Perfect Tetris Game

Suhjune Yim

December 2020

1 Impact and Objectives

Tetris, created by Alexy Pajitnov in 1984, is a game that consists of 7 different blocks called tetrominos played in a board of 10 units wide and 20 units tall.

Although Tetris is a geometric game, there are only handful of researches or theorems with scientific proofs and most of these theorems are either in a classic Tetris game or have fixed or limited variables which cannot be observed in a standard Tetris game. For example, Heidi Burgiel's first theorem states that Tetris games that only consists of alternating S and Z-pieces will always end before placing 70,000 pieces.[1]

Since the release of the Tetris game, there has been very few research on this game, especially today. It is estimated that Tetris has been sold about 495 million copies as of 2014 but academic research has not been conducted recently.[3] I hope this project to establish further interest in Tetris beyond just playing the game but to have further research on it.

In this project, I aim to study the Tetris with modern guidelines to see if it is possible to play perfect clears constantly. I expect that keeping tracking of the number of each tetromino used will most likely result in a perfect clear.

2 Background of Research and Methodology

2.1 Background of Research

Definition 2.1. *Hold is a feature that saves currents tetromino and switches to a previously held tetromino. If none has been held, holding will save current tetromino and proceeds to play the next tetromino.* [2]

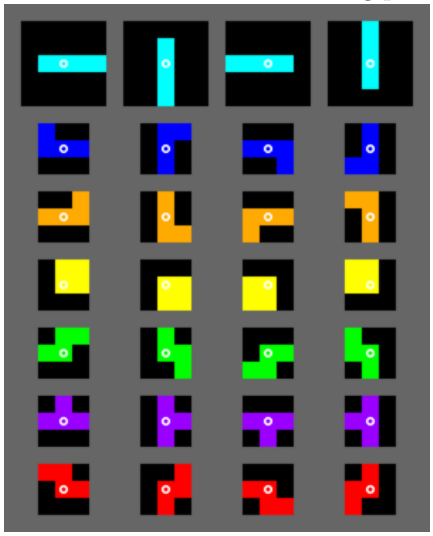
Definition 2.2. *7 bag radomizer where the game generates a bag containing all 7 types of tetrominos ordered randomly.Once all the tetrominos in a bag are played, a new bag will be generated with random order of tetrominos.* [2]

Definition 2.3. *Super Rotation System, or SRS, fixes turning points for each tetrominos which defines the behaviour of tetrominos when they spin.*[2]

Definition 2.4. *Perfect Clear, or PC, is achieved when there is no residue on the board after certain number of clearing lines.*

Before we proceed, we need deeper understanding of SRS.Each tetromino has fixed turning point:

Figure 1: Demonstration of turning point[4]



When rotated, cross product of the following vectors are applied:

$$Clockwise = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} Counter - Clockwise = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

For instance, we consider a case with two tiles: One turning point tile and another tile attached above that turning point. Since the turning point is the center, attached tile has coordinate of $(x, y) = (0, 1)$. When rotation takes place, cross product is applied:

$$\begin{pmatrix} 0 & 1 \end{pmatrix} \times \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} (0 \cdot 0) + (1 \cdot 1) & (0 \cdot -1) + (1 \cdot 0) \end{pmatrix} = \begin{pmatrix} 1 & 0 \end{pmatrix}$$

After a rotation takes place, SRS checks 5 different positions by giving offsets to the center tile.[4]

Figure 2 shows offset data for each tetromino. 0, R, 2, and L each stands for Original position, Right, Upside Down, and Left respectively.

When you rotate from R to 2 state, you apply and test for the offset data at row R. These tests will be carried out sequentially from Offset 1 to Offset 5 and the successful test will be chosen as a valid rotation.

