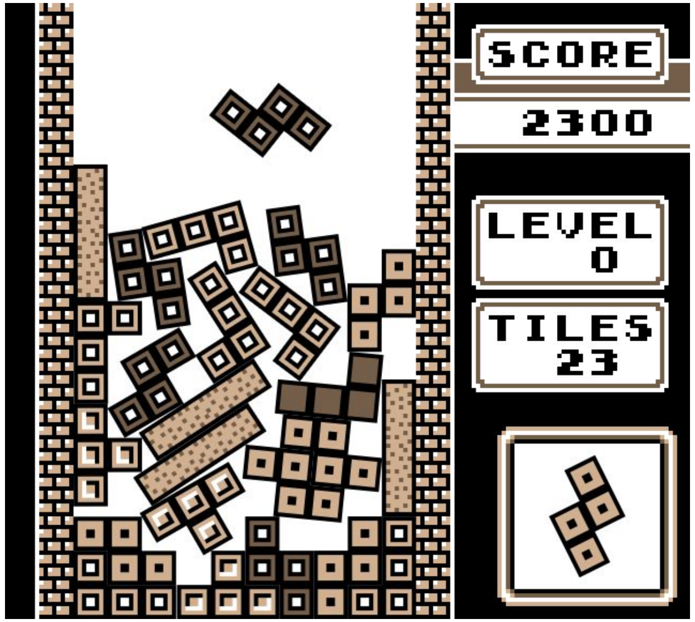
# Count of Monte Carlo

## Applying Monte Carlo Tree Search to Tetris




Samir Khays Kevin Holdcroft Yoshua Nava Viktor Tuul

# Contents

- Problem Introduction
- Methods
  - Search tree representation
  - State transition function
  - Monte  Carlo Tree Search
  - Heuristics
- Results
- Future Work

Title Image Source: http://stabyourself.net/images/screenshots/nottetris2-1.png
KTH Image Source: https://project-sparks.eu/wp-content/uploads/2014/04/KTH_Logotyp_RGB_2013.jpg

# Tetris

- Classic video game
- Guide 7 types of pieces onto a grid
- Goal to fill rows completely
  - Full rows disappear
- Know only current and next piece
  - Depending on variant
- Stochastic, non-deterministic

