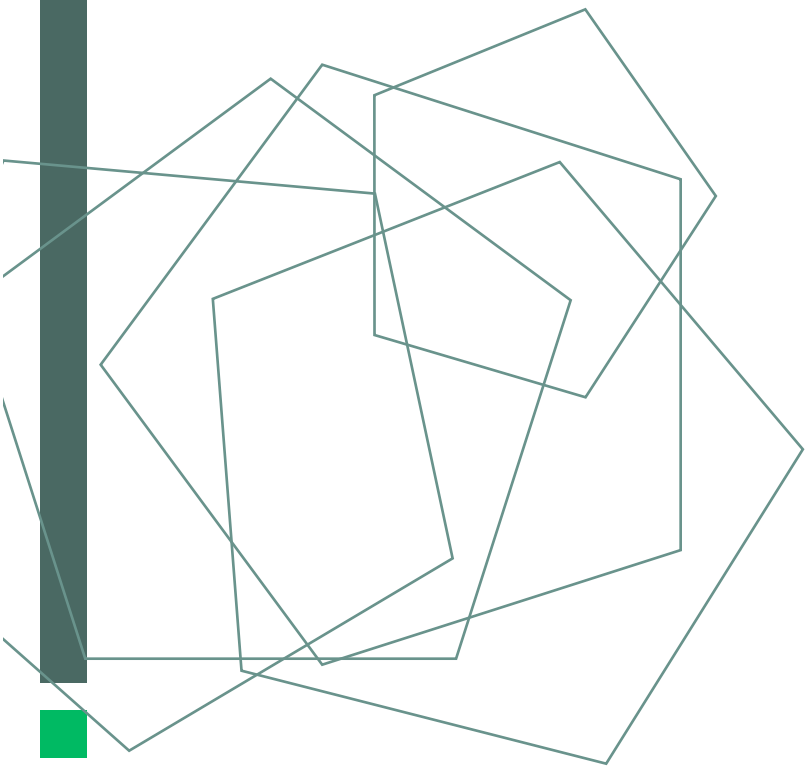
University of Waterloo

Spring 2014

CS 246 Assignment 5 Quadris



Designed By:

Lin Ai (20502890)

Meng Dong (20502751)

Contents

Introduction

This document outlines the implementation and structure of the Quadris project, designed by Lin Ai and Meng Dong, for Spring 2014 CS 246 Assignment 5 group project.

Differences From Design

Structure

- Merging left, right, down, rotate methods into class Block, which were originally in class MoveBlock and the class MoveBlock was removed
- Including all methods of Cell in the class Cell and the classes cellStat and Action were removed
- Adding class Game to support main.cc and control the progress of the whole game
- Removing the class Level and adding a method to control the blocks generated by different levels in the class Game
- Updating score directly when it is changed without a stand alone function to calculate and record.
- See the comparison of UML for more details

Feature

- Adding the skip feature to throw away the current block and skip to the next. The command can be executed at most twice on one block.

UML Diagram

- Removing moveBlock
- Removing Level
- Removing cellStat
- Removing Action
- Block
  - Removing blockCleared():Boolean
  - + getTracex(): int
  - + getTracey(): int
  - + getBlockx(x:int): int
  - + getBlocky(y:int): int
  - + getType(): char
  - + legalmove(command:string, t:int): bool
  - + left(): void
  - + right(): void
  - + down(): void
  - + rotateClock(): void
  - + rotateCounter(): void

    - + rotate(direct:string): void
    - + leftbottom(): void
- Board
    - Removing calScore():void and getScore():int
    - + init(r:int, c:int): void
    - + genBlock(type1:char, type2:char): void
    - + left(t:int): void
    - + right(t:int): void
    - + down(t:int): void
    - + rotate(t:int, direct:string): void
    - + drop(type:char): void
    - + gen(): void
    - + checkfull(r:int): bool
    - + clearRow(): void
    - + drawBoard(): void
    - + gameover(): void
    - + skip(type:char): void
    - + friend operator«(out:ostream&, b:Board&): ostream&
- Cell
    - Removing getStatus():Boolean
    - +setType(c:char):void
    - + getType():char
    - + undrawCell():void
    - + friend operator«(out:ostream&, c:Cell&): ostream&
- Adding Game
    - + start(): void
    - + end(): void
    - + srtLevel(n:int): void
    - + textonly(): void
    - + setfile(s:string): void
    - + restart(): void
    - + reafile(): string
    - + genlevelblock(level:int): char

