Demo of Final Project: Biquadris

Kevin Jin, Yibo Ma, Yiran Sun
CS246 Spring 2021
University of Waterloo
August 13, 2021

Directory

- The arguments of all demos that are not specifically declared with special flags are ./biquadris by default. To save time, the program can be run with a -text flag to display text only.

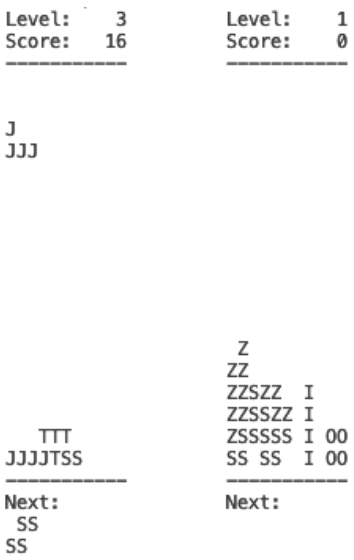
Interface

We have provided two types of interface for the game: text-based display and graphical display.

In both modes, the current block is generated in the upper left corner of the game board(18 rows*11 columns, with 3 reserved rows). All the current blocks will be located in the game board as well. The next block to come will be exhibited in the lower left corner of the board, and the scoreboard in a single window. The block types are colour-coded, each type of block being rendered in a different colour as shown below.

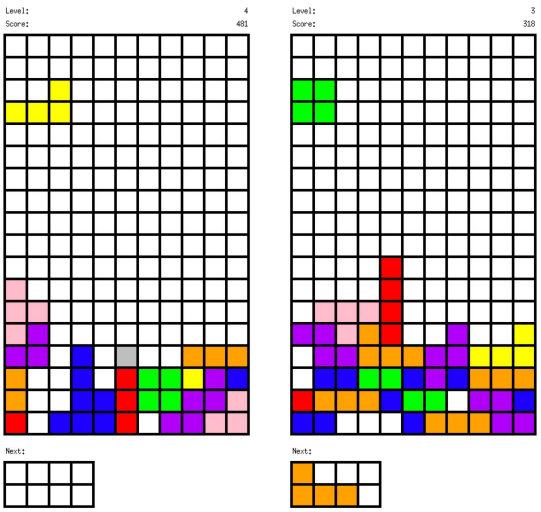
At the top of the game board, there are two additional rows: Level Row, which represents the game levels of the two players, and Score Row, which exhibits the current scores of the two players.

Text-based Interface Demo:



In this demo, the player who is playing the game now is player 1, with the game board on the left. The difficulty level is 3 and his score is 16. Player 1's current block is an L-block, and the next block that is coming is an S-block. Player 2's game level is 1, and the current score is 0, and nothing is displayed in the lower left corner of his/her board because it is not his/her turn now.

Graphic display Interface Demo:



We can see that player 2 is playing with the current block since the left corner below the board is showing the upper coming block. The I-block on player 1's board is already dropped and player 1 has an L-block as the

coming block. Different blocks are displayed using different colors. Above the board, the current level and the current score is recorded and displayed (now both players are at level 0 with scores initialized with 0).

Note that we use grey to represent the 1x1 block falls in level 4. We believe that this is a better colour since brown is easy to mix with red, which is the colour for an I-block. Grey is a clearer color for players to notice and identify that they have not cleared 1 row after dropping 5 blocks. We all agree that Grey is a more suitable colour than Brown. Please do not deduct our mark for this reason! Changing color is really easy and we could handle it!

Command-Line Options

-text: Runs the program in text-only mode. No graphics are displayed. The default behaviour (without -text) is to show both text and graphics. Note, under this command, the program will be displayed faster..

-seed [number]: Sets the random number generator's seed to [number]. If you don't set the seed, you always get the same random sequence every time you run the program.

- Without command-line argument:

  Arguments: *./biquadris*

  Inputs: *levelup levelup drop drop drop quit y*

- With command-line argument:

  Arguments: *./biquadris -seed*

  Inputs: *10 levelup levelup drop drop drop quit Y*

  The first game shows what happens if we do not have this command line argument. Everytime when generating the first block in level 2, the block will always be the I-block. However, in the picture below, we generate a block using seed 10 as command line argument, and the generated block becomes Z-block.

-scriptfile1 [file]: Uses [file] (if valid) instead of sequence1.txt as a source of blocks for level 0, for player 1.

- Without command-line arguments:

  Arguments: *./biquadris*

  Inputs: *7drop 7drop quit Y*

- With command-line arguments:

  Arguments: *./biquadris -scriptfile1 easy1.txt*

  Inputs: *7drop 7drop quit y*

  Note that if the end of the file is reached, the file is read again from the beginning -scriptfile1 [file] does not affect the default file for player 2.

