Pranjal Modi 6/9/23
AP Computer Science A Period 1

Tetris in Processing: Final Document

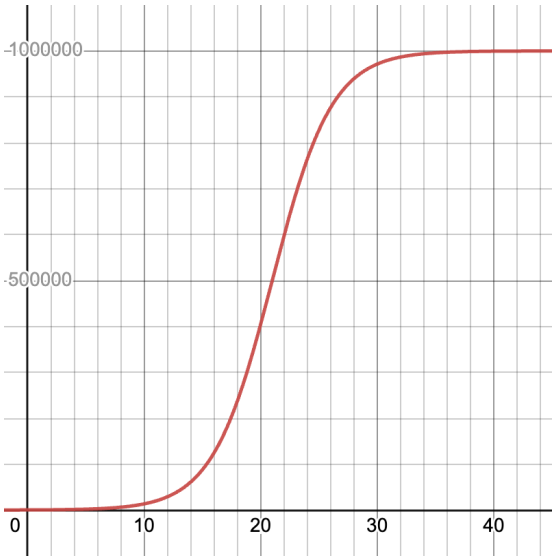
I. Description

Group Members Names: Pranjal Modi.
Group Name: Tetris Production.
Project Title: Tetris in Processing.
Class Period: Period 1.
Full Description: My project is a recreation of Tetris with the Processing programming language; it contains many of the original game's functionalities, such as an integrated starting screen, a high score tracker, and a proper block-dropping algorithm. It also contains some additional features, including the progression game mode, whose level-based nature differs from the score-based original version. To maintain faithfulness to pure Tetris, though, the normal rule set, where the game continues until new blocks cannot fit on the board, is also included under the arcade game mode. Regardless of the chosen game mode, however, certain functionalities and rules are preserved across the entire project; full support for *rotations*, *horizontal movement*, *board generation*, *trajectory outlining, scoring*, and *block updating*, for instance, has been implemented into the game as a whole, and is available for all users. Various display systems are also available to improve communication with the user; stop and pause buttons are stationed in the top-left and top-right corners of the screen, respectively, while a panel on the center-right of the window displays the next three blocks to be dropped. Furthermore, the potential ending location of the current block is outlined on the board, which updates in response to users' changes to the block's location and/or orientation. Score and level indicators are also displayed when necessary, providing a means for a player to determine their level of progress. Considering that the game ends when a newly spawned block cannot fit onto the board, users must have a means of generating more free space on the screen; to that end, a *line-clearing* functionality has also been added; once a horizontal line has been filled with blocks, it turns white for a short time and then disappears, shifting all of the above blocks down by one square. These cleared lines need not be adjacent, and up to four lines can be removed at a time (as the maximum length of any implemented block is four); these clears are also crucial to increasing the overall score, as the number of points awarded for them increases exponentially (with a base of 20) based on the number of simultaneous clears. Thus, this feature allows the user to better utilize their skill to achieve in-game progress; this is exceptionally important for the progression mode, where the score required to reach the next level increases logistically. Points are also awarded for successfully dropping a block, although these only increase linearly based on the size of the object (ten points per filled square). Once a game ends (either due to a loss or a willing exit), the user will be returned to the starting screen, with a potentially updated high score and the opportunity to play again. At this stage of completion, glitches have been almost completely eliminated, and any potential player can run the program without difficulty; in keeping with that, the various game components (i.e. game board, starting screen, and transition screens) are also

heavily integrated, making the project more cohesive. Visually, a variety of gradient-based animations are displayed on the starting screen, along with an original Tetris logo, which also contributes to the clean atmosphere of the overall game. In all, this game has a wide range of functionalities, with some hewing to the original Tetris game (i.e. block rotation and movement, scoring, user displays, starting screen, user input, board generation, etc.) and others serving as new additions to the original (i.e. multiple game modes, level-based progression system, animations, etc.), making it a new product in its own right. ***No additional libraries were used in this project.***

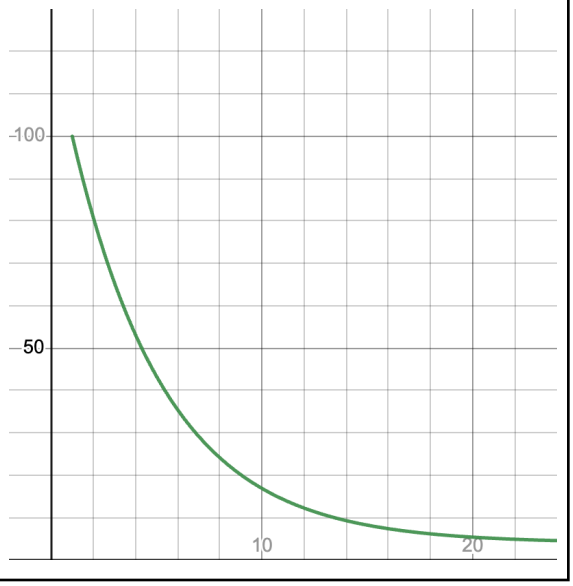
Additional Information: With regards to the level-based progression game mode, the graphs and functions for the score requirement and level speed are shown below, with the input representing the level number (referred to as the *seed* in the program). Note that a lower *speed value* actually represents a *faster* overall game based on how the project was produced. Furthermore, note that all the produced values are truncated to integers before being utilized in the game itself. Level numbers start at one.

- Score Requirement:

Function	Graph
$s(l) = \frac{999999}{1 + e^{-0.391182247(l-21)}}$	

Note: The corresponding information for the speed function is on the next page due to space restrictions.

