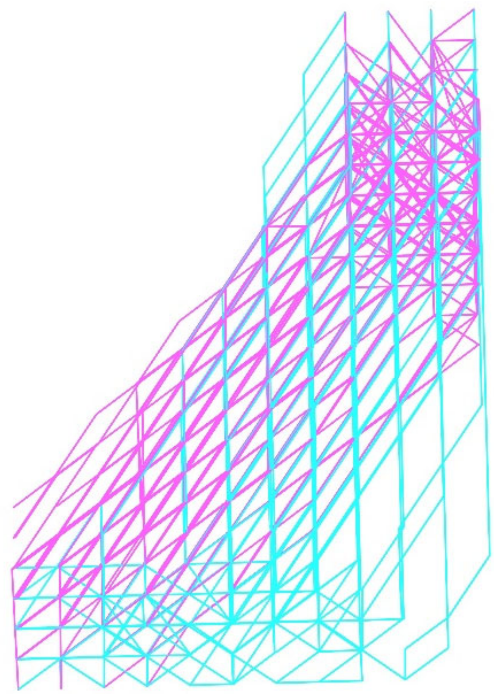


3D Autonomous Checkers

Marco Juliani
AC Studio 2
Page Count: 18

Overall Objectives

- Extend traditional 2D Chinese Checkers into a 3D version
- Have autonomous 'armies' compete with each other in reaching each other's base
- Incorporate 'ladder' behavior characteristic in checkers
- Incorporate A* Shortest Path to make shortest possible game
- Train opponent 'armies' to outperform their rival

Inspiration[1]

THE SHORTEST GAME OF CHINESE CHECKERS
AND RELATED PROBLEMS

George I. Bell
Tech-X Corporation, 5621 Arapahoe Ave Suite A, Boulder, CO 80303, USA
gibell@comcast.net

Received: 3/4/08, Revised: 10/13/08, Accepted: 12/20/08, Published: 1/5/09

Abstract

In 1979, David Fabian found a complete game of two-person Chinese Checkers in 30 moves (15 by each player) [Martin Gardner, Penrose Tiles to Trapdoor Ciphers, MAA, 1997]. This solution requires that the two players cooperate to generate a win as quickly as possible for one of them. We show, using computational search techniques, that no shorter game is possible. We also consider a solitaire version of Chinese Checkers where one player attempts to move her pieces across the board in as few moves as possible. In 1971, Octave Levenspiel found a solution in 27 moves [Ibid.]; we demonstrate that no shorter solution exists. To show optimality, we employ a variant of A* search, as well as bidirectional search.

A.1 Shortest games

Chinese Checkers in 30 moves (Figure 5), 10 man armies, 6-move rules (by David Fabian): c2-d2, h8-h6, d1-d3, i6-g6, a3-c3-e3, g9-g7-g5, a2-c2-e2-e4, h7-h5-f7, e4-f4, i9-g9-g7-e7, d3-c4, g6-g4-e4-e2-c2-a2, a4-c2-e2-e4-g4-g6-e8-e6, i8-i6-g6-g4-e4-e2-c2-a4, a1-a3-c3-c5, f9-h7-h5-f5-f3-d3-d1, c5-d5, e7-e5-c5-c3-a3-a1, b2-b4-d4-d6-f6-f8-h8, i7-g9-g7-e7-e5-c5-c3-a3, c1-e1-c3-c5-e5-e7-g7, h6-f8-f6-d6-d4-b4-b2, b3-b4, g8-g6-g4-e4-e2-c2, b1-b3-b5-d3-f3-f5, g5-e5-c5-c3-c1, f5-f6, f7-f5-f3-d3-b5-b3-b1, e6-g6-g8-i8, h9-h7-f7-f5-f3-d3-b5-b3 (red wins).

Chinese Checkers in 36 moves, 15 man armies, 6-move rules: e1-e2, g8-g6, c1-e1-e3, h6-f6, e3-e4, f9-f7-f5, a1-c1-e1-e3-e5-g5-e7, g7-g5-e5-e3-e1-c1-a1, a5-b5, i7-g7-g5-e5-e3-e1-c1, c3-a5-c5, g9-i7-g7-g5-e5-e3-e1-c3-a5, a3-c3-e1-e3-e5-g5-g7-i7-g9, i9-i7-g7-g5-e5-e3-e1-c3-a3, a4-c4-c6, e9-g7-g5-e5-e3-e1-c3, c6-d6, i5-i7-g7-g5-e5-e3-e1, a2-a4-c4-c6-e6-g4, i8-g8-e8-e6-c6-c4-a4-a2, d2-f2, i6-g8-e8-e6-c6-c4-a4, b3-d3-f1-f3-d5-d7-f7-h5, f5-f7-d7-d5-f3-f1-d3-b3, d1-f1-d3, h8-h6-h4-f4-d4-d2, b1-d1-f1-f3-d5-d7-f7-f5, h9-f9-f7-d7-d5-f3-f1-d1-b1, b4-b6-d4-f4-h4-h6-h8, h7-h9-f9-f7-d7-d5-f3-f1-d1, c2-c4-c6-e6-e8, g6-e6-c6-c4-c2, b2-b4-b6-d4-f4-h4-h6, f6-f4-d4-b6-b4-b2, d3-f1-f3-d5-d7-f7-f9-h9, f8-d8-f6-f4-d4-b6-b4 (red wins).

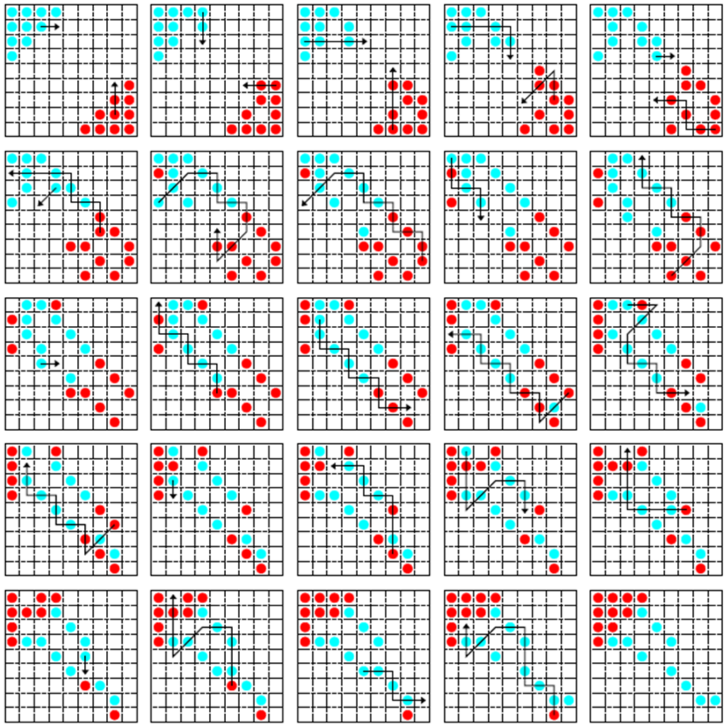


Figure 5: David Fabian's 30 move game of Chinese Checkers.