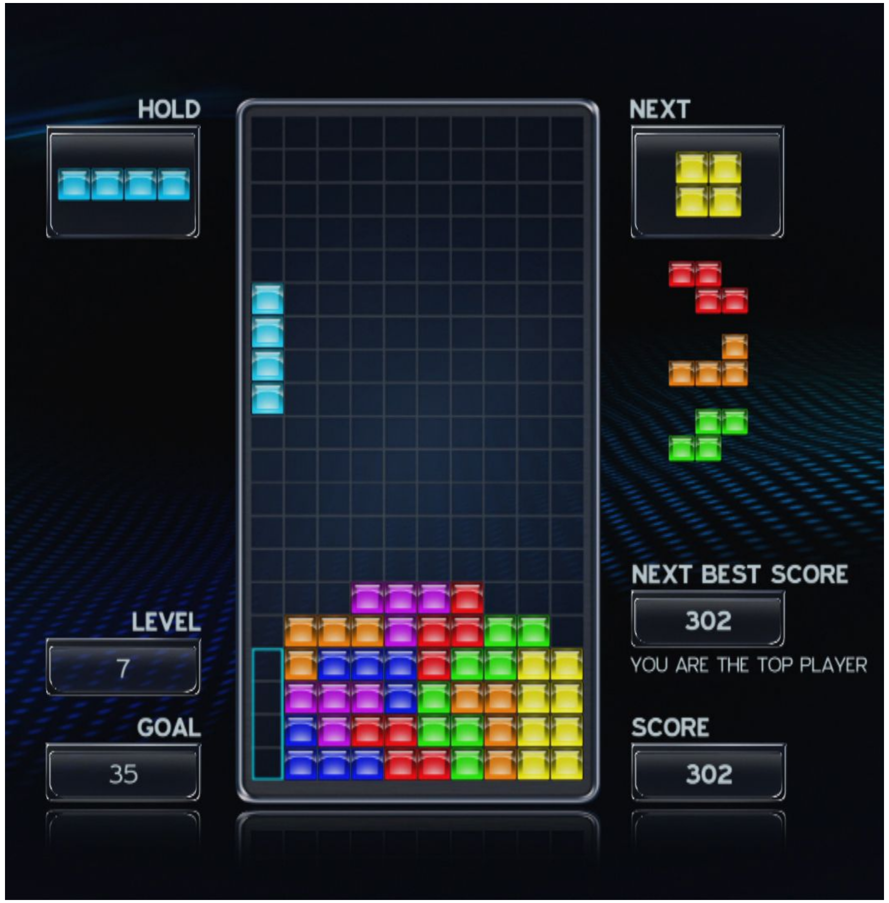





Image Source: http://mashthosebuttons.com/gallery/tetris_psn/marathon_hi_res.jpg

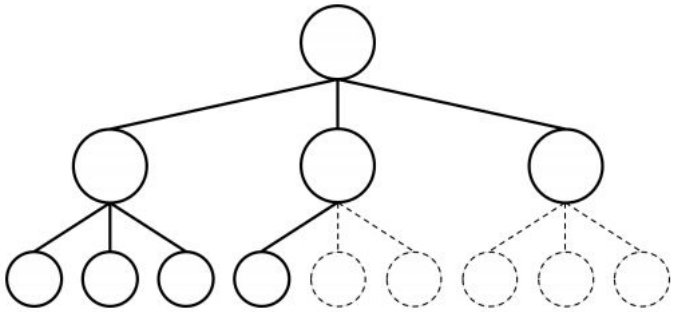
# Work done

- Implemented two different AI capable of playing Tetris
  - Monte Carlo Tree Search (MCTS)
  - Breadth-First Search (BFS)
- Originally implemented MCTS
  - Project goal
- BFS used to tune heuristics
  - Ended up performing excellently

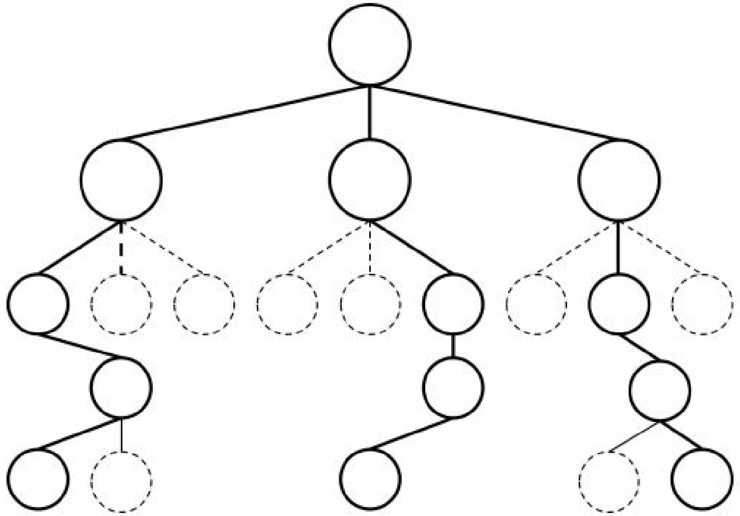
# Search tree representation

- Search tree algorithm utilization
  - Represent the game in a search tree
  - Nodes being the current and possible future game states

- State transition function
  - Needed in order to produce a search tree
  - Was not included in the open source game