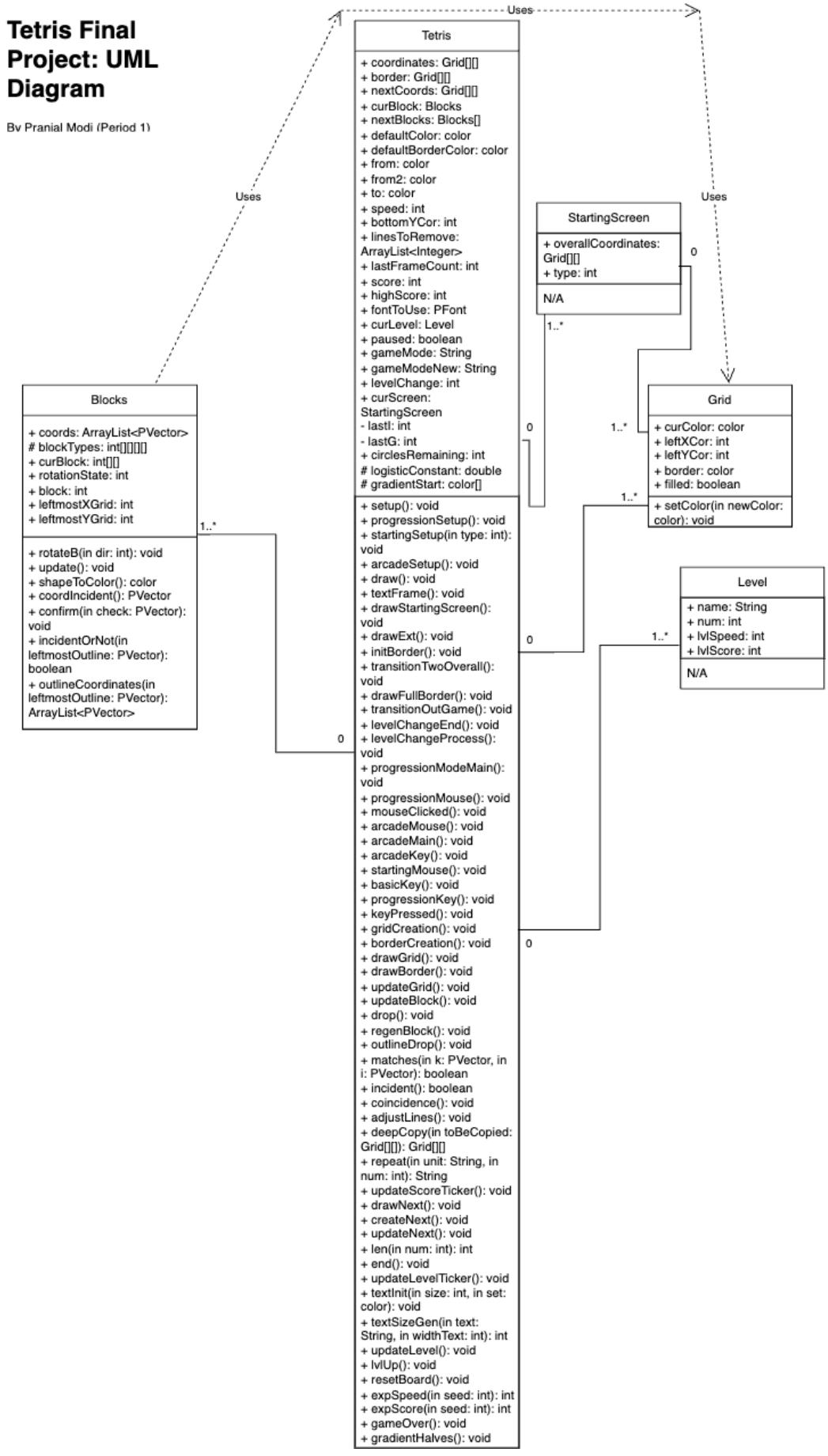
- Level Speed:

Function	Graph
$v(l) = 96(1.25)^{-1(l-1)} + 4$	

II. Log: Not Included—Individual Project

Note: The UML Diagram is on the next page due to space restrictions.

III. UML Diagram



IV. How does it work?

After clicking the “Run” button on the processing interface itself, the game window will be displayed; at runtime, a starting screen is displayed, which consists of a gradient-based animation, a Tetris logo, a high score display, and a green play button. Once the user is ready to start an actual round of Tetris, they should click the play button with their mouse, which will lead to another interim screen. This screen consists of two gradient-based animations, each of which surrounds a string of text (either “Arcade Mode” or “Progression Mode”); the objective of “Arcade Mode” is to achieve the highest possible score, while the objective of “Progression Mode” is to achieve the highest possible level. After choosing between these options, the user must click on the animation that corresponds to their desired game mode, as indicated by the displayed text. After this is complete, a short transition animation will play, after which a round of the chosen game mode will be initialized and started. Clicking the **left** and **right arrow keys** will move the block in the corresponding direction, and holding down the **down** or **up arrow keys** will speed up the current block; clicking on a non-button part of the screen will instantly drop the current block to its inert position. Furthermore, using the top-left red button will stop the game and force a return to the starting screen, while using the top-right gray button will pause or unpause the game, depending on its current state. Holding the **shift** and **A** or **D** keys *simultaneously* will rotate the block clockwise or counterclockwise, respectively. Note that some rotations may not be processed due to invalid space, in which case the block will not appear to visibly change; there is also a varying number of rotation states per block, so rotation mechanics may seem different in different situations. For progression mode, once the user *exceeds* the posted score cutoff at the top of the screen, the displayed text will turn green and they will be sent to the next level after a short delay. Looking at the center-right display will provide the identities of the next three blocks, while the bottom display indicates the current score. Once the user ends the game via the top-left button or by reaching the top of the screen, the game will end and the starting screen will be displayed once again after a short interim state. From this point, they can choose to close the game window or start another game.