J, L, S, T, Z Tetromino Offset Data

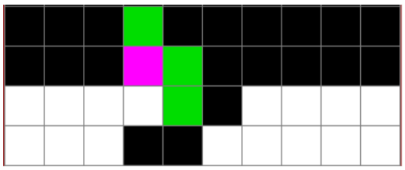
	Offset 1	Offset 2	Offset 3	Offset 4	Offset 5
0	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)
R	(0, 0)	(+1, 0)	(+1,-1)	(0,+2)	(+1,+2)
2	(0, 0)	(0, 0)	(0, 0)	(0, 0)	(0, 0)
L	(0, 0)	(-1, 0)	(-1,-1)	(0,+2)	(-1,+2)

I Tetromino Offset Data

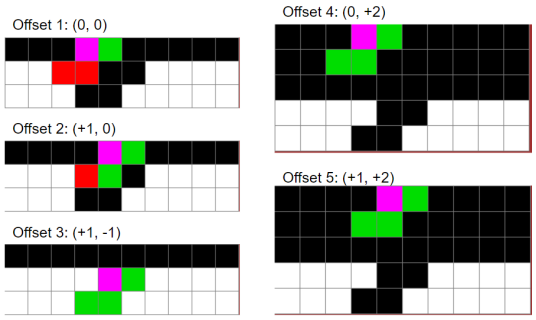
	Offset 1	Offset 2	Offset 3	Offset 4	Offset 5
0	(0, 0)	(-1, 0)	(+2, 0)	(-1, 0)	(+2, 0)
R	(-1, 0)	(0, 0)	(0, 0)	(0,+1)	(0,-2)
2	(-1,+1)	(+1,+1)	(-2,+1)	(+1, 0)	(-2, 0)
L	(0,+1)	(0,+1)	(0,+1)	(0,-1)	(0,+2)

O Tetromino Offset Data

	Offset 1	Offset 2	Offset 3	Offset 4	Offset 5
0	(0, 0)	No further offset data required			
R	(0,-1)				
2	(-1,-1)				
L	(-1, 0)				



Above figure shows an S tetromino currently at R state with purple tile as its turning point. There is no way to drop the S tetromino into that empty S-shaped space so rotation is required. Since the S tetromino is at R state, rotating in a clockwise direction will take offset data at row R from the offset data table.



Offset 1 and 2 are not valid options due to collision with existing tiles as shown in red. Offset 3, 4 and 5 do not have any collision so they pass the test. Hence, offset 3 will be chosen as valid rotation.

2.2 Methodology

My project will consider two cases: a practical Perfect Clear at the beginning and check whether Perfect Clear can be carried out endlessly. For the first case, we assume that all the pieces in the first bag will be played like the figure below.

This is one of most intuitive pattern as we only have to consider filling in the 4 by 4 area from 3^{rd} column to 6^{th} column.

We can pick 3 out of first 4 pieces from the second bag. This leads to a permutation of ${}_7P_4 = 840$. The availability for a solution of perfect clear will differ based on the first piece in the second bag. I checked all the 840 sequences and sorted the data. The result is as follows:

First piece in the second bag	No. of successful patterns	Probability of successful patterns
I	88	73.3%
L	81	67.5%
O	58	48.3%
Z	63	52.5%
T	103	85.8%
J	52	43.3%
S	50	41.7%

T tetromino had exceptionally high successful perfect clear rate compared to others but in general, the cumulative perfect clear rate would be 58.9% with 495 out of 840 sequences.

Now we need an algorithm to find perfect clear solution to check the result above and to solve for solutions for our second case. In the second case, there is no limitation on the play field so there will be a lot of possible sequences. To have a brief idea, we first compute for the number of sequences of tetrominos when there is no hold feature:

A total of 40 units have to be filled so we need 10 tetrominos while the 7 bag randomizer act as a number system with base 7. The 1^{st} bag will be played empty and we can choose up to 3 pieces from the 2^{nd} bag. Then we are left with 4 tetrominos in the 2^{nd} and need 6 more from 3^{rd} bag. We follow this pattern and we obtain the following result without using hold feature:

$$\sum_{n=1}^{7} x_n! \cdot 7Py_n = 2,399,040$$

where we define $x_n = 7 - y_{n-1}$ and $y_n = 10 - x_n$ with initial value of $y_0 = 0$.

There are total of 7 different tetrominos that can be held and there are total of 10 places in the sequence that the hold can be used. Hence from the result above, we need to multiply the following:

$$7 \times 10 \times \sum_{n=1}^{7} x_n! \cdot 7Py_n = 163,732,800$$

There will be some duplicate sequences and there is also a chance that the hold is used to swap with the same type of tetromino. We can approximate that there are about 100 million sequences.