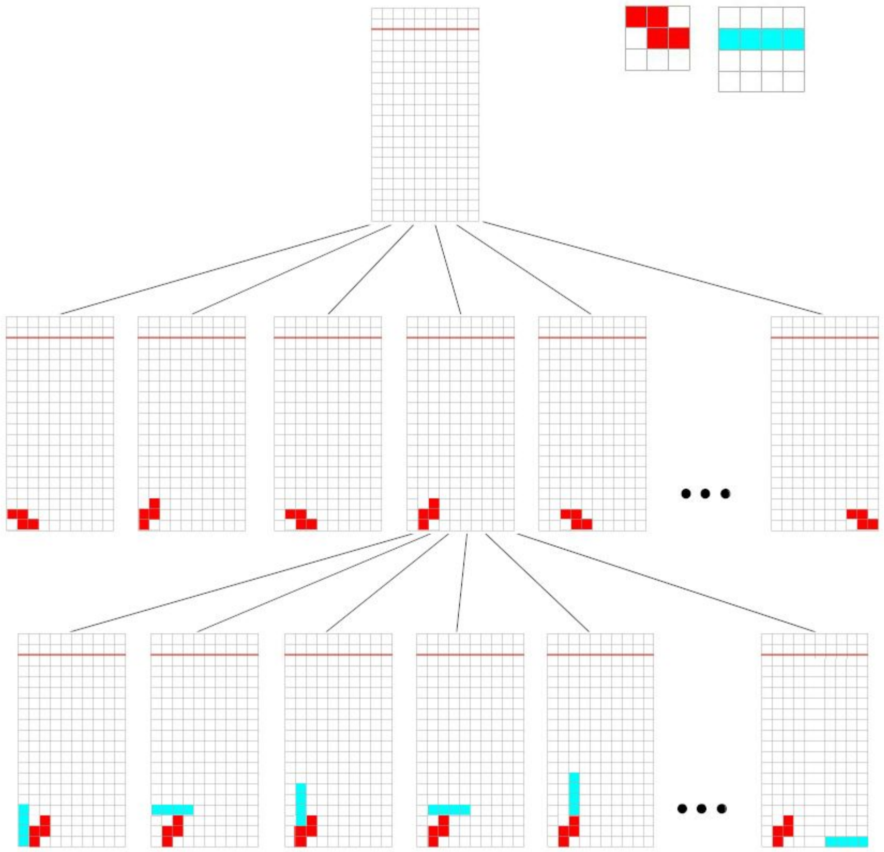
Search utilizing Breadth First Search



Search utilizing Monte Carlo Tree Search

# State transition function

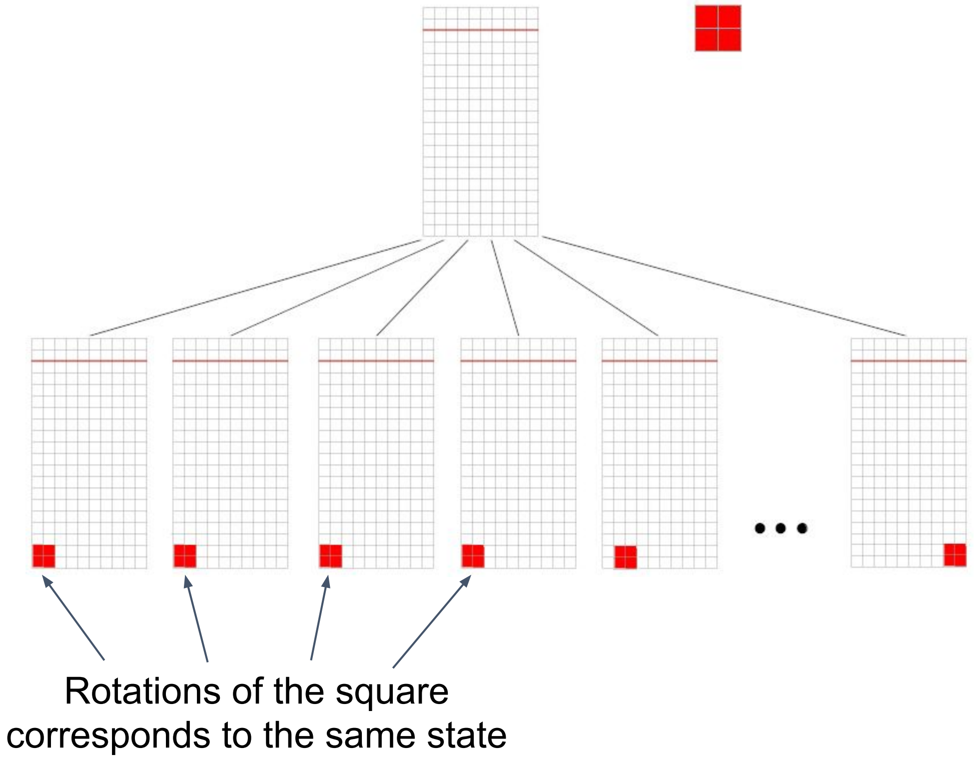
- State transition function
  - Given any state (board + falling brick) → get every possible child state
  - In practice: copies of the game and simulations for every brick orientation/place combination
  - Assignment of randomly chosen bricks → MCTS




source: http://totologic.blogspot.se/2013/03/tetris-ai-explained.html

# Avoidance of duplicate states

- Local hash tables
  - Efficiency when expanding child nodes (avoidance of duplicate states)
  - Example: square block rotations



# MCTS

MCTS

- **Start ->** Sampling -> Expansion -> Simulation -> Backprop -> **Start**
- Sampling criteria: UCB1.