- With invalid files:

  Arguments: *./biquadris -scriptfile1 empty.txt*

  Arguments: *./biquadris -scriptfile1 invalid.txt*

  Arguments: *./biquadris -scriptfile1 doesnotexist.txt*

-scriptfile2 [file]: Uses [file] (if valid) instead of sequence2.txt as a source of blocks for level 0, for player 2.

- Without command-line arguments:

  Arguments: *./biquadris*

  Inputs: *7drop 7drop quit y*

- With command-line arguments:

  Arguments: *./biquadris -scriptfile2 easy1.txt*

  Inputs: *7drop 7drop quit Y*

  Note that if the end of the file is reached, the file is read again from the beginning. -scriptfile2 [file] does not affect the default file for player 1.

- With invalid files:

  Arguments: *./biquadris -scriptfile2 empty.txt*

  Arguments: *./biquadris -scriptfile2 invalid.txt*

  Arguments: *./biquadris -scriptfile2 doesnotexist.txt*

-startlevel n: Starts the game at level n. The game starts in level 0 by default if this option is not supplied.

- Without command-line arguments:

  Arguments: *./biquadris*

- With command-line arguments:

  Arguments: *./biquadris -startlevel 1*

  Note that the block is generated differently and the level on the top of the board becomes 1.

Multiple Combinations for Command-Line Options Demo:

Arguments: *./biquadris*

- Starts the game in level 0 with a graphical interface. Blocks are generated by default according to sequence1.txt and sequence2.txt.

Arguments: *./biquadris -text -startlevel 3*

- Starts the game in level 3 with a text-based interface. Blocks are generated by random.

Arguments: *./biquadris -startlevel 1 -seed 5 -text*

- Starts the game in level 3 with a text-based interface. Blocks are generated by random seed 5, i.e. the first block for player one will always be a J-block and the second block for player one will always be an I-block.

Arguments: *./biquadris -scriptfile1 file1.txt*

- Starts the game in level 0 with a graphical interface. Blocks for player 1 are generated according to file1.txt, while blocks for player2 are generated according to sequence2.txt.

Game Rules

Scoring: The game is scored as follows:

1. When a row or multiple rows is cleared: when a line (or multiple lines) is cleared, you score points equal to (your current level, plus number of lines) squared. (For example, clearing a line in level 2 is worth 9 points.)

2. When a block is completely eliminated: when a block is completely removed from the screen (i.e., when all of its cells have disappeared) you score points equal to the level you were in when the block was generated, plus one, squared. (For example if you got an O-block while on level 0, and cleared the O-block in level 3, you get 1 point.)

   Inputs: *sequence level0_clear2row.txt drop blind*

   - The score for player1 now is (2 + 0) ^ 2 + 4 * (0 + 1) ^ 2 = 4 + 4 = 8, since two rows have been cleared in level 0 and 4 blocks that have been generated in level 0 have been completely cleared.

Special Effects: If a player, upon dropping a block, clears two or more rows simultaneously, a special action is triggered. A special action is a negative influence on the opponent's game. When a special action is triggered, the game will prompt the player for his/her chosen action. Available actions are as follows:

1. **blind:** The player's board, from columns 3-9, and from rows 3-12, is covered with question marks (?), until the player drops a block; then the display reverts to normal.

   Inputs: *sequence level0_clear2row.txt drop blind drop*

   - The blind effect is turned on. Player 2's board is covered with "?" 's. After player 2 has dropped a block, the blind effect is turned off.

2. **heavy:** Every time a player moves a block left or right, the block automatically falls by two rows, after the horizontal move. If it is not possible for the block to drop two rows, it is considered to be dropped, and the turn ends.

   Inputs: *sequence level0_clear2row.txt drop heavy* down down down

   - The heavy effect is turned on. This first two down's each causes the block to move down for 3 rows, and the last down ends player2' s turn . Since with the heavy effect, when a block touches the bottom, it automatically ends the player's turn.

3. **force:** Change the opponent's current block to be one of the player's choosing. If the block cannot be placed in its initial position, the opponent loses. (i.e. force Z)

   Inputs: *sequence level0_clear2row.txt drop force T*

   - The force effect is turned on, the next block that is coming for player 2 is now a T-block.

Game End: The game will end when there is no room for one player to generate a new block:

Inputs: *sequence commands.txt drop drop*

- The game ends because there is no room for player 1 to generate a new O-block. When the game ends, a message is displayed to indicate which player wins the game or if the game ends with a tie. The highest score throughout the game is also displayed, together with player 1's score and player 2's score. Note that the highest score is not necessarily equal to the higher score between player 1 and player 2.

- Once the game ends, the player who has a higher score between the two wins the game. Both players' games end at the same time since we do not want the other player to wait for a longtime. We now live in a society which has a fast paced lifestyle. For the graph display, the window simply shuts down as the game ends.

