

CS 246 Group Term Project - Tetris

Michelle Neuburger, Kriti Shrestha, Yiran Zhang

May 1, 2014

1 Introduction

For our term project, we are going to code the game of tetris.

Game Description: Tetris is a popular block game. The main components of the game are variously shaped tetrominos, which the player must stack on top of each other so as to create as few gaps as possible between the blocks. Once a row is completely filled with blocks, it is cleared away so as to make space for new tetrominos. The tetrominos are placed in a location specified by the player. The objective is to keep the game going for as long as possible without reaching the top-most row. In other words, the player must try to clear as many rows as possible. Points are awarded for rows cleared.

2 Tentative functions

a. `void createGameboard(int rows, int columns, char grid[][columns];`

createGameBoard: The game board will be represented by a rectangular 2D matrix displayed as a box in the `displayGameBoard` function. The matrix is a 2D array and the size of the array will be defined as a macro. The game board displays possible locations for a tetromino across the bottom. These locations will be named by numbers 0,1,2,... and so on and these column numbers will be elements of the bottom-most row of the matrix. Each row will also be indexed by the numbers 0,1,2,... on the left border of the box. The matrix will be updated when the user chooses to place a certain tetromino on the board. This is done in the

updateGameBoard function. The next tetromino to appear in the game will be created in the createTetromino function and will also be displayed alongside the game board by the displayTetromino function.

- b. **void displayGameboard(int rows, int columns, char grid[][columns], int bottom[TESTCOLS]);**

displayGameBoard: The function takes the game board array and the size of the board as arguments and returns void. It prints the game board to stdout. The function loops through the rows, printing every character in the board array, either blank space, or @. At the end of each row a newline is inserted. Below the printed matrix are numbered labels to identify each column.

- c. **void createTetromino();**

createTetromino: Each tetromino is a 2D array with 4 rows and 4 columns. There are 7 different shapes - I,O,T,J,L,S, and Z. Depending on the shape of the tetromino, the elements of the array are either blank spaces, or the character @. We also pre-determine the one special center character #, which is used to track the position dn display the tetromino. Each tetromino array has exactly 4 non-empty elements. There is a central element in each tetromino. The tetromino is printed to the console by the function displayTetromino. When the player chooses to place a tetromino on the board, the specified locations on the board are updated based on the elements of this array. Each tetromino is assigned a number between 0-6 which helps identify the tetromino shape.

- d. **void randomselectionOfTetromino(char tetro[][4]);**

randomSelectionOfTetromino: There are a total of 7 different shapes of tetrominos in the game - I,O,T,J,L,S,Z. Each of them will be each assigned a number from 0 to 6. Then, the function will generate a random number between 0 to 6 . The function will return the generated number. This number is argument that is taken in the displayTetromino function.

- e. **void rotateTetromino(char tetro[][4]);**

rotateTetromino: This function takes the tetromino array and the central element of the array as argument and accepts valid input from the

player to determine the new shape of the tetromino. It returns an updated tetromino. It will use matrix rotation to update the tetromino. Each tetromino is rotated clockwise, anticlockwise, vertically, or horizontally about its central element that acts as pivot. Rotation can be implemented by using nested loops that modify desired elements of the 2D array.

- f. **void displayTetromino(char t[][4]);**

displayTetromino: This function will take the response from player of several rotations and display the tetromino. This is also related with addTetro;

- g. **int specifyLocation();**

specifyLocationOnBoard: This function takes no argument. The function prompts the user to input a number corresponding to a column of the game board. This column is the desired location for the next tetromino. The function checks for valid input and converts the character entered into a number, which it then returns. The position entered will be indicated location for the 'center' character # in each tetromino.

- h. **int addTetro(int rows, int cols, char board[][cols], char tetromino[][4], int desig);**

addtetromino: The initial board has only empty spaces. The elements of the board will be updated depending on the selection of the tetromino. Depending on the specific location the player chooses to place the tetromino, after each step, the columns and rows inside the matrix will be updated.

- i. **int checkCompleted(const int rows, const int cols, char board[][TCOLS]);**

checkCompletedRow: This function takes the game board array as an argument and returns the index of the row that is completely filled. It also returns true if any row is completed. Using a nested loop, it checks the game board array row-by-row and records any row in which there are no non-empty elements. The row index is then used by the function clearCompletedRow to empty out that particular row.

- j. **int clearCompleted(int rows, int cols, char board[][TCOLS], int compidx);**

clearCompletedRow: The function takes as input an integer representing the index of the row that has been detected to be completely full, in addition to the game board itself and the size of the board. If the row filled is not the top row, the function iterates from the end of the full row up until the end of the top row (from the end of each row to the beginning, from higher-index rows to lower-index rows) and sets each array entry to be the same as the one directly above it. Then the function iterates through the top row and sets each entry to be blank.

k. **void updateScore(int *score, int signal);**

updateScore: The score will be zero at the initial state. In each round, if the checkCompletedRow returns true, then the player will win 10 points in the score, i.e. the winscore +=10. The new updated score will be score = score + winscore.

l. **int invalidMove(int c)**

int invalidMove: This function will keep track of the current height of the position which player chooses to place the tetromino. This function is used to determine whether the position player chooses is valid to put and whether the current height has exceeded the maximum rows available in the gameboard.

m. **int youLose()**

youLose: The game will end in the situation when at least one column on the board is stacked to the top row with tetrominos. To check whether the board is filled, the function will keep track of the top row, i.e. the row macro. If any spot of the top row is occupied, then the game ends.

3 List of function prototypes:

- void createGameboard(int rows, int columns, char grid[][columns];
- void createGameboard(int rows, int columns, char grid[][columns];
- void createTetromino();
- void randomselectionOfTetromino(char tetro[][4]);

- void rotateTetromino(char tetro[][4]);
- void displayTetromino(char t[][4]);
- int addTetro(int rows, int cols, char board[][cols], char tetro[][4], int desig);
- int youLose();
- int invalidMove(int c);
- int clearCompleted(int rows, int cols, char board[][TCOLS], int compidx);
- int checkCompleted(const int rows, const int cols, char board[][TCOLS]);
- void updateScore(int numrow);
- void updateScore(int *score, int signal);

4 Game Board and Tetrominos

This is tentatively what the game board will look like:

10											
9											
8											
7											
6											
5											
4											
3											
2											
1											
0	1	2	3	4	5	6	7	8	9	10	

We will use the character @ as building blocks of the tetrominos. Here is what the tetrominos will look like: And we use the symbol # to indicate the

'center ' of each tetromino. Scale and spacing of objects will be adjusted.

```

@@@@  @@      @      @@  @      @      @@
      @@  @@@  @@  @@@  @@@  @@@  @@

```

5 Team member roles

At this time, each of us is responsible for the description and implementation of certain functions. Here is how we have divided the tasks:

- a. **Kriti** - createGameBoard, createTetromino, rotateTetromino, chechCompletedRow
- b. **Michelle** - displayGameBoard, specifyLocationOnBoard, imposeTimeLimit, clearCompletedRow
- c. **Yiran** - displayTetromino, randomSelectionOfTetromino, updateGameBoard, updateScore, endGame

6 Dependency Graph

We group the functions based on the the functionalities and their relationships and then divide these functions in several groups which will be separate files that contains the functions. The above functions are divided into five categories: gameboard, tetromino, score, termination, player.

We then create the dependency graph that contains the separate files. The dependency graph include gameboard.c , tetromino.c , score.c , termination.c, player.c main.c and their header files: gameboard.h, tetromino.h, score.h, termination.h, player.h.