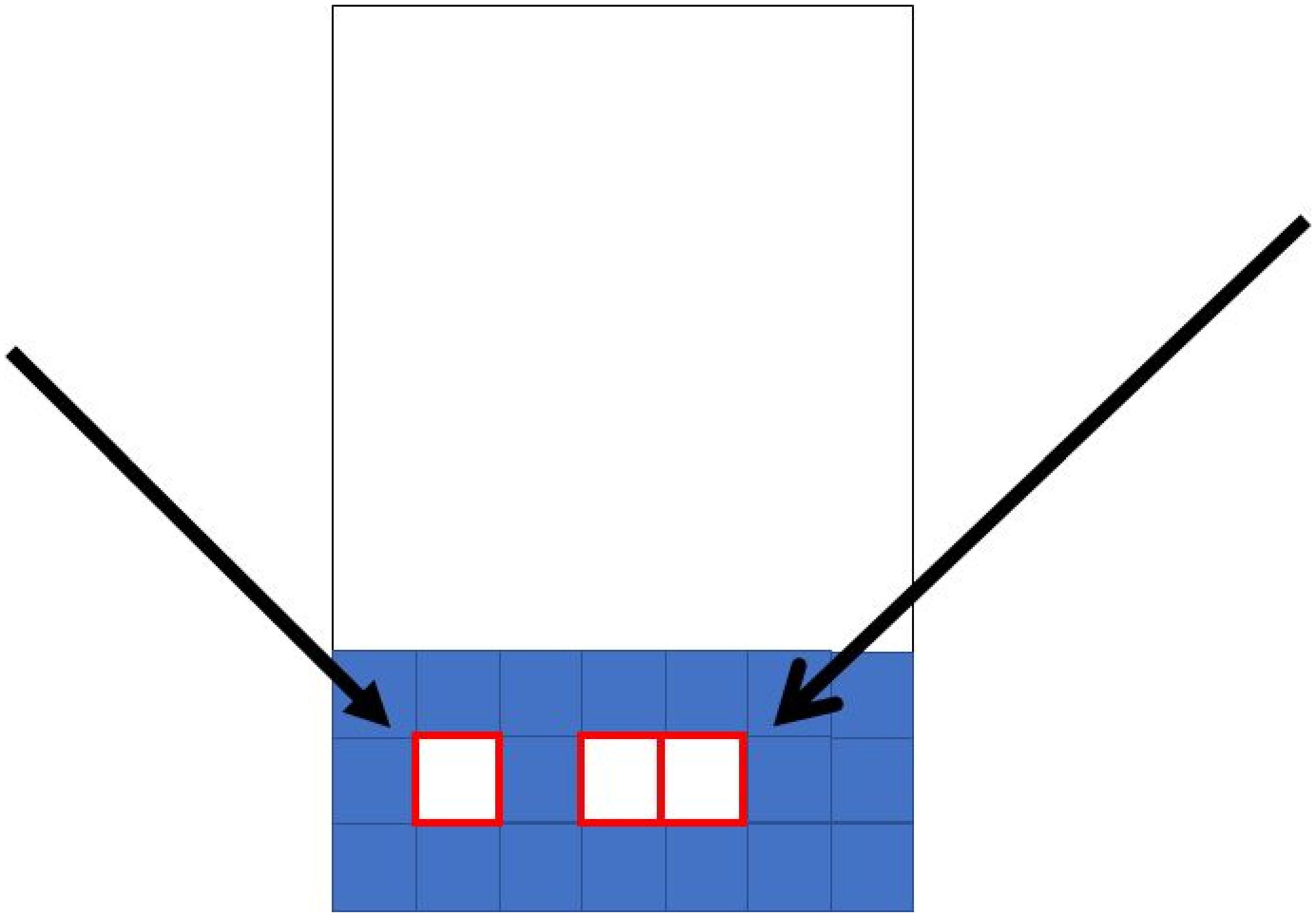
For simulation:

- First, tried out random (light) playouts.
- Then heavy playouts with our BFS.
- Performance limitations for deeper search.
- Winning condition in Tetris? Design decision.

# Heuristics

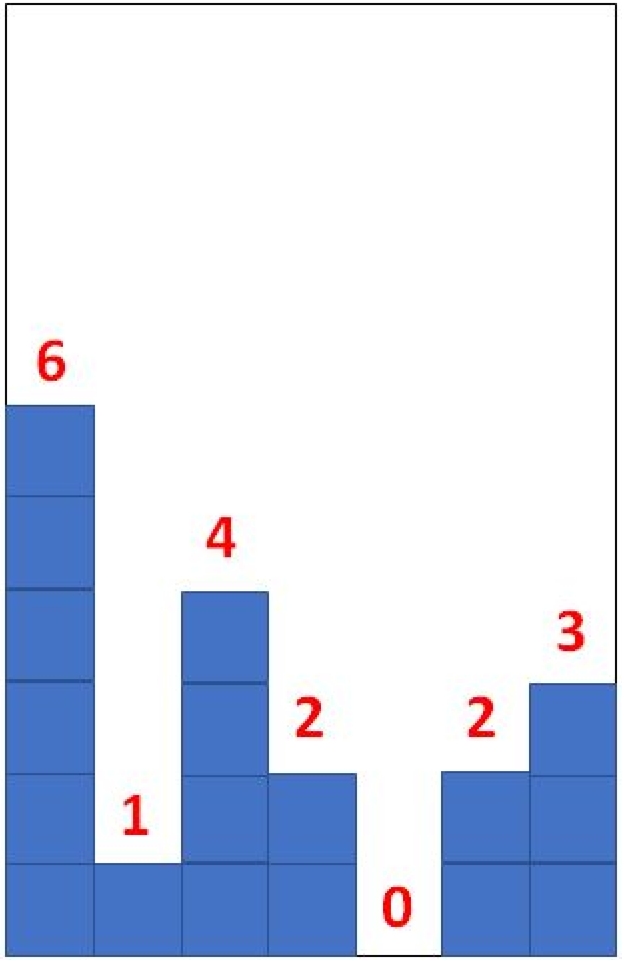
- Number of holes
- Aggregate height
- Cleared lines