Levels of Difficulty

The game supports the following difficulty levels:

- **Level 0:** Takes its blocks in sequence from the files sequence1.txt (for player 1) and sequence2.txt (for player 2), or from other files, whose names are supplied on the command line. If you get to the end of one of these files, and the game hasn't ended yet, begin reading the file again from the beginning. This level is non-random, and can be used to test with a predetermined set of blocks.

- **Level 1:** The block selector will randomly choose a block with probabilities skewed such that S and Z blocks are selected with probability 1/12 each, and the other blocks are selected with probability 1/6 each.

- **Level 2:** All blocks are selected with equal probability.

- **Level 3:** The block selector will randomly choose a block with probabilities skewed such that S and Z blocks are selected with probability 2/9 each, and the other blocks are selected with probability 1/9 each. Moreover, blocks generated in level 3 are "heavy": every command to move or rotate the block will be followed immediately and automatically by a downward move of one row (if possible).

- **Level 4:** In addition to the rules of Level 3, in Level 4 there is an external constructive force: every time you place 5 (and also 10, 15, etc.) blocks without clearing at least one row, an 1x1 block (indicated by * in text, and by the colour brown in graphics) is dropped onto your game board in the centre column. Once dropped, it acts like any other block: if it completes a row, the row disappears.

  Inputs: *3levelup right*

  - The block moves one column to the right, then moves one row downward since it's in level 3.

  - Note if 4 blocks generated in level 4 have been dropped without clearing any rows but the level is changed, the next time the level is set back to 4, the count does not start from the beginning, but starts from 4. This means that if the next block is dropped without clearing any rows, an 1x1 block will fall.

  Inputs: *4levelup sequence level4.txt drop*

  - An 1x1 block falls in the middle column, since player 1 has dropped 5 blocks without clearing any row.

Commands Interpreter

left: Moves the current block one cell to the left. If this is not possible (left edge of the board, or block in the way), the command has no effect.

Inputs: *ri right righ left left lef*

- The block moves to the right three times and then moves to the left three times.

Inputs: *left lef*

- Both operations have no effect because the block is already on the far left of the board.

Inputs: *right right 3levelup left*

- The block moves to the right 2 times, then the block moves one cell to the left, and moves one cell down, since the current level is 3.

right: Moves the current block one cell to the right.

Inputs: *rig ri right*

- The block moves to the right three times.

Inputs: *20rig*

- The block moves all the way to the right edge of the board.

Inputs: *3levelu right*

- The block moves one cell right, and moves one cell down, since the current level is 3.

down: Moves the current block one cell downward.

Inputs: *dow down do*

- The block moves down three times.

Inputs: *20dow*

- The block moves all the way to the bottom.

Inputs: *4levelu do down*

- The block moves 4 rows down, since the current level is 4.

Inputs: *3levelup 3dow*

- The block moves 4 rows down, since the current level is 3.

clockwise: Rotates the current block 90 degrees clockwise. If the rotation cannot be accomplished without coming into contact with existing blocks, the command has no effect.

Inputs: *clockwise*

- The block rotates clockwise.

Inputs: *sequence anti_rotate.txt clockwise*

- Invalid operation, since there is no room for the I-block to rotate clockwise.

counterclockwise: Rotates the current block 90 degrees counterclockwise.

Inputs: *counterclockwise*

- The block rotates counterclockwise.

Inputs: *sequence anti_rotate.txt counterclockwise*

- Invalid operation, since there is no room for the I-block to rotate counterclockwise.

drop: Drops the current block. It is (in one step) moved downward as far as possible until it comes into contact with either the bottom of the board or a block. This command also triggers the next block to appear. Even if a block is already as far down as it can go (as a result of executing the down command), it still needs to be dropped in order to get the next block.

Inputs: *dr*

- The current block is dropped.

Inputs: *5drop*

- 5 blocks are dropped in a row from their initial positions.

levelup: Increases the difficulty level of the game by one. The block showing as next still comes next, but subsequent blocks are generated using the new level. If there is no higher level, this command has no effect.

Inputs: *2levelup leveldo*

- The board level is changed to level 2, then changed again to level 1.

leveldown: Decreases the difficulty level of the game by one. The block showing as next still comes next, but subsequent blocks are generated using the new level. If there is no lower level, this command has no effect.

Inputs: *levelu leveld leveldown*

- The board level is changed to level 1, then changed back to level 0. The last leveldown has no effect because the level is already set to 0 and cannot be decreased.

Inputs: *10levelup*

- Invalid operation, since level needs to be in the interval of 0-4

norandom file: Relevant only during levels 3 and 4, this command makes these levels non-random, instead taking input from the sequence file, starting from the beginning. If a file is empty or contains no valid block types, then an error message will be produced indicating that the file is invalid.