For the algorithm, Depth First Search (DFS) binary tree is used to check for perfect clear solutions. It searches for solutions by placing each piece in every position and iterate the same process for all the rotation state. Since we are considering a perfect clear after 4 lines, we limit the grid size to be 10 units wide and 4 units high. Within the algorithm, we will represent the turning point as 2 and the remain tiles as 1 in the matrix representation. For example, J and Z tetrominos will have the following matrix representations:

$$J = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} Z = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix}$$

The first case had the average of 58.9% successful rate so I expect that success rate for second case will be at least equal or even higher. Hence we let the algorithm to terminate once it has found at least one solution for current sequence. This will make the algorithm to fully run only for sequences that cannot end up in a perfect clear.

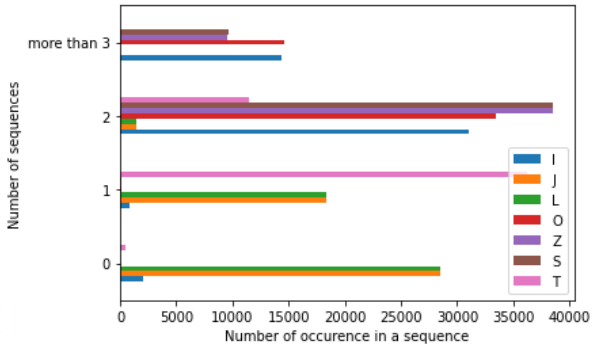
3 Results and Discussion

The algorithm could not finish within its run time for there were too many computation needed and there was not enough memory for it. On the other hand, the first case where we only have 840 sequences, the results were the same which verifies that my manual search for solutions were correct and it also means that the algorithm is working as intended.

Since the algorithm could not run properly, I fixed first three tetrominos of a sequence and managed to run a total of 81,365,201 sequences and obtained 48,232 sequences without a PC solution. This is about 99.9407% of successful perfect clear rate. From the non-PC sequences, I found an interesting pattern that many sequences were ending with a T tetromino and it turns out that 45,308 out of 48,232 sequences were ending with a T tetromino. I also checked to see if there is any relationship between in the number of occurrence of each tetromino in a sequence and being a non-PC sequence. I took the data and counted how many times each tetromino appears in a sequence.

Non-PC sequences tend to have more O, S and Z tetrominos while having less J and L tetrominos. I thought this was because, when converted into 2 by 3 matrix, S and Z tetrominos have empty spaces separated while J and L tetrominos have theirs placed together. However, I realized that this is not true for O and T tetrominos. O tetromino, despite having its empty spaces together, it showed similar pattern as S and Z tetrominos while vast majority of sequences had only 1 T tetromino in a sequence.

	I	J	L	O	Z	S	T
0	2028	28447	28448	0	0	0	443
1	829	18351	18361	132	140	114	36269
2	30975	1434	1423	33464	38501	38513	11520
more than 3	14400	0	0	14636	9591	9605	0



4 Conclusion

There were some limitation of my project. My algorithm could not finish running within its run time which can be improved by optimizing the algorithm. One of the most effective way would be making the algorithm to check for multiple solutions simultaneously. Moreover, there are many ways to search for a perfect clear solution. Although I used DFS binary tree in the hope of computational speed, with proper optimization, breadth first search would be more appropriate if we were to obtain all the possible PC solutions for any sequences.

Despite 99.9% success rate of PC after 4 lines, there is a slight chance that does not have a solution for performing a PC. This is because in my project, I set the limit to be exactly 4 lines. Once we remove that limit and allow various numbers of line clears required for a perfect clear, I believe it is theoretically possible to perform perfect clears endlessly. For instance, any combination of a sequence containing JLOOI will always have a 2-line perfect clear. If a certain sequence does not have a PC solution after either 2 or 4 lines, extending the line clear limit to 6 or 8 will open up the sequence to have at least one perfect clear solution.

References

[1] Burgiel, Heidi. "How to Lose at Tetris." *The Mathematical Gazette*, vol. 81, no. 491, 1997, pp. 194–200. *JSTOR*, `www.jstor.org/stable/3619195`.

[2] "Tetris Guideline." *Hard Drop*, `harddrop.com/wiki/Tetris_Guideline`.

[3] Sirani, Jordan. "Top 10 Best-Selling Video Games of All Time." *IGN*, 19 May 2020, `www.ign.com/articles/2019/04/19/top-10-best-selling-video-games-of-all-time`.

[4] "SRS." *Hard Drop*, `https://harddrop.com/wiki/SRS#How_Guideline_SRS_Really_Works`.