# The Number of Holes heuristic

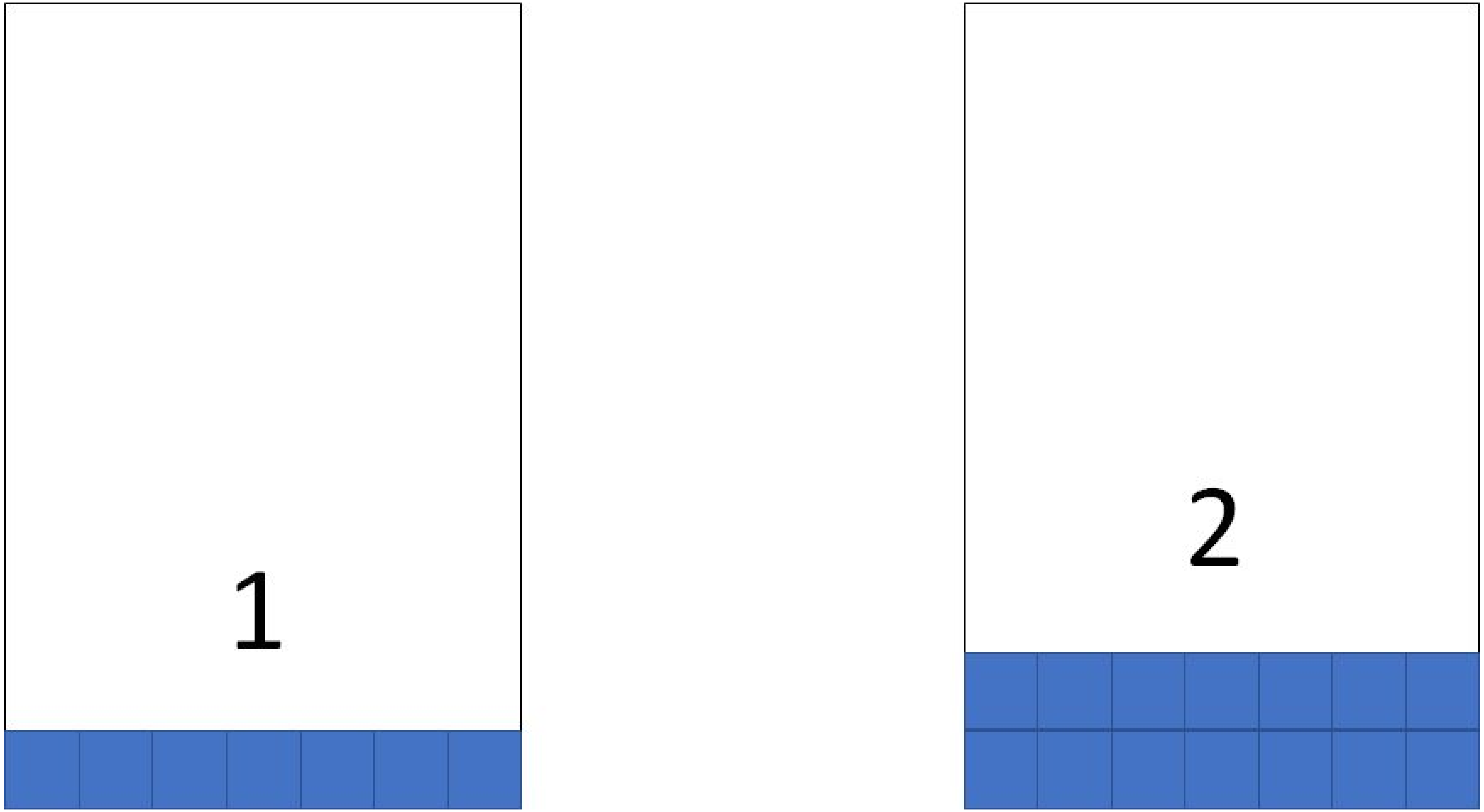


# The Aggregate Height heuristic

$$\sqrt{\sum_{i=0}^{9}(column\_height_i)^2}$$



