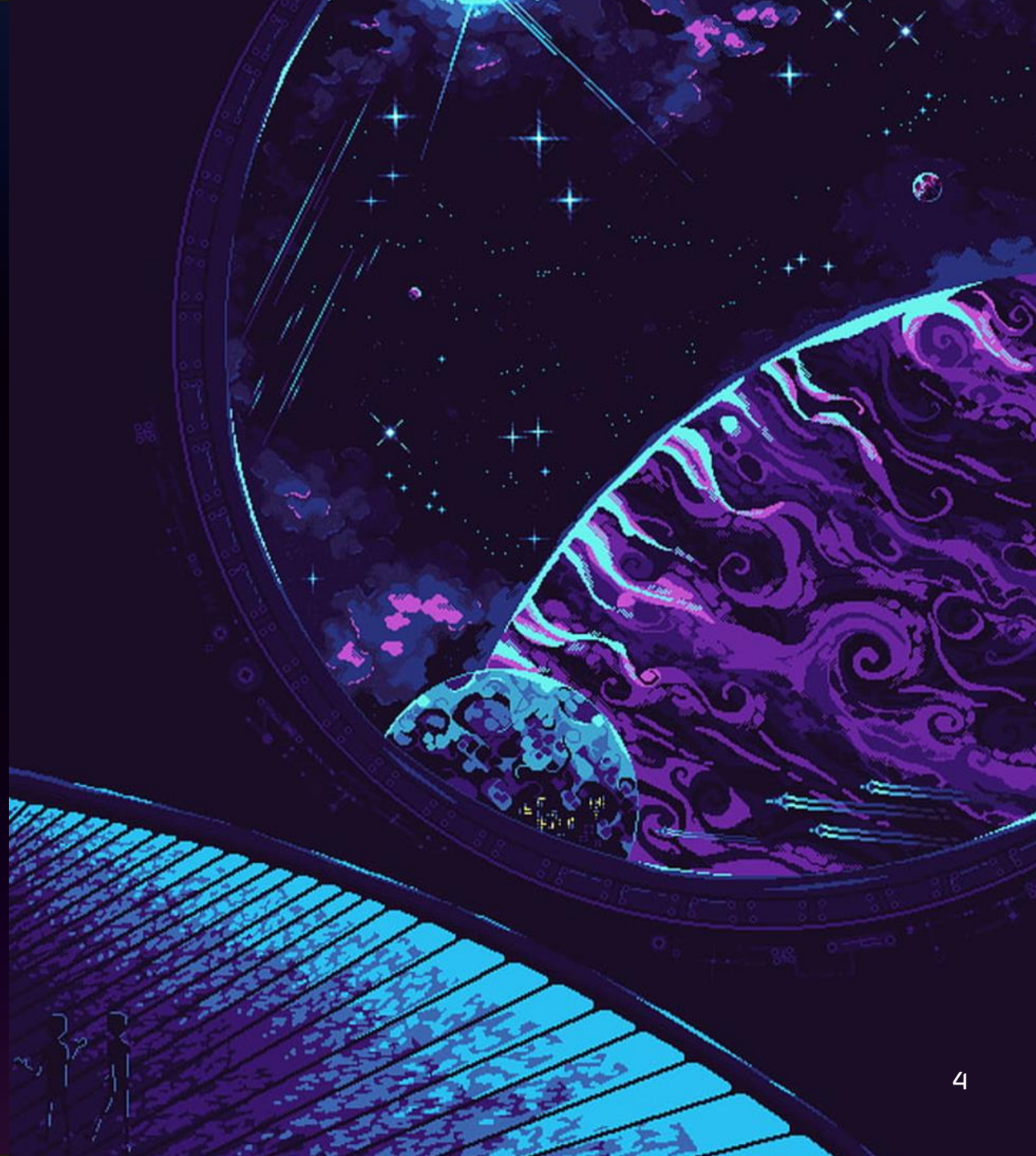
FORMULATION OF THE PROBLEM...