Quadris UML Diagram

Block

+ drawIBlock(): void
+ drawJBlock(): void
+ drawLBlock(): void
+ drawOBlock(): void
+ drawSBlock(): void
+ drawZBlock(): void
+ drawTBlock(): void
+ genBlock(): void
+ getTracex(): int
+ getTracey(): int
+ getBlockx(x:int): int
+ getBlocky(y:int): int
+ getType(): char
+ legalmove(command:string, t:int): bool
+ left(): void
+ right(): void
+ down(): void
+ rotateClock(): void
+ rotateCounter(): void
+ rotate(direct:string): void
+ leftbottom(): void

Board

+ init(r:int, c:int): void
+ genBlock(type1:char, type2:char): void
+ left(t:int): void
+ right(t:int): void
+ down(t:int): void
+ rotate(t:int, direct:string): void
+ drop(type:char): void
+ gen(): void
+ checkfull(r:int): bool
+ clearRow(): void
+ drawBoard(): void
+ gameover(): void
+ skip(type:char): void
+ friend operator«(out:ostream&, b:Board&): ostream&

Game

+ start(): void
+ end(): void
+ srtLevel(n:int): void
+ textonly(): void
+ setfile(s:string): void
+ restart(): void
+ reafile(): string
+ genlevelblock(level:int): char

Cell

+ getX(): int
+ getY(): int
+ getLevel(): int
+ getType(): char
+ setType(c:char): void
+ drawCell(): void
+ undrawCell(): void
+ friend operator«(out:ostream&, c:Cell&): ostream&

Xwindow

+ fillRectangle(int,int,int,int,int): void
+ drawString(int,int,string,int): void

Implementation Strategy

Block

In the Block class, two integer arrays, blockx[4] and blocky[4], are set up to trace the four coordinates of each tetromino. Cells in a board where a block exists can be addressed by using these coordinates. By calling the draw functions in the Block class, the seven different shapes of tetrominoes can be drawn. To indicate a cell contains a block, the cell status is set true. For example, if a Z block is to be drawn, we simply set the four cells' status, who form a shape 'Z', to be true, as the graph shows:

F	F	F
T	T	F
F	T	T

The functions for motion (for example, left, right, rotate) of a block are added into the Block class as well. The four coordinates of a block are traced in each of the motion functions.

Board

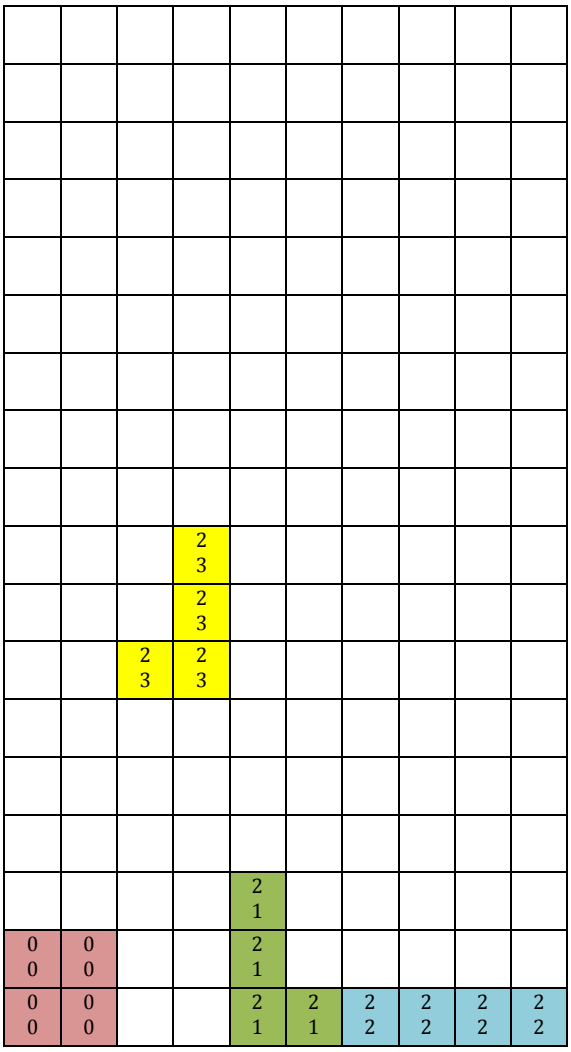
A board is a two-dimensional array of cells, with a size of 18*10. Each cell stores its own information, including status, type, and coordinates. Each cell also stores its own blockindex, which indicates which block it belongs to. In the Board class, two integer arrays, blocklevel and cellexist, are set up to store the level and number of cells remains in each of the blocks. The blocks are numbered by the index of the arrays, which is corresponding to the blockindex of each cell. When a block is dropped, a blockindex is assigned to the four cells where the block is dropped. The graph on the next page shows the assignment of blockindex. The upper number indicates the level in which the block is generated, and the bottom number indicates the blockindex of a block.