# The Cleared Lines heuristic

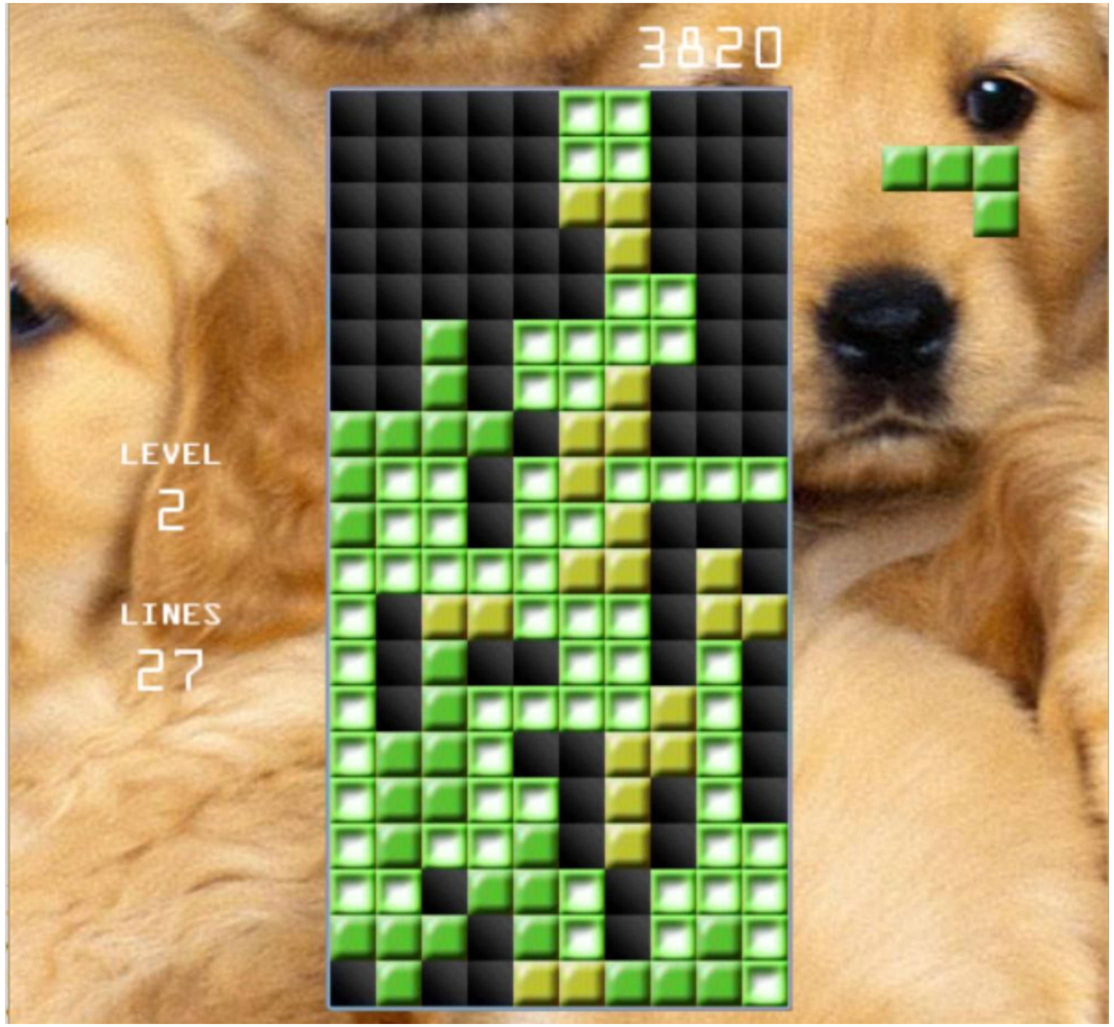



KTH Image Source: https://project-sparks.eu/wp-content/uploads/2014/04/KTH_Logotyp_RGB_2013.jpg