In summary, some of the properties that a short solution usually has are:

1. The winning player only jumps (on odd boards).
2. The first ladder is built by both players during the first α moves.
3. The second ladder is completed by the losing player on moves after α.

[1] Bell, George , I. 2009. The Shortest Game of Chinese Checkers and Related Problems. https://arxiv.org/pdf/0803.1245.pdf

Looking Forward

- Can we exploit 'higher dimensional solution space' (game permutations existing in 3D vs 2D) to make ever-shorter checker games using very efficient laddering?

- Will the 'ladder behavior' inherent in checkers combined with higher permutation space (and an implementation of A* shortest path as in the paper below) yield shorter games than those that have been documented below?

The Shortest Game of Chinese Checkers and Related Problems

Simple Goal-Oriented Motion (2D)

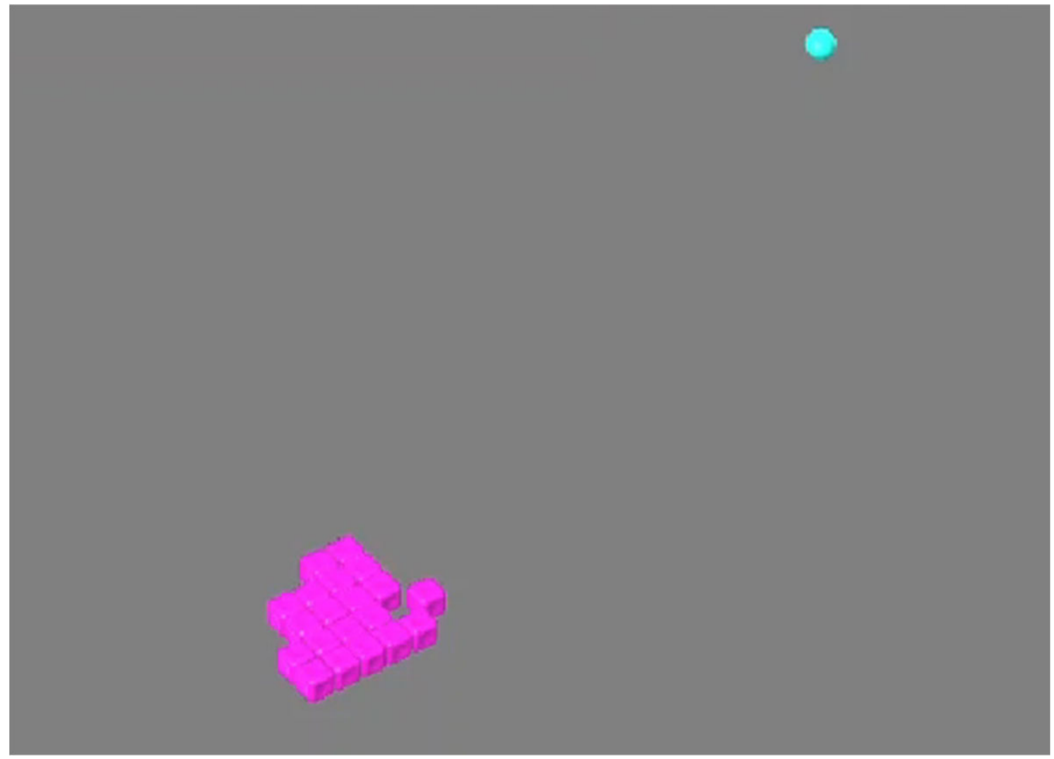
- Euclidean distance of neighbors to target.
- *Raycasts* and *box colliders* to determine neighbors and possible moves.
- One step at a time:
  - Either
    - 1, 0, 0
    - 0, 1, 0
    - -1, 0, 0
    - 0, -1, 0

0,1,0

-1,0,0 1,0,0

0,-1,0

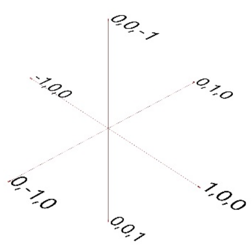
Possible Moves/
Neighbor
Selection



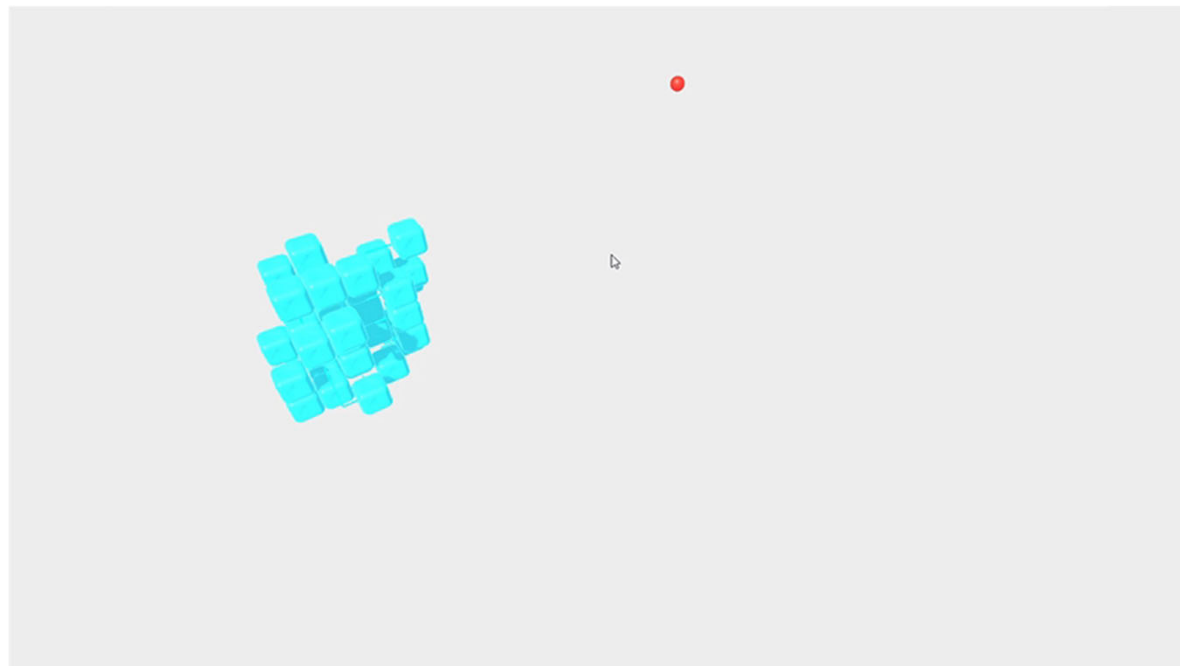
Progress Video

Simple GOM(3D)