When a block is completely cleared, for example, the 'I' block in the left board below, all cells that have greater blockindex than that of the cleared block will be reassigned a blockindex that is one less than their previous one. For example, the remaining cell of the 'L' block in the right board below will then have the blockindex 2. This is to make sure that the maximum blockindex is within the maximum number of blocks that could have on the board, which is 150 blocks in total (since 3 extra rows are reserved, the actual board is 15 * 10).

The level and number of cells remains in each block are used to calculate the total score. When a row is filled and cleared, the score is calculated. If rows are cleared, the score is calculated by the following equation:
*score = score + (level + number of rows cleared) * (level + number of rows cleared)*.
For each block cleared, the score is calculated by the following equation:

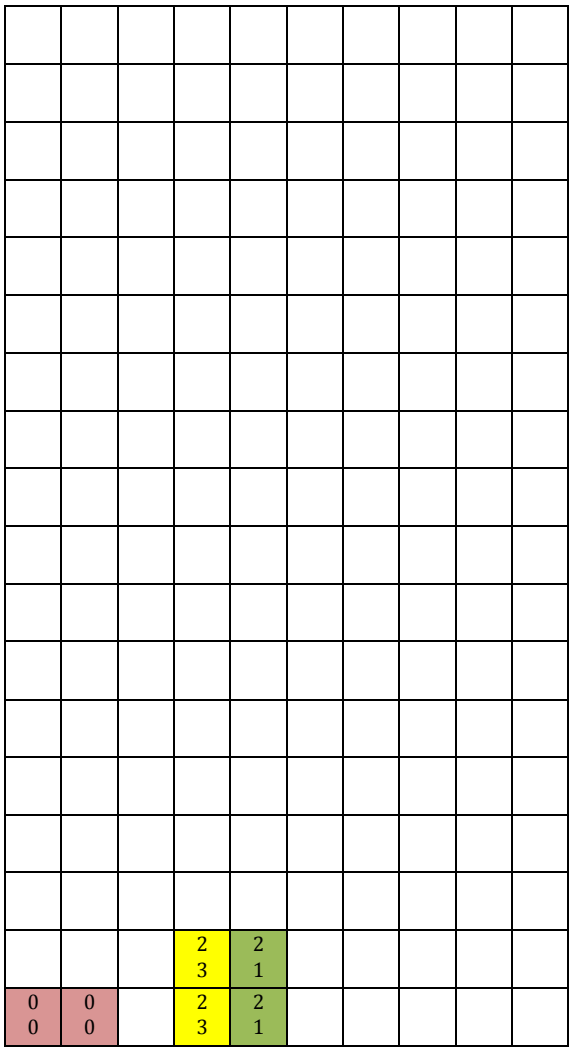
*score = score + (level + 1) * (level + 1)*, where level is the level in which the block is generated.

Before dropping:



blocklevel[3] = 2, cellexist[3] = 4

After dropping:



blocklevel[3] = 2, cellexist[3] = 2

Rotate

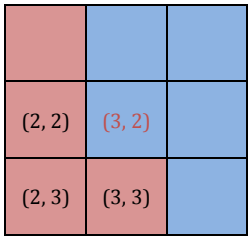
In order to update the coordinates of each cell in a block after rotation, the coordinate of the central cell in a three by three grid, which contains the block, is traced. All shapes of tetrominoes, expect the 'I' block, can be contained in a three by three grid. There are two steps to obtain the new coordinates after the rotation.

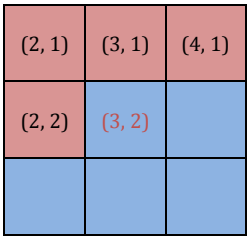
Step 1:

Let the central cell have the coordinate (x0, y0). When a block is rotated clockwise, the new coordinates of each cell are as follows:
(-(y – y0) + x0, (x – x0) + y0).