Inputs: *norandom sequence3.txt*

- Invalid operation, since this command is not valid for level 0, 1 or 2.

Inputs: *3levelup norandom sequence3.txt*

- Blocks will be generated according to the file sequence3.txt from now on.

Inputs: *3levelup norandom sequence3.txt 2leveldown*

- Level 3 is set to non-random but it does not affect level 1.

Inputs: *3levelu norandom empty.txt leveup norandom invalid.txt noran doesnotexist.txt*

- Invalid files are given, the block generator still generates blocks by default instead of reading in the invalid files.

random: Relevant only during levels 3 and 4, this command restores randomness in these levels.

Inputs: *3levelup rand*

- Nothing happens since the level is set to 3, blocks are randomly generated by default.

Inputs: *3levelup nor sequence3.txt drop dr ran*

- Randomness is restored in level 3.

sequence file: Executes the sequence of commands found in file.

Inputs: *sequence anti_rotate.txt*

- Runs all the commands in the file anti_rotate.txt

I, J, L, O, S, Z, T: Replace the current undropped block with the stated block type. Heaviness is determined by the level number. Note that, for heavy blocks, these commands do not cause a downward move.

Inputs: *seque anti_rotate.txt L*

- The current block is changed to an L-block.

restart: Clears the board and starts a new game, while the highest score will be recorded.

Inputs: *7dr restart*

- The game board of the current player is cleared and the score of the player is reset to 0.

rename [old command name] [new command name]: Renames an existing command. If a command does not exist, it cannot be renamed. A command cannot be renamed to an existing command unless they are the same (i.e. rename left left). If the rename is successful, the old command name will no longer be valid.

Inputs: *rename counterclockwise ccw*

- The counterclockwise command will no longer be valid, instead, players can call ccw to rotate the block counterclockwise.

Inputs: *rename ccw counterclockwise*

- This operation is invalid since ccw is not an existing command.

Inputs: *rename counterclockwise hint*

- This operation is invalid since hint is an existing command.

hint: Provides a hint for the current player.

Inputs: *hint*

- A hint message about playing the game is provided to help the player.

help: Displays a list of available commands.

Inputs: *help*

- A command guide is displayed.

quit: Ends the game with a confirmation from the player and displays the necessary information to determine the winner of the game.

Inputs: *sequence level0_clear2row.txt drop quit Y*

- The game ends and player1 wins.

Multiplier Prefix [prefix][command]: Commands can take a multiplier prefix, indicating the number of times that the command should be executed. It is not valid to apply a multiplier to the restart, hint, norandom, or random commands. (If a multiplier is supplied, it would have no effect).

Inputs: *5rig 3lef rename clockwise cw 5cw*

- This operation will cause the block to first move 5 times to the right, then move 3 times to the left, then rotate clockwisely for 5 times.
- Note that 5right, 3left, 5cw are all considered as 1 move.(instead of 5, 3, 5). Hence in the level 3,4 heavy mode(not special action), the block will drop down by one cell if available. In the mode of heavy special action, when calling right or left with a multiplier, the block will drop down two cells if the current level is not 3 or 4, and 3 cells if the current level is 3 or 4.
- When the multiplier is applied on the drop command, we do not want 1 player to control the fate of the other player. Hence, for the fairness of the game, suppose that player1 types in 6drop, then his turn only ends after he drops 6 blocks. If he/she does not have space for blocks during the dropping of 6 blocks, the game ends, and the winner will be the one who has higher score (both players' games end at the same time since that we do not want the other player to wait for a long time).

Suggestions for Invalid Commands: If an invalid command is called, the program will display a suggested command by searching for the command that is closest to the given command.

Inputs: *le r colckwi nra own ctcls*

- Suggestions are given accordingly: left, right, clockwise, norandom, down, clockwise.

Complex Inputs with Sequences of Commands:

Inputs: *sequence commands1.txt drop blind seq commands2.txt restart sequen commands3.txt rest sequence commands4.txt drop heavy sequence commands5.txt sequence commands6.txt drop drop*

- *sequence commands1.txt drop blind*: clears two rows and triggers special effect, the score is calculated as (2 + 2) ^ 2 + 1 * (0 + 1) ^ 2 + 3 * (4 + 1) ^ 2 = 16 + 1 + 75 = 92
- *seq commands2.txt restart*: restarts the game for player 2, level and score are set back to 0
- *sequen commands3.txt rest*: restarts the game for player 1, level and score are set back to 0
- *sequence commands4.txt drop heavy*: clears two rows and triggers special effect, the score is updated accordingly
- *sequence commands5.txt*: performs movements and rotations with effect heavy
- *sequence commands6.txt drop drop*: player 1's board has no space for generating a new O-block, the game is ended