ECE 462 Object-Oriented Programming using C++ and Java

GPA Specification for Stage 3*

Fall 2009

Purdue University, West Lafayette

Note: This document is subject to revision. In case a major change is applied, you will be notified by a Blackboard discussion thread in the "Q&A on GPA Specification" section to download an updated revision. The revision number can be seen at the footnote of this page. Please always refer to the newest revision of this documentation.

1 Coordinate Systems

1.1 Playfield

The size of the playfield is 20 blocks in width and 30 blocks in height. A location in the field is denoted by coordinate (x, y). The coordinate $(0, 0)$ corresponds to the top-left corner block in the playfield, and $(19, 29)$ corresponds to the lower-right corner block.

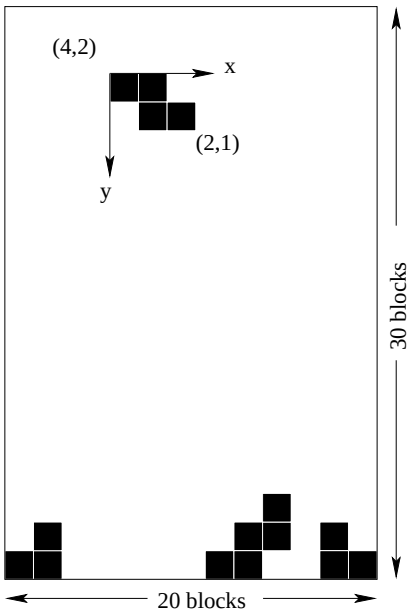

Fig. 1. Tetris playfield and tetris piece

* Revision 0.7 updated on October 16, 2009

1.2 Piece

A Tetris piece is located in the playfield by the coordinate of its top-left corner in the playfield. In addition, a Tetris piece has its own relative coordinate system within the piece.

For example in Figure 1, the falling Z-shaped piece is located at $(4, 2)$ in the playfield. This indicates that the top-left corner block of the piece is at $(4, 2)$ in the playfield (fifth column from left, third row from top), yet within the piece its coordinate is $(0, 0)$.

The *width* of the piece is defined to be the minimum number of columns it occupies (no rotation is involved here). In Fig. 1, the width of the falling piece is 3; the *height* is defined likewise but with respect to the minimum number of rows, and therefore the height of the falling piece in Fig. 1 is 2.

2 Game Conventions

2.1 Initial location for new pieces

After the fixation of a previous piece, a new piece emerges at the top of the playfield (i.e. the y-coordinate value is 0). The x-coordinate value is calculated using the following formula:

$$x = \left\lfloor \frac{20 - w}{2} \right\rfloor$$

where w is the *width* of the piece (see definition in Section 1.2). The mathematical notation $\lfloor p \rfloor$ means the largest *integer* that is less than or equal to real number p. The number 20 in the formula simply denotes the width of the playfield. The result of the calculation is to justify the piece at the center horizontally.

2.2 Rotation

In this assignment, a rotate operation of a Tetris piece is define to be a clockwise rotation by 90°. A counter-clockwise 90° rotation must be achieved by three such rotation operations. Likewise, 180° rotation is done by performing clockwise rotation twice.

A rotated piece should be aligned to the top-left most position, meaning that at least one block in the piece has x-coordinate value equals to 0 and at least one block (may not may not be the same block) has y-coordinate value equals to 0. Fig. 2 gives an example of rotation. It is not necessary that a block lies at $(0, 0)$ in the piece however, as shown in this example.

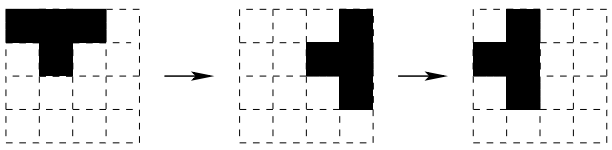

Fig. 2. Rotation of a Tetris piece

Rotate operation does not change the location of the piece in playfield, that is, the location of the top-left corner of the piece in playfield. It is consequently impossible for a falling piece to "rotate its way up".

A rotate operation will not be admissible if it causes...

- any block inside the piece to go out of the boundary of the playfield, or
- any block inside the piece to overlap an existing block fixed in the playfield.

Unlike most Tetris versions, in this assignment, rotate operations are always allowed as long as the destination can accommodate the rotated piece, even if there seems to be collision during the rotation.

3 Network Protocol

3.1 Establishing connection with server

The port number used by the server is 9462. Your program must provide user with a dialog or an input box so that the address of the server may be entered. The address could be either an IP address or a DNS name.

After being successfully connected to the server, the client immediately receives a string "`ID?\n`". The client should respond with a string "`ID=`*playerid*`\n`", where *playerid* should be letter 'g' immediately followed by your group number in Stage 3.