- Euclidean OR Manhattan distance of neighbors to target.
- *Raycasts* and *box colliders* to determine neighbors and possible moves.
- One step at a time:
  - Either
    - 1, 0, 0
    - 0, 1, 0
    - 0, 0, 1
    - -1, 0, 0
    - 0, -1, 0
    - 0, 0, -1



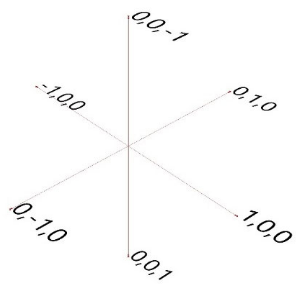
Possible Moves/
Neighbor
Selection



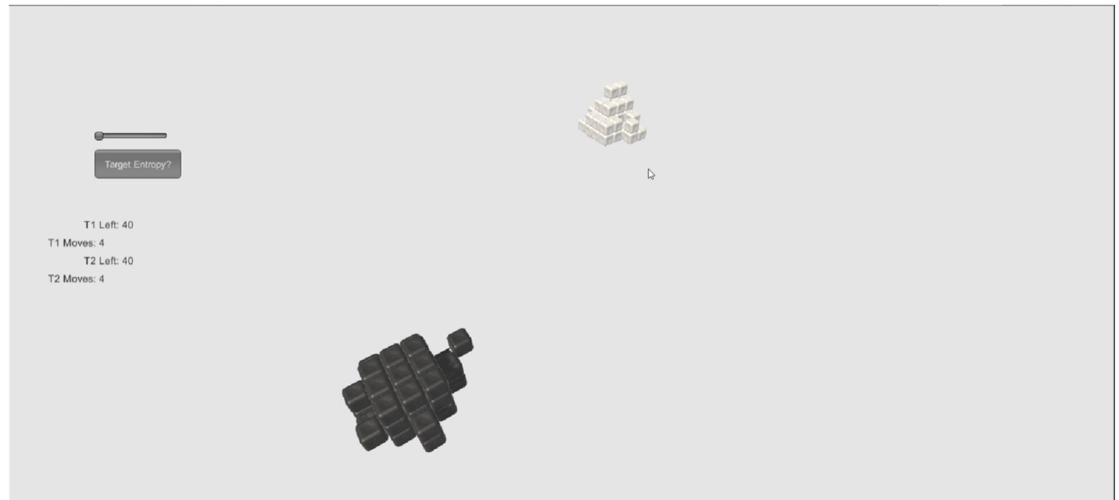
Progress Video

Simple GOM 2 Teams(3D)

- Incorporating *LineRenderer* to visualize paths taken
- One step at a time:
  - Either
    - 1, 0, 0
    - 0, 1, 0
    - 0, 0, 1
    - -1, 0, 0
    - 0, -1, 0
    - 0, 0, -1



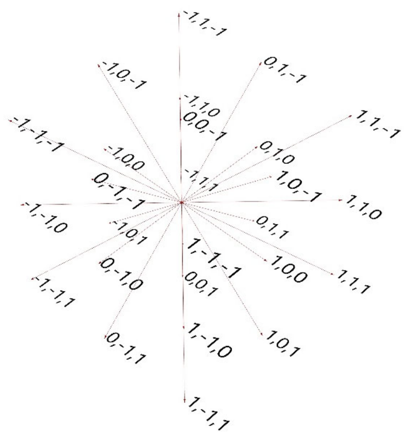
Possible Moves/ Neighbor Selection



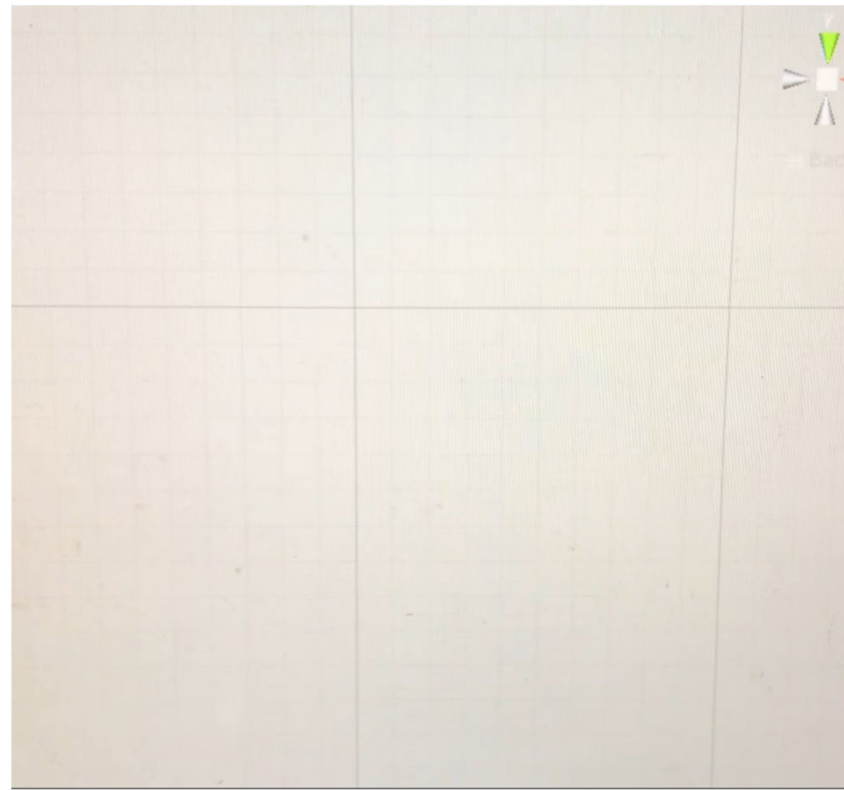
Progress Video

Refactored GOM 2 Teams(3D)

