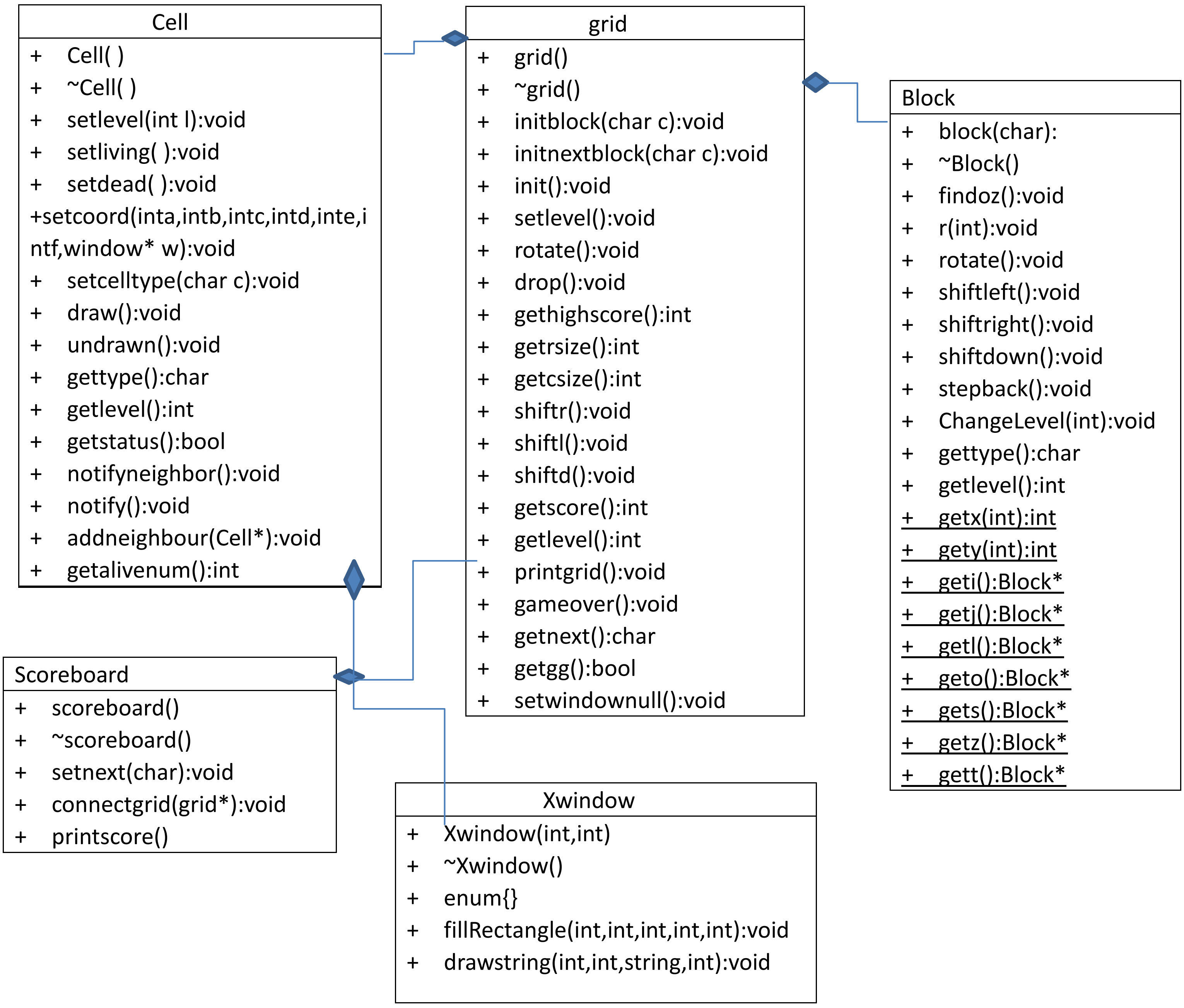

Design:

Used observer pattern in cell class

Used singleton in block class

Basically the program is consisted of 5 Classes: cell, grid, block, scoreboard and window

Grid:

Grid class is the main part of the program. It will create the grid and fill it with the cell. There are two pointers of Block in the grid in order to handle the current block, next block and moving of the block (shift and rotate)

Cell:

Cell class is connected to the window class. When turn on somewhere in the grid is actually setting the cell to alive. When cell is set to alive it will notify the window class to draw it on the graphical display. Oppositely, when a cell is set to dead, it will notify the window class to undrawn it on the graphical display. Each cell has 3 neighbors since every block has 4 cells in it. When a cell is set to dead it will notify its neighbors (observer pattern). When its number of living neighbors becomes 0 and the cell is in the line which needs to be cleared, it's time to add the score of clearing a complete block.

