# Senior Project Analysis Report

# Battle Bot AI – Patriot Bot

James Johnston

Professor Lynne Slivovsky

December 2015

# Table of Contents

# Overview:

The purpose of the **Patriot Bot** is to compete in **The AI Game's AI Block Battle competition**. It attempts to rate its **Tetris Fields** with a scoring algorithm that selects the best future field based upon the possible **piece placements**. I started working on this program during the summer by learning required programs to create a Battle Tetris engine. This included a **JLWGL**, which is a java implementation of **OpenGL** for display, and java thread-handling techniques. By the end of the summer I had a basic system up and running. During the school year I refined the system, added major functionality to piece management, and adjusted scoring variables. My project peaked in **17th** place during the ongoing competition.

# Functional Requirements:

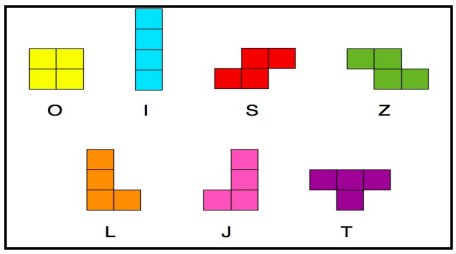

*Figure 1 – All possible tetrominoes*

**Tetris** is a single-player game played on a 10 wide and 22 tall field, the bottom 20 of which are visible. Play is divided into **rounds** containing some number of timed **steps**. At the beginning of each round, a **tetromino**, or a geometric shape consisting of 4 squares collected orthogonally, appears at the top of the field. All possible tetrominoes can be seen in **Figure-1**. During a step, the **player** can input any number of commands to either rotate the piece by 90-degress counter-clockwise or clockwise, or move the piece one space left, right, or down. If the piece can complete the command without intersection with another piece or leaving the field, it will execute. When a step's timer concludes and the piece attempts to move down one space. If this move would cause a collision with another piece or case the piece to leave the field, the piece is locked in place and the round concludes. If there are any rows that contain 10 solid blocks across at the end of a round, that horizontal line is cleared from play. All blocks above this line are then shifted down, and the score increments. A new piece then appears at the top of the field for the player to control, starting a new round. If this newly inserted piece intersects with a piece on the field, the game is over. Figure-2 shows an in-progress game of where the player controls an 'I' piece. When the piece drops 5 spaces and can drop no further, it will clear 4 lines. The bottom-right of Figure-2 shows the next piece that will appear at the top of the field. The general goal in a game of Tetris is to clear lines as quickly as possible. As the score increases, the step timer gets shorter and the pieces start dropping faster, making accurate piece placement harder. At the

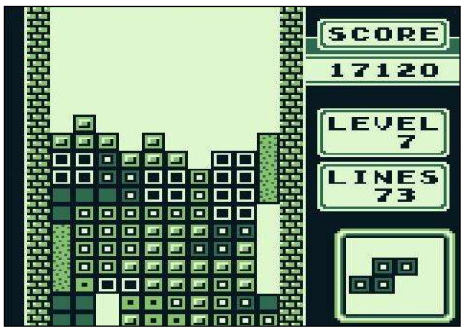

*Figure 2 – Class Tetris*

highest levels, the player has 8 frames, or a little over 1/3 seconds, to properly align their piece before it hits the bottom of the field.

**Battle Tetris** employs the same basic framework for a multiplayer game with several new rules. In this game an additional field is added for the other player. Each player has their name and score displayed above their field. Score is determined by the lines you clear and combo. Combo start at 0 and increments on every consecutive turn a player clears a line. If that player fails to clear a line, combo is set to 0. Score increases every time you clear lines according to Figure-3, plus one less than combo. So a combo score of 5 increase the score of sending a line by an additional 4 points. A T-spin is when you clear a line that can only be cleared by rotating a 'T' piece. This can be seen in Figure-4. A perfect clear happens when a player's field has no solid blocks in it at the end of a turn. A player sends 1 **garbage line** to their opponent for every 3 points of score they earn. A garbage line is a horizontal line containing 8 or 9 randomly placed solid blocks which appear at the bottom of the field and push any blocks above further upward. These are the gray lines that can be seen in BlockCrusher's field in Figure-4. Note how Stranger has scored 9 points, which resulted in the sending of 3 garbage lines. Additionally, a black, un-clearable line will appear at the bottom of the field once every 20 rounds to ensure the games end reasonably fast. There are the black lines at the bottom of both player's fields in Figure-5. Skips allow a player to skip over the current piece and not have any piece place it for the current round.