- [,,] Array with Cell states
  - Better Performance
  - Path-related querying abilities
- Incorporating *LineRenderer* to visualize paths taken
- One step at a time:
  - Either
    - 1, 0, 0
    - 0, 1, 0
    - 1, 1, 0
    - 1, -1, 0
    - -1, 1, 0
    - -1, -1, 0
    - -1, -1, -1
    - 1, 1, -1
    - 1, -1, -1
    - 1, -1, 1
    - 1, 1, 1
    - -1, -1, 1
    - -1, 1, -1
    - -1, 1, 1
    - 0, 1, 1
    - 0, -1, -1
    - 0, -1, 1
    - 0, 1, -1
    - 1, 0, 1
    - 1, 0, -1
    - -1, 0, 1
    - -1, 0, -1
    - 0, 0, 1
    - -1, 0, 0
    - 0, -1, 0
    - 0, 0, -1

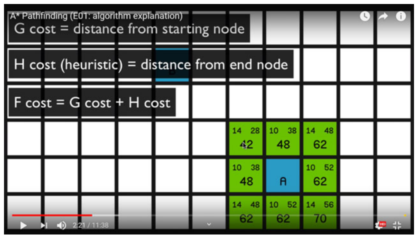


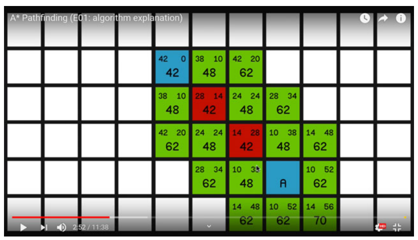
Possible Moves/ Neighbor Selection



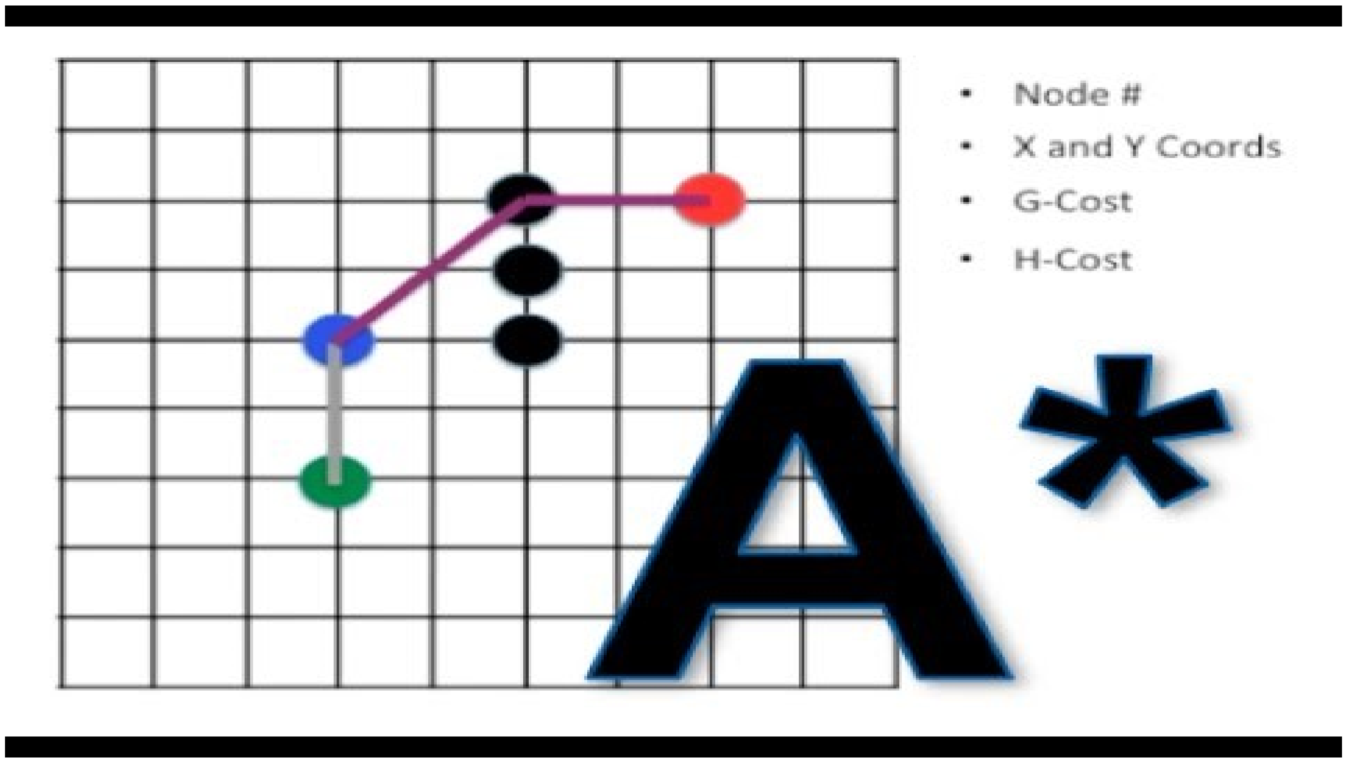
Progress Video

A* Shortest Path Review:





https://www.youtube.com/watch?v=-L-WgKMFuhE

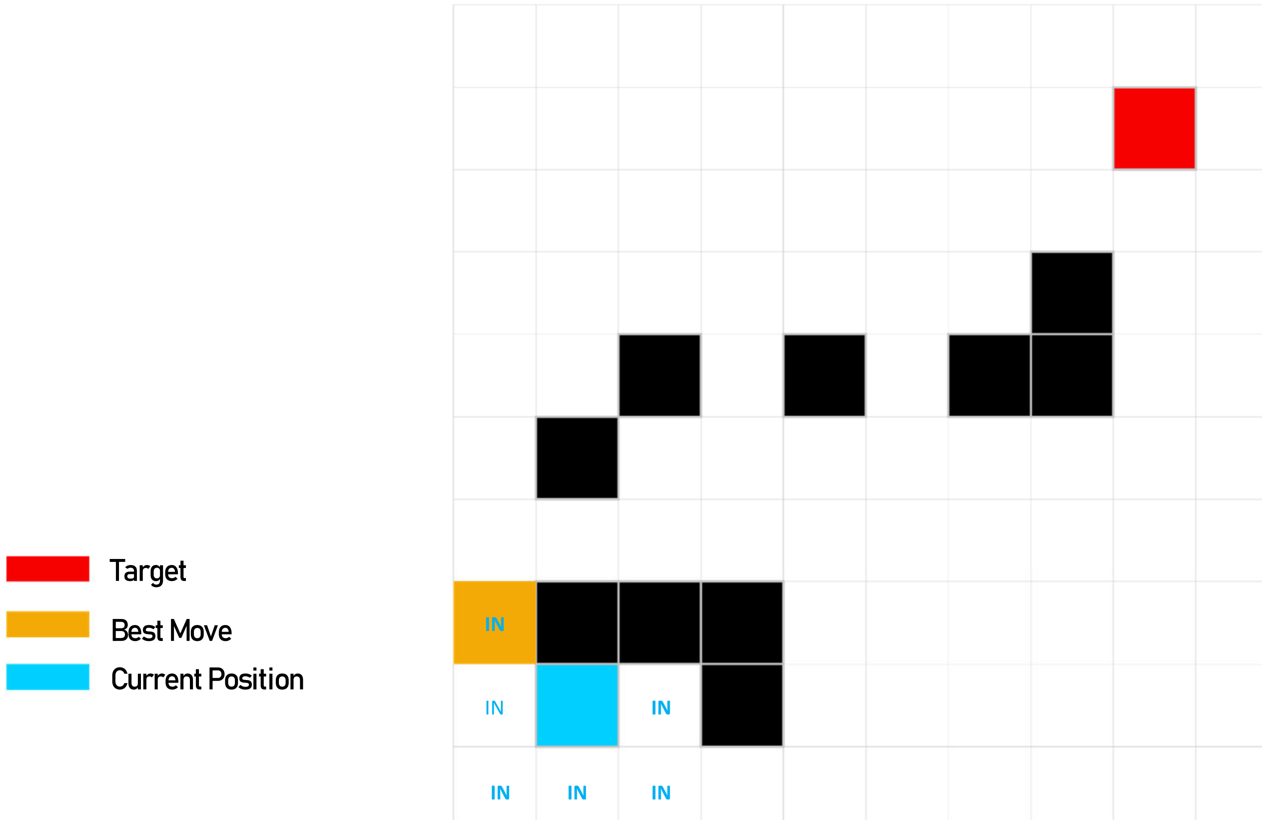


https://www.youtube.com/watch?v=pKnV6ViDpAI

(Go to 5: 40 for the comparison between Djistra's and A* algorithms)

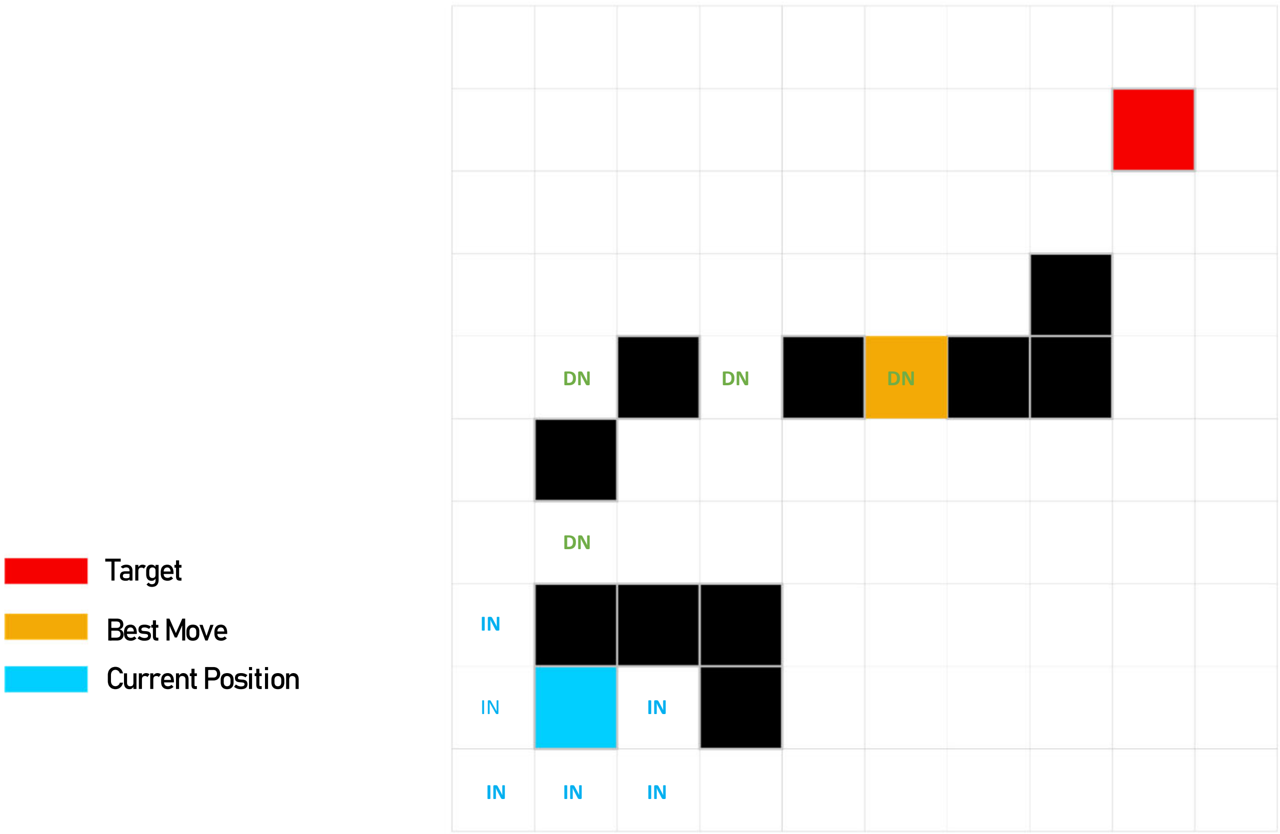
Move Selection

1. Find Immediate Neighbors (IN) = 6
2. Sort neighbors by their Manhattan(to target) and their Euclidean(to active cell)
3. Make the best ranking move