Scoreboard:
The scoreboard part stores the high score, and control the display in text mode.
Block:
The block part control the block from create, to doing command include rotate, shift left, shift right, shift down, change level, and allow grid to go one step back when it is out of grid.
We use singleton for every block we create to ensure that we do not initial other blocks expect the 7.

Difference:

Cell:

1 change the function secoord(int a,int b,int c,int d,in e,int f, window* w)
Reason:

Add more parameters to the setcoord function. In order to handle the window display we need add height,width,x-coordinate and y – coordinate to cell class. So we need to rewrite the setcoord function.

Grid:

1 turnon(int,int), turnoff(int, int), cleargrid(), clear(), clearline(int) move from public to private
Reason:

during the programming, we found that those function above is only used by grid class itself. So it's better to put it in the private field.

2 add initnextblock(char), gethighscore(), getrsize(), getcsize(), gameover(), sethighscore(int), getnext(), getgg(), setwindownull()
Reason:
initnextblock(char) getnext() :

those two function are need. Forgot to add in the UML before due date 1

gethighscore() sethighscore() :

first two functions are used for scoreboard class to track the h ighscore and rewrite the highscore in grid class

getgg():

this function is used to check whether the game is over. Since gg is a private field in grid class so we need write a function in order to get it outside of the class

setwindownull() :

this function is used when using –text option.

Scoreboard:

1 delete addscore(int) changelevel(int) gethighscore() getscore()

Reason:

We originally think that score board should control all the features that related to score. During the programming process, we believe that put some parts in the grid can make it easier to control.

2 add printscore()

Reason:

We decided to save a pointer from grid to scoreboard to reduce the possible circle include problem and other possible problem. So we decide to move the control of print from grid to scoreboard.

Block:

1 add ~Block()

Reason:

We did not consider that we should delete a singleton pointer. This is used by the function in the atexit (cleaner) to perform delete the space for singleton pointers.

Window:

No change

PRNG:

Delete class PRNG

Approach of requirements:

1 blocks: done

2 board: done

3 display: done

4 next block: done

5 command interpreter: done

6 scoring: done

7 command-line interface: done

Answers to questions

Block:

How could you design your system to make sure that only these seven kinds of blocks can be generated, and in particular, that non-standard configurations cannot be produced?

Answer:

Use singleton method to make sure that only these 7 kinds of blocks are generated.

How could you design your system to allow some game levels to occasionally generate transparent blocks, or skippable blocks? Could a block be both transparent and skippable?

Answer:

We can add a field in block class called “btype”, which is an integer that randomly picking up from 0 to 19. 0 means both transparent and skippable block. 1 means skippable block. 2 means transparent block. Otherwise, the block is normal.

Next block:

How could you design your program to accommodate the possibility of introducing additional levels into the system, with minimum recompilation?

Answer:

We can use visitor pattern to reduce the minimum change. The general action can be specialized by visitor, and when we add other level, we can improve different actions by different visitors.

Command interpreter:

How could you design your system to accommodate the addition of new command names, or change to existing command names, with minimal changes to source and minimal recompilation? How difficult would it be to adapt your system to support a command whereby a user could rename existing commands?

Answer:

Using the decorator pattern can mimics original commands behaviors which could handle different command that perform the same things.

What lessons did this project teach you about developing software in teams?

Answer:

1 Read the documents carefully. Understanding what to do is important, since it is pretty hard to correct one part of the program when the whole program is done.

2 communications is important in the team work, because files depend on each other and files are written by different person.

3 Good division of work can save time and motivate the team members.

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

Answer:

1 Start early so that we can have enough time to correct error in this project, because sometimes the project will behave differently from what we expect.

2 design a test suite for the program.

3 use valgrind check the memory leak