The next expected question from server is "`GAMETYPE?\n`". If the client wants to play with a network opponent, the response should be "`GAMETYPE=tetris\n`"; if the client wants to enter the single-player algorithm test, the response should be "`GAMETYPE=tetris-qualifier\n`". This choice is determined by the current game mode. Your program should be able to handle both of them to fulfill the requirements in the specification on the web page[1].

After successfully reponding to the above two queries, the client now expects to see an acknowledging message "`ACCEPTED.\n`" from the server.

Table 1. Summary of server messages and responses

Server Message	Client Response	Condition
`ID?\n`	`ID=`*playerid*	
`GAMETYPE?\n`	`GAMETYPE=tetris`	(for vs play)
	`GAMETYPE=tetris-qualifier`	(for algorithm test)
`ACCEPTED.\n`	(no response)	

3.2 Initializing game

After receiving "`ACCEPTED.\n`", the client now listens to the server for the following messages, which may or may not come in sequential order:

- `JOIN` *playerid*`\n`: this message is sent if an opponent with ID *playerid* has joined you in the game. You can prepare to setup the other playfield for your opponent;

[1] https://engineering.purdue.edu/OOSD/F2009/Assignments/GPA/stage3.html

– `QUIT` *playerid*`\n`: this message is sent if an opponent has quitted the current game. This message may also be sent *during* the game.
– `READY?\n`: this message asks if the client is ready to start the game. The expected response is a string "`READY\n`" to the server. The game may not may not start immediately after the response, depending on whether you are the first or the second player in the game. The first player has to wait until a second player connects to the server, joins the game and responds with the ready message.

Table 2. Summary of server messages and expected reactions

Server Message	What to do
`READY?\n`	Respond `READY\n`.
`JOIN` *playerid*	Prepare opponent's playfield.
`QUIT` *playerid*	Get prepared that the opponent quitted.

3.3 Starting game

When game is about to start, the server will send the first pieces for both you and your opponent in the format "*playerid*:`PIECE` *piece*`\n`". Here *playerid* is used to distinguish the recipient of the incoming piece (because the pieces for the two players may be different), and *piece* is a string consisting of '0's and '1's which encodes a Tetris piece using the rule given in Stage 1. The piece generated by the server is always aligned to the top-left most position.

A "`START\n`" message tells the client to start the local game interaction. The second piece may come before or after this message but there is no guarantee on the order. So far your local client should be able to render a falling piece in the playfield, while holding at least one extra piece for displaying "Next Piece".

3.4 In game

During the game, your client may frequently receive messages from the server in an undeterministic manner (meaning that they may arrive at any time, in any order). All messages are in the form "*playerid* : *command*`\n`". *playerid* identifies the recipient of the message; *command* can be one of the following strings:

– `LEFT`: move the falling piece to the left by one block.
– `RIGHT`: move the falling piece to the right by one block.
– `ROTATE`: rotate the falling piece by 90°clockwise (aligned to top-left most position).
– `FALL`: move the falling piece down by one block. If this piece cannot move down any more, fix the piece in the playfield and the next piece appear at the top of the playfield.
– `ATTACK` *line*: add a line to the bottom of the playfield. *line* is a string of "0"s and "1"s, whose length is exactly 20. A "1" means an occupied block and "0" means an empty block (from left to right). If multiple `ATTACK` commands arrive, process them sequentially (i.e. the last line received will be added to the very bottom).
– `PIECE` *piece*: add a piece to the "Next Piece" queue. *piece* is a string which encodes a Tetris piece. All incoming pieces should be stored in a queue which follows the FIFO (first in first out) discipline.

– `GAMEOVER`: declares gameover. Note that this command only stops the game of the specific player, **not** the whole game.

When the local player presses a key to trigger a game action (left, rotate, etc.), the client informs the server by sending a message "*playerid*`:`*command*`\n`". *playerid* is the player ID of the local player, and *command* can be `LEFT`, `RIGHT`, `ROTATE` or `FALL`, corresponding to the triggered action.

Client should **not** process the actions triggered inside the playfield until it receives the returned message from the server. If the message is not returned, the actions can be considered rejected at server side.

3.5 Ending game

When all players have declared gameover, conclusive messages will be sent to all clients. These messages are in the form "*playerid*`:WIN\n`" or "*playerid*`:LOSE\n`". Each client will receive one such message for every player. The client should then display the game result for the local player.

To start a new game, the client should disconnect from the server and start the protocol over.

4 Supplementary Material

4.1 Server program

Please check the following URL for downloading the server program. You may use the server to test your client program.

http://code.google.com/p/purdue-wl-ece462/downloads/list

Should you find any bug or issue, please post it in the following URL with detailed description.

http://code.google.com/p/purdue-wl-ece462/issues/list

For each valid bug or issue report (the first to report), you will get 0.5 bonus point added into your participation bonus.

4.2 Questions and suggestions

Please post your questions and suggestions in "Q&A on GPA Specification" section on Blackboard discussion board.