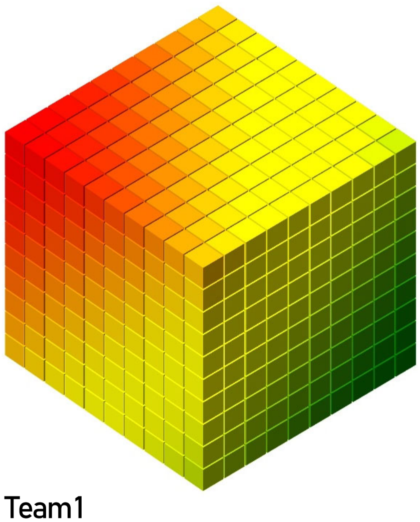


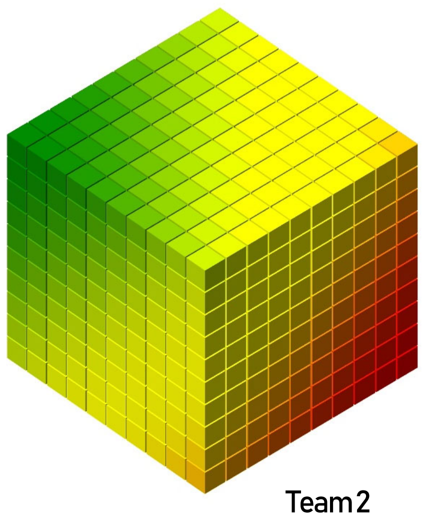
Move Selection– Including Ladder

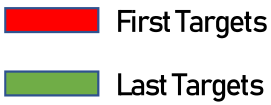
1. Find Immediate Neighbors (IN) = 6
2. Find Derived Neighbors (DN) = 4
3. Put IN and DN in one bucket = 10
4. Sort neighbors by their Manhattan(to target) and their Euclidean(to active cell)
5. Make the best ranking move



Target Selection:





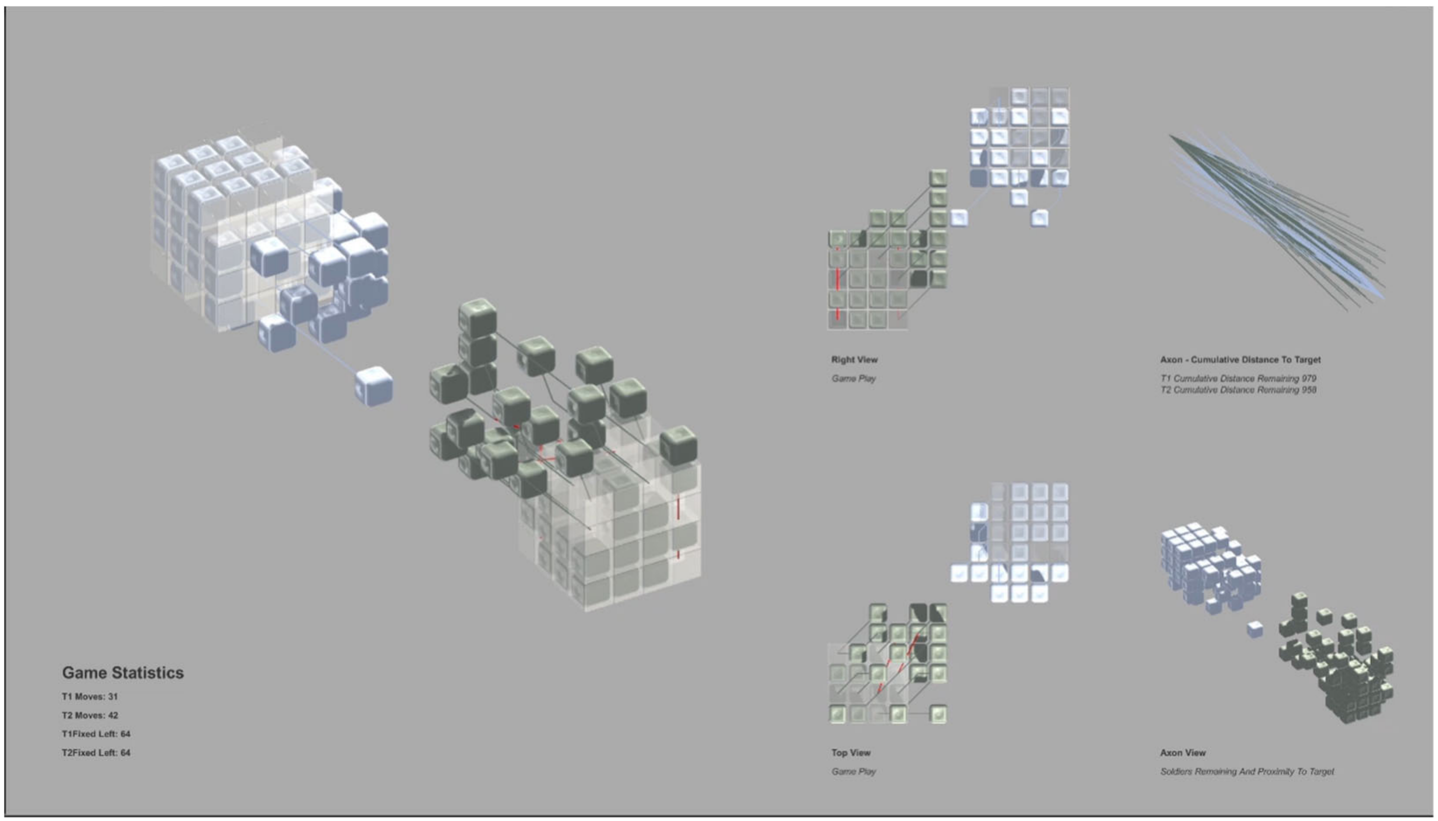


Pseudocode

(At each step):