Operators:

1) ***Name***: Move(x),
   Preconditions: MoveDownStraight(x), MoveUpStraight(x), MoveUpDiagonically(x)
   Cost: 1

2) ***Name:*** Capture(x),
   Preconditions: CaptureDiagonicallyDown(x)
   Effect: capturing opponent's piece
   Cost:1

3) ***Name:*** Cannibalism(x),
   Precondition: CannibalismDiagonicallyDown(x)
   Effect: capturing my own piece,
   Cost: 1



IMPLEMENTATION WORK

- The game is implemented in python, and we are using the library pygame.
- We applied 2D matrices (list of lists) to represent the rows and the column of the board. Also, we used dictionaries to store all the valid moves for each piece and tuples to represent each coordinate.
- The algorithm that our AI is going to use is Mini Max.
- The AI will evaluate it's moves depending on:

  a. **winning conditions :** capturing the opponent's "T" piece or reaching your "T" piece to the opponent's exact opposite corner.

  b. **other strategies**

THE APPROACH

- ***Evaluation:*** Through the evaluation we obtain a score, the highest score refers to red pieces and the lowest to the blue. According to the strategy we use we multiply by a value.
  The **heuristics** we used are :

✓ **Distance to opponent's king**, assessing the distance between the AI's piece and the opponent's piece which is winning condition if the opponent's piece is captured.

✓ **Distance to opponents' corner**, evaluating the distance between the AI's king and the opponent's corner, reaching there is a winning condition.

✓ **Number of pieces,** indicates an advantage or disadvantage based on the number of pieces remaining on the board

✓ **Diversity of piece choices,** this ensures that the AI maintains flexibility and doesn't commit to a single type of piece.

✓ Worthiness of captures, prioritizes capturing high value pieces or strategic positions.

THE APPROACH...

- ***Operators:***

1. **Same level movement:** this operator allows the piece to move at the same level square if it's not occupied by another piece.

2. **Different level movement**: this operator allows the piece to move to a different level square, the piece can only move down straight, or it can move up straight and diagonally.

3. **Capture:** this operator allows the piece to capture the opponents piece if some conditions are met. Checks if the target piece is larger or equal in size to the capturing piece and it allows capturing only if movement is diagonal down

4. **Cannibalism:** The same conditions with capture but captures same team's piece for strategic purposes.

IMPLEMENTED ALGORITHM

- The algorithm that we implemented is called **“Minimax”** . It is a decision-making algorithm commonly used in two player games because it used the fact that the two players are working towards opposite goals. The algorithm continuously evaluates possible future game states by alternating between maximizing and minimizing players.
- In our game the maximizing player is the red pieces, and the minimizing players is the blue pieces. For each players turn consider all possible moves and evaluates them using the “evaluate” function (which referred to it in the previous slide) to determine the best move. The maximizing player aims to maximize the evaluation score while the minimizing player tends to minimize it. The algorithm explores the game tree up to a certain given depth.
- We also implemented **alpha beta cuts**, which is used to improve the efficiency and optimize the minimax algorithm, it accomplishes that by pruning branches of the game tree that are guaranteed not to lead to a better outcome. This helps reduce the number of nodes that need to be evaluated.

CONCLUSIONS

- Through this project we implemented our theory knowledge of search games algorithm into an actual abstract board game. We managed to create an AI who could make its own decisions which were guided by heuristics and could evaluate the best one. Moreover, the mini max algorithm was proved effective and ensures good strategic planning for the AI. Lastly, as for the efficiency of the AI alpha beta cuts made the algorithm more optimized making the AI “think” faster.

MATERIAL USAGE

- **Languages:**
  - Python and PyGame library
- **Software:**
  - Visual Studio Code
  - Power point to make the presentation

BIBLIOGRAPHIC SEARCH

Understanding of the game:

- https://youtu.be/T4knZmJbx-I?si=7m4DeB25YWZY3uWy
- https://www.youtube.com/watch?v=nLs3NwXkm_A&pp=ygUSdGVycmFjZSBib2FyZCBnYW1l
- https://classicreload.com/win3x-terrace.html

Code source:

- https://www.youtube.com/watch?v=vnd3RfeG3NM&ab_channel=TechWithTim
- https://www.youtube.com/watch?v=LSYj8GZMjWY&ab_channel=TechWithTim
- https://www.youtube.com/watch?v=_kOXGzkbnps&ab_channel=TechWithTim

Algorithm:

- https://cs.stanford.edu/people/eroberts/courses/soco/projects/2003-04/intelligent-search/minimax.html

MARIA MITKOU UP202311354
DOMINGOS NETO UP202108728
FLORENTIA-STYLIANI DIAMANTOPOULOU UP202311353