| | |
|---|---|
| Single line clear | 0 |
| Double line clear | 3 |
| Triple line clear | 6 |
| Quadruple line clear | 10 |
| Single T-spin | 5 |
| Double T-spin | 10 |
| Perfect clear | 18 |

*Figure 3 – Scoring System*

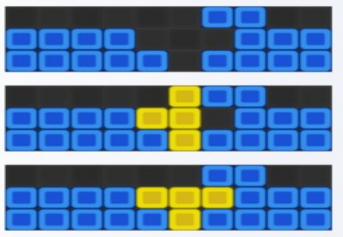

*Figure 4 – T-Spin example*


*Figure 5 – The AI Game's online Tetris system*

## Competition Details:

The AI Block Battle competition is hosted by a website called **The AI Games.** Users can submit code in their preferred langue to be compiled. The system works on a text based interface, where the user submitted code must read in field-related information and print out movement instructions. After the user selects an active version, their bot will be randomly placed in ladder matches against similarly rated opponents. Match making follows a standard **ELO rating system**. Ever bot enters the competition with a 1400 rating. Winning increases your rating while losing decreases your rating. Winning against a higher rated opponent or losing against a lower rated opponent makes the rating change more severe, while winning against a lower rated opponent or losing to a higher rated one makes it less severe. The system tries to match similarly rated bots against each other. The goal of this system to adjust ratings until bots with the same rating will each win about 50% of matches. Current ladder rankings are displayed on a leaderboard. This system will continue to automatically run games until some unannounced, future date when the final competition begins. Then the top bots will be placed into a single elimination tournament seeded from the ladder. The total prize pool is a little over $2000 for the top 8 bots is.

## Field Rating system:

All successful Tetris AI bots use a field rating system. The program looks over the field to analyze it for certain traits for all the possible piece placement locations. The different metrics are weighted to adjust a total field score, the best of which is used for piece placement. I broke my field analysis into 8 categories:

**Holes:** A hole is exactly what it sounds like. If there is a gap in the field where an empty space has a solid block anywhere above it, the hole score increments by 1. This metric is important because if a line has a hole in it, it is much harder to clear the line, leading to worse combos and a higher chance of losing by topping out.

**Bumps:** A bump is a measure of unevenness of the field. Judging from the highest solid block in each column, the bump score increments by 1 for each vertical unit of difference between each row. The goal is to make the solid portion of the field as flat as Pieces generally fit better on flat surfaces, since causing less holes and bumps that way. It also helps to avoid board states with high peaks or valleys, which are harder to place pieces in.

**Covered:** Covered is a measure of the solid blocks on top of a hole. Each solid block on top of each hole increments this score by 1. This score helps to remove holes once they are created. If many solid blocks are stacked on top of a hole, clearing the hole becomes much harder. This metric reduces the effectiveness of piece placement

**Well:** A well is a valley at least 3 units deep and exactly 1 unit wide. Deep wells, at least 5 units deep, have a multiplier attached. Wells are extremely bad because they can only be filled using the long 'I' piece. Since there is no guarantee that one of these will appear, a well makes clearing lines without causing holes difficult. The effect can quickly snowball into situations requiring multiple 'I' pieces to avoid holes. Before this metric was implemented, deep wells were the single biggest cause of lost games.

**Max:** Max scores based on the maximum height of the field. If the maximum height of the field gets too high it could prevent piece movement, and the bot is more likely to lose. There is also an offset, which increases the height at which max height starts to count. This metric reduces the effectiveness of piece placement except in cases where it allows for better movement when placing pieces.