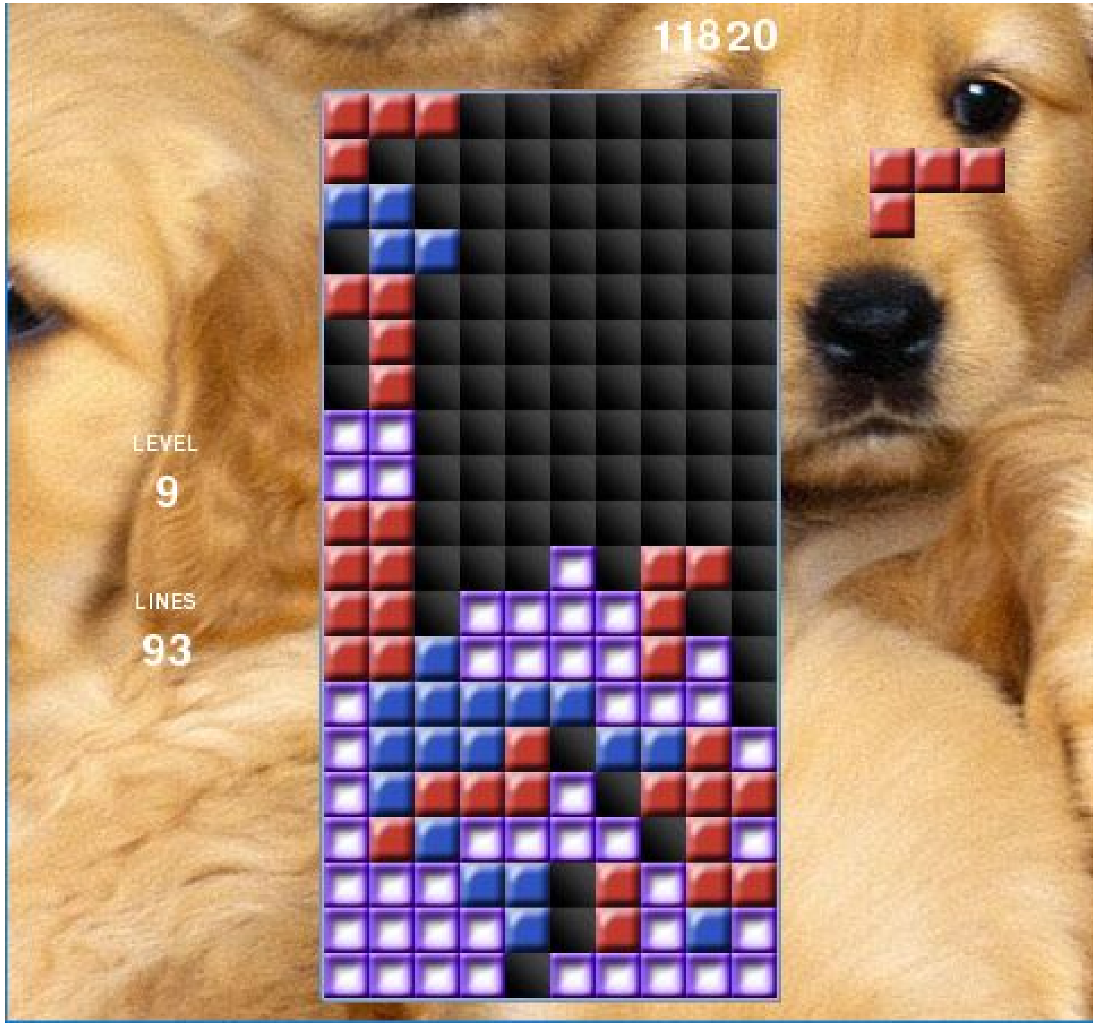
# Results

| Method | Lines Cleared | Score |
|---|---|---|
| BFS (depth 2) | 93 | 11820 |
| BFS (depth 1) | 40 | 5770 |
| MCTS | 27 | 3820 |