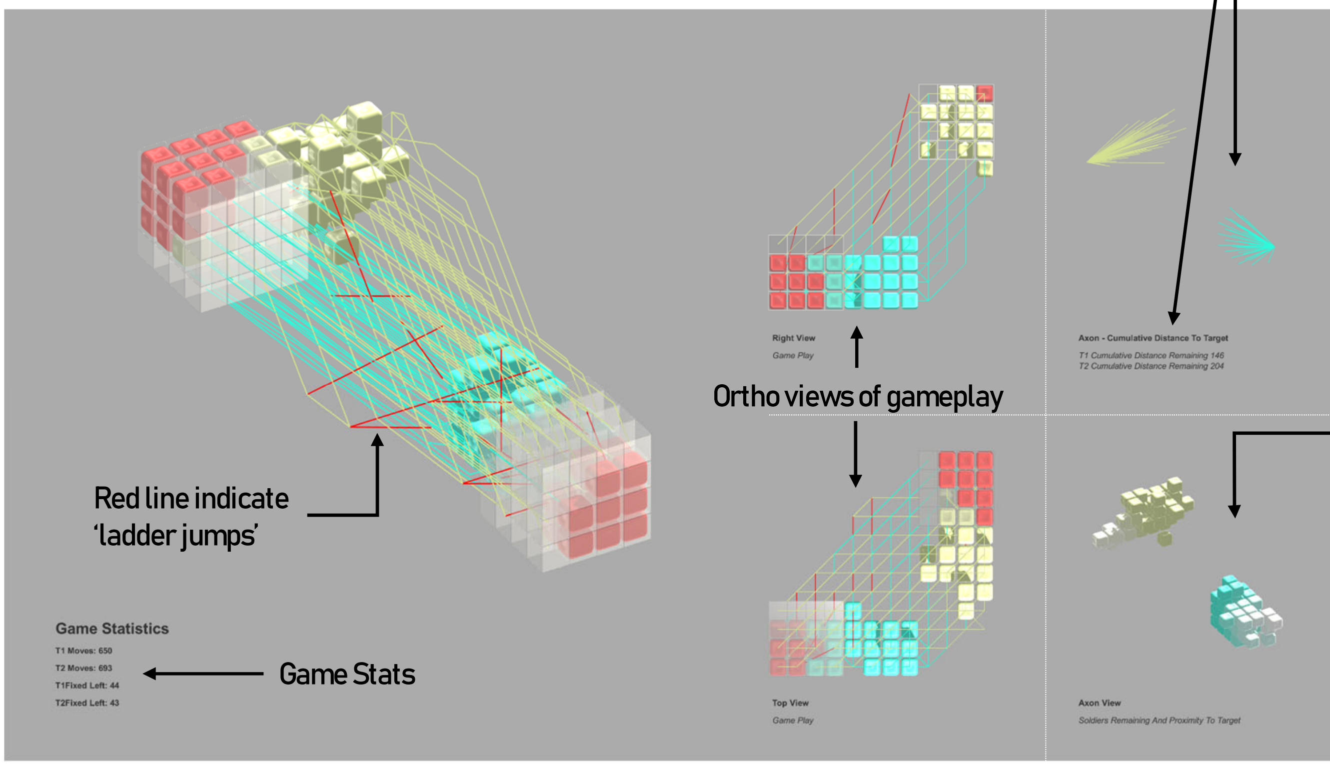
```
If(activeCell == Targets[0])
    Targets.RemoveAt(0);
```

Shortest Path Visualization:



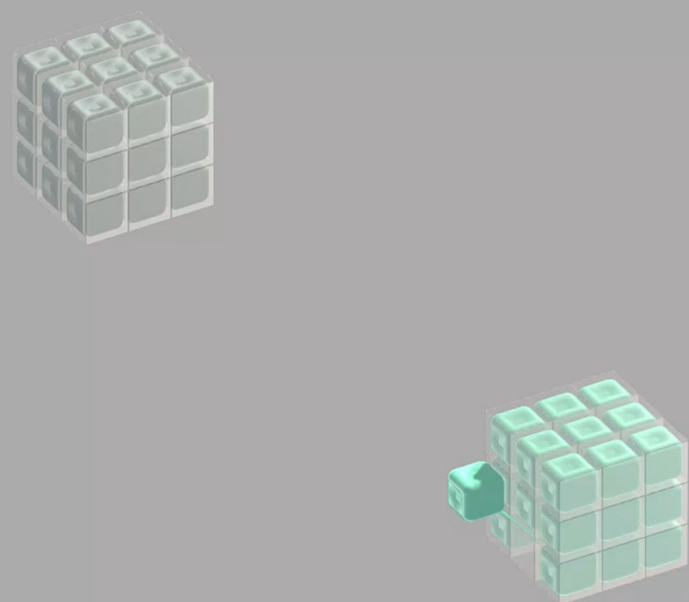
Progress Video

Interpreting the video:



Video Frame Explanation Diagram

Final – 0:33



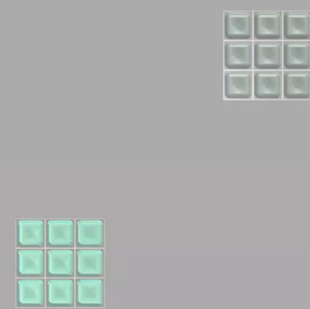
Game Statistics

T1 Moves: 0

T2 Moves: 1

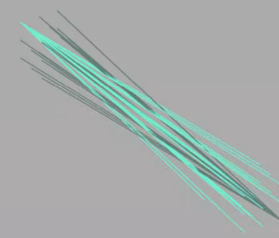
T1Fixed Left: 27

T2Fixed Left: 27



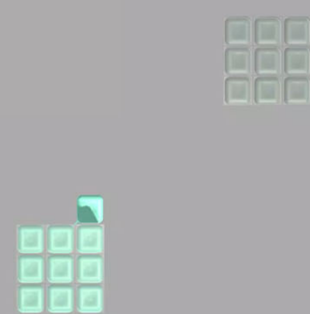
Right View

Game Play



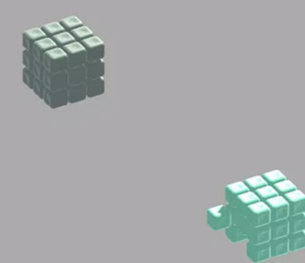
Axon - Cumulative Distance To Target

T1 Cumulative Distance Remaining 361
T2 Cumulative Distance Remaining 360



Top View

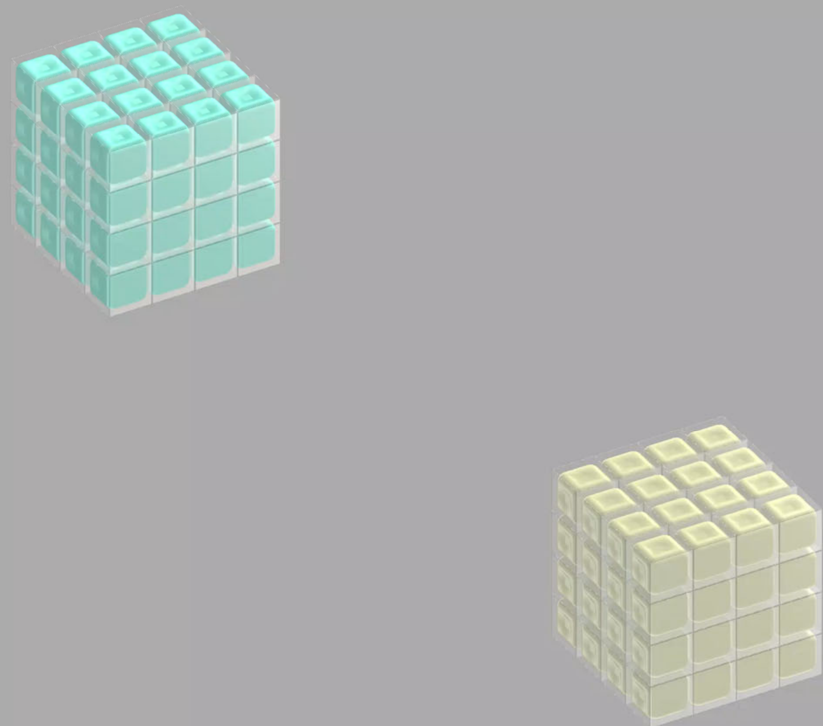
Game Play



Axon View

Soldiers Remaining And Proximity To Target

Final – 2:42



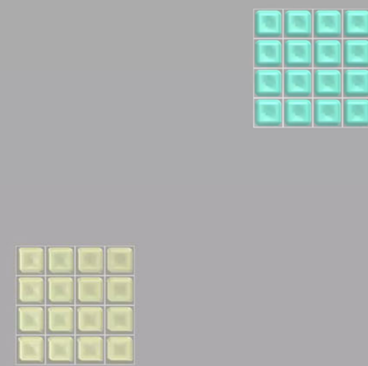
Game Statistics

T1 Moves: 0

T2 Moves: 0

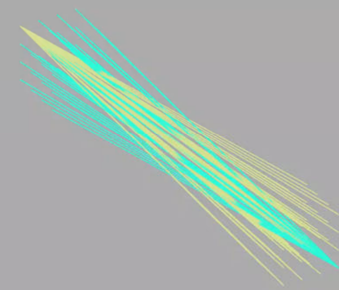
T1Fixed Left: 64

T2Fixed Left: 64



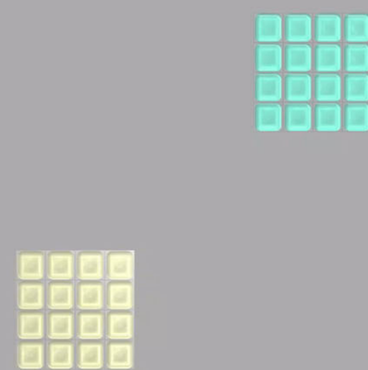
Right View

Game Play



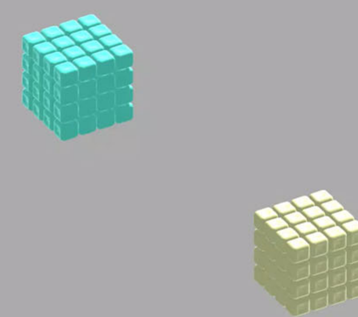
Axon - Cumulative Distance To Target

T1 Cumulative Distance Remaining 1024
T2 Cumulative Distance Remaining 1024



Top View

Game Play



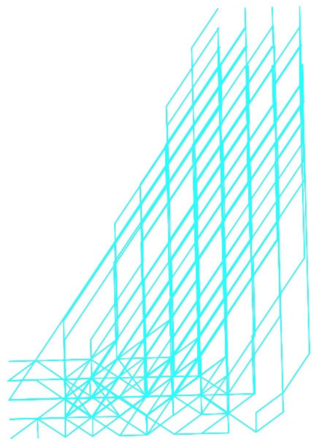
Axon View

Soldiers Remaining And Proximity To Target

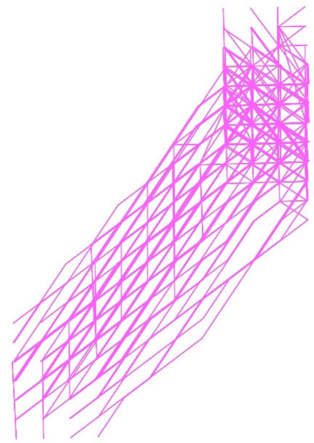
CSV Export & Data Viz

- These visualizations represent the data that might be used to train the game (positions, sequence, and number of moves).

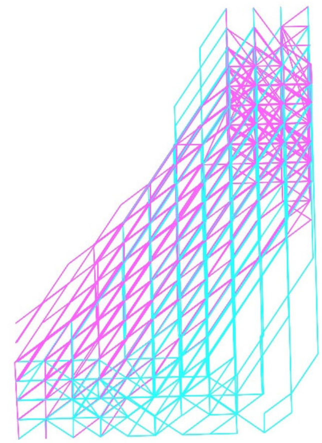
- A hypothesis might be that the game with the smallest number of lines and the shortest sum of the lengths might be the shortest game.



Team1 Paths



Team2 Paths



Combined Paths

Combinatorial Game Theory (CGT)

- Branch of mathematics and theoretical CS that typically studies sequential games with perfect information[2]
- What is a combinatorial game[3]:
  - Two players – *order in which they make moves is only distinguishing factor*
  - No chance *(only who gets to go first).*
  - Perfect information- *state of game fully visible by both players*
  - Turn based.
  - Absolute winner *(no ties or draws).*
  - Winning condition- *clearly defined by a 'terminal position'*
- "The type of games studied by CGT is also of interest in artificial intelligence, particularly for automated planning and scheduling. In CGT there has been less emphasis on refining practical search algorithms (such as the alpha–beta pruning heuristic included in most artificial intelligence textbooks), but more emphasis on descriptive theoretical results (such as measures of game complexity or proofs of optimal solution existence without necessarily specifying an algorithm, such as the strategy-stealing argument)."[4]

[2] https://en.wikipedia.org/wiki/Combinatorial_game_theory

[3] https://is.muni.cz/th/325040/fi_b/Combinatorial_games.pdf

[4] https://en.wikipedia.org/wiki/Combinatorial_game_theory