**Lines:** The line score is incremented if a lined is cleared, in accordance with the aforementioned scoring system. Clearing lines is good since it sends lines to the opponents and prevents the bot from losing. This metric reduces the effectiveness of placement.

**T-spin:** T-spin increments in score if the piece placement would cause a T-spin. It the potential for earning skips and sending your opponent lines. This metric reduces the effectiveness of placement.

**T-adjustment**: T-adjustments increments in score if the piece placement would allow for a future T-spin. It depends heavily on the ability of this bot to clears lines faster than they are received. It significantly reduces clear speed, and if the algorithm cannot clear them faster than they arrive, the bot will lose despite sending more lines. This metric is currently turned off.

# Piece Management:

The most technical aspect of this project was ensuring accuracy of piece testing locations and movement. The program had to test each piece placement possibility, and then do the same for one move into the future. To make sure each piece could actually be placed, it would be moved to the far left of the screen then dropped into placed as it moved to the right of the screen. This handled the basic moves, but there were several more complicated actions to consider. After the piece got to the bottom, it might be advantageous to move the piece left or right. This helps eliminate overhangs, which are holes with one side exposed to open spaces. It might also want to turn a piece at bottom, allowing pieces the fit through gaps and fit into locations that would be unreachable through standard movement. At first all of these move were attempted for every single location, which increased calculation time dramatically. The base calculation took a maximum of 40, which comes for the 4 possible piece turn alignments and the 10 wide field. The analysis would have to then be calculated 40 times per initial fiend to check the second piece, leading to a maximum of 1600 calculations. The new moves increased the base number from 40 to roughly 150, thus requiring 22500 total calculations. To reduce this number, I made functions to check and see if a spin or lower movement would accomplish anything. This changed played a major role in reducing game time by 90%, as the numbers would suggest.

# Tournament Testing System:

One of the things that make this competition interesting is that there is no correct answer. There is no real way of knowing how to weight scores without testing them, or if they're are correct or relevant. Since you can only run matches on the live tournament system once every 5 minutes, I created my own game system to allow for faster testing. My tournament testing system ran my bot against itself, giving different weights to the scores and printing out results. With this approach, making games as fast as possible became important. Faster games meant more data upon which to base adjustments. Battle Tetris is a high variance game. There is a lot of randomness in the way garbage lines appear, so the system needs many trials to get accurate results. To get information about changes with an accuracy of 1-2%, the tournament needs to run at least 2000 games. When I first implemented this system, games took 3 seconds to run. After making changes in my algorithm for efficiency, games took 0.3 seconds to run. The initial adjustments from this system increased the rating of my bot from 1500 to 1800, jumping from 60$^{th}$ place to 30$^{th}$ place. It also helped to eliminate unhelpful scoring metrics, like T-adjustment. One weakness with the system is that you can only test against your own bot. If you bot was, for instance, worse at sending lines to its opponents than most others it'd be facing, it might adjust for less efficient clearing than is required. Additionally, scores may have local maxima, so testing results might discourage the more efficient weighting. Finally, scores may be related in unforeseen ways. Holes and Covered, for instance, are cleared related. Weighing holes very highly while might increase or decrease the effectiveness of a certain hole weight. There is no way to tell how interconnected all the scores are. So getting the best set of scores would theoretically involve a number of tests that scales in factorial with the number of scores.

## Results:

My bot peaked at 17th place of over 200 competing bots. Patriot Bot currently sits 34th place. Interestingly, its ELO rating, at 1980, is only slightly lower than at its peak. That means that over the past month ELO and inflated significantly. This comes as no surprise, as old players are constantly improving their bots to move the ceiling up, while a stream of new bots keep the floor down.

## Future Adjustments:

The main future adjustment is to include new scores to better rate the field. One critical one is some adjustment to the hole scores. The current system rates all holes equally, where in reality some holes are more detrimental than others. Holes located above other holes are worse because to clear them you typically end up covering the lower hole even further. Holes covered by rows that are almost complete are not as bad either. Also, the T-spin system also needs to be improved in some way to make it more practical and less detrimental to clearing speed. Finally, there needs to be some system to use and adjust for skips, which can be very powerful.