When a block is rotated counterclockwise, the new coordinates of each cell are as follows:
((y – y0) + x0, -(x – x0) + y0).
The following graph shows the first step of a clockwise rotation of a 'T' block.

Before rotation: (-(y – y0) + x0, (x – x0) + y0) After rotation:



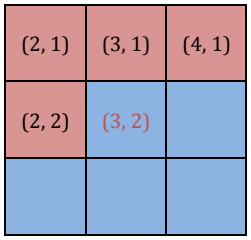


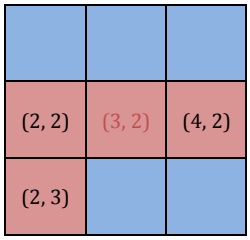
'I' block can be fit into a four by four grid. However, when a 'I' block is rotated, no matter clockwise or counterclockwise, it can only be either vertical and horizontal, so the new coordinates can be easily obtained.

Step 2:

When a three by three grid is rotated, the block is then to be moved to the left bottom of the grid.

Before: After:





Next Block

By adding a private field, Block *next, in the Board class, the type of the next block can be traced. After dropping the current block, the next block is generated, with the type being traced. The pointer then points to the next block that will be generated.

Level

The process of generating blocks in different levels is realized by adding a function genlevelblock(int level) in the class Game. To control the probability of generating different blocks, the remainders of random numbers divided by 6, 8, 11 are assigned to different blocks. When field level is not 0, the program will call genlevelblock to get appropriate blocks

Graph

Under graphic mode, first set the background color to black and text color to white then draw the needed text. A method called drawboard() is created in the class Board. Turn the integer score, level and hi score into strings and draw it on the board every time after a block is dropped. In the drawboard() function, fill the cell if the status is true and unfill it if the status is false by drawcell() and undrawcell() methods. The coordinates of the cells are (x * 25 + x + 1), (y * 25 + 93 + y).

Command Interpreter

To simplify the input command so that only as much of a command as is necessary to distinguish it from other commands is entered, substring of an input command is read. For example, to move a block one step to the right, the player can simply enter "right" or "ri", the program will take in the entered command, and compare its substring from the first letter to the second, and see if it equals "ri". Any command with "ri" as the first two letters will be interpreted as a "right" command. Similarly, any command with "lef" as the first three letters will be interpreted as a "left" command.

To allow multiplier prefix, input command is separated into two parts: the multiplier and the actual command.

Additional Feature: Skip

An additional feature is added to this program to make it more entertaining. The feature is called skip, which allows the player to skip at most two blocks that they do not want consecutively. By entering the command "skip" (or any substring starting from "s"), player can skip the block they do not want, and have the next block immediately. To achieve this, a function skip is added into the Board class. By calling this function, the current block is thrown right away, and the next block turns into the current block.

Final Questions

1. What lessons did this project teach you about developing software in teams? If you worked alone, what lessons did you learn about writing large programs?

We worked as a team on this project. During the first several days, we found it hard to reach an agreement on how to structure the whole program. We wavered between separating the seven shapes of tetrominoes into seven different classes and combining them into one class. We also had different opinions on how to build the block. One suggested building an array of pointer to cells, which could allow us to see each block as a whole, whereas the other member suggested storing only the coordinates of the four cells in a block. After careful discussion, we soon achieved our agreement and started implementing.

During our implementing, we often felt that teamwork could be more efficient than working alone. By splitting the workload, we spent less time in total

finishing the whole project. We also found that both members had their own strengths in different areas, which made it much easier for us to cooperate, and at the same time, enabled us to simplify our code and strengthen the functionality of the program much easier. Both members were very innovative, and could both come up with a lot of smart ideas that the other could not think of.

This project allowed us to realize how important communication is when developing software in team. When different opinions come across, the most effective way is to discuss them together as a team. If new ideas come up, sharing with partner make it easier to put the ideas into practice. Better communication makes better programs. What made us in advantage was that both members are roommates, which allowed us to communicate with each other often and easily. This also helped us finish the project ahead of time.

2. What would you have done differently if you had the chance to start over?

If we had the chance to start over, we would put more thoughts into the UML diagram design before we start implementing. Our final UML diagram turns out having many differences than the original one. A badly designed UML diagram may mislead the whole implementation, which could waste the programmers a lot of time doing useless work.

Conclusion

Overall, we felt that this project was an excellent experience. This project gave us an idea how software development in team is, and helped us better understand the importance of communication and time management in software development. We are looking forward to more experience like this.