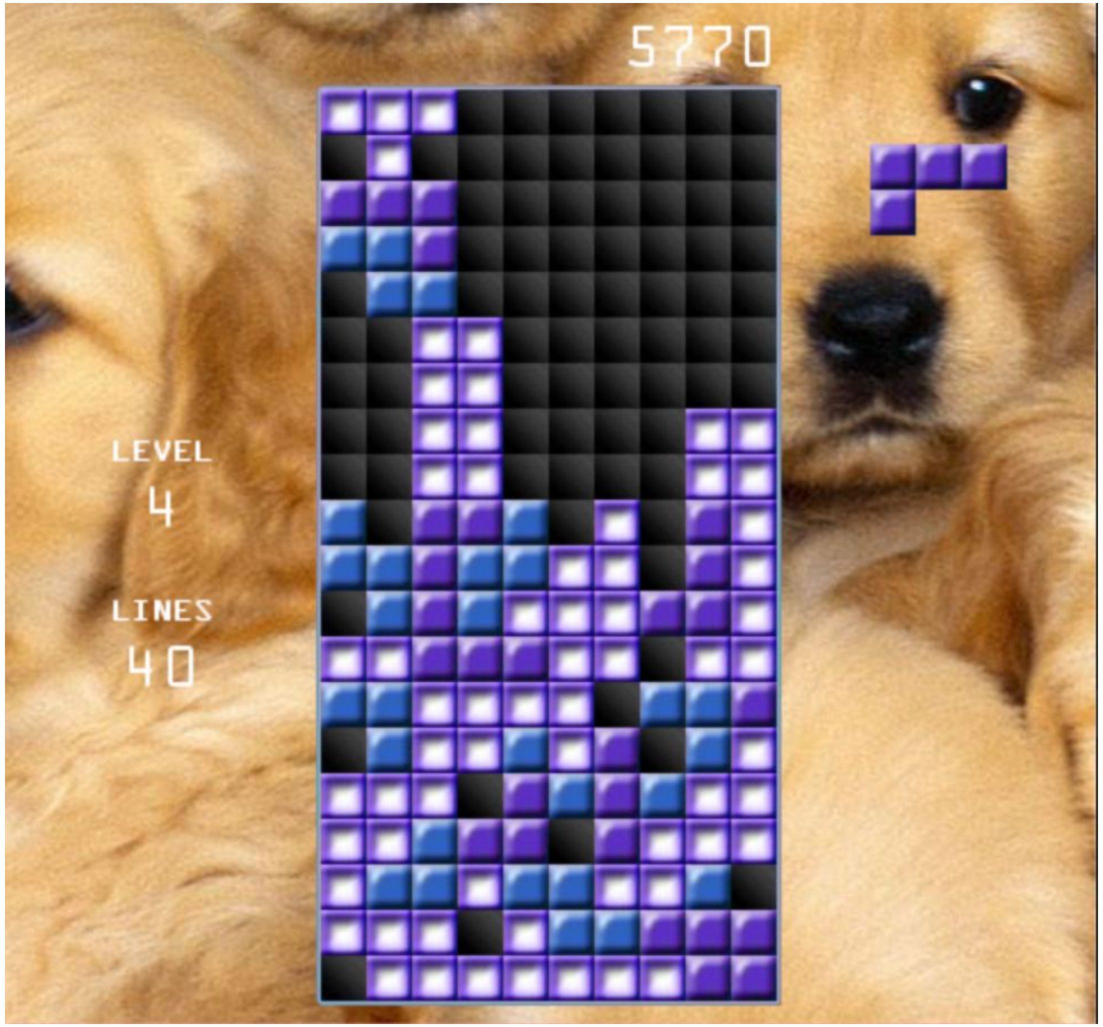
# Results MCTS



# Results BFS (depth 2)

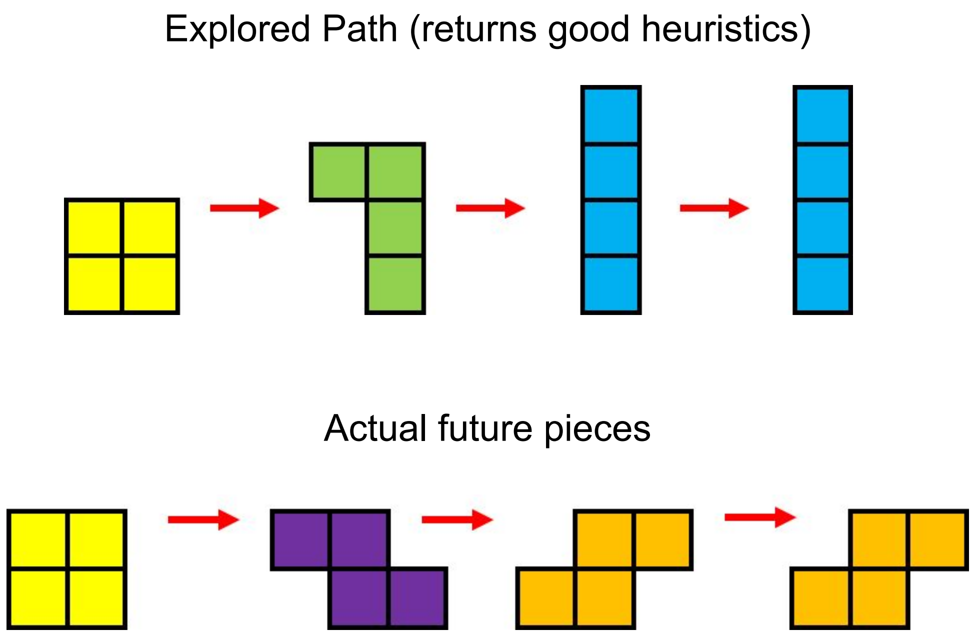


# Results BFS (depth 1)



# Discussion

- Believe MCTS biased by having random paths with ideal piece drops
  - Will move based on anticipation of future pieces
    - e.g. two straight pieces in a row
  - Tetris is stochastic
  - When pieces never come, past move is in a bad spot
    - “Counting chickens before they hatch”
  - Performance limitations



# Discussion

- BFS works since it tries to optimize the map based on current piece
  - Current heuristics try to make the best possible map
  - Only works with what it already has
  - Similar to how actual people play
- MCTS robust to pieces, rather than to map state
  - Attempts to place piece in a way to optimize random child states
  - Theoretically valid, but biased toward piece choice

# Future work

- Play in Real-time
- Could Improve MCTS
  - More Heuristics
  - Genetic Algorithms for Heuristics
    - Including MCTS *'C'* value
  - Translate to cpp instead, or use pipes
    - Python too slow
- Bug in the later stages of the game
  - Causes pieces to stack in a column to one side

# Final Comments

- MCTS catered towards scenarios when there is a clear advantage towards searching deep
    - Situations with few actions
    - Scenarios where actions can be undone based on new information
    - MCTS more suitable for games like Pac-Man
- Thank-you to Professor Tumova and Professor Jensfelt for this class and for your insight
    - You TAs were great too

# Questions?