C++ 프로그래밍 설계 과제 : 테트리스 만들기 3단계

20132926 전호현

요 약

본 과제에서는 상속과 다형성을 이용한 객체지향 설계를 통해 총 7개의 테트로미노 블록이 무작위로 떨어지는 테트리스 게임을 완성한다.

---

1. Tetris.cpp , Tetris.h

3단계의 테트리스 클래스는 2단계의 모든 기능에 더해서, 테트로미노를 랜덤으로 정하는 기능과, 현재 테트로미노와 다음에 나올 테트로미노에 대한 변수를 정의한다. 또한, 사용자로부터 이름을 입력받아 저장한다.

- 테트로미노를 랜덤으로 정하는 기능

  다음에 Insert할 테트로미노 블록을 랜덤으로 정하기 위해 rand()함수를 이용한다. rand()함수를 사용하기 위해 전처리기에 #include <cstdlib>을 포함해 준다. 그런데 이 rand()함수는 프로그램을 다시 수행할 때마다 같은 숫자들이 생성된다. 이를 해결하기 위해 rand함수의 시드를 설정해주는 srand()함수를 사용한다. 하지만 시드가 같을 경우 매번 같은 수를 같은 순서로 생성하기 때문에, 시드값을 계속해서 바꿔주기 위해 시간값을 사용한다. 여기서 시간값을 이용하기 위해서 전처리기에 #include <ctime>을 포함하고, srand(time(NULL)) 값을 넣어주면 시드값이 시간에 따라 계속 변화하기 때문에 완벽한 난수 처리가 이루어진다.

  이 난수 처리를 이용하여 nextBlock 값을 rand()%7+1 로 주어서, nextBlock의 숫자 범위가 1~7 사이에 생성되도록 한다. 그리고 현재 블록이 무엇인지를 나타내 줄 curBlock에 nextBlock값을 넣어주고, 새로운 블록을 생성하는 반복문이 수행될 때마다 nextBlock값과 curBlock값이 변화하도록 구현하였다. NextPane에 nextBlock을 나타내 줄 수 있도록 nextBlock값을 board b로 넘겨준다.

- 사용자로부터 이름을 입력받아 저장하는 기능

  게임 시작 전에 사용자로부터 이름을 입력받아 출력가능하도록, name변수를 선언하고 이 변수를 board의 setName 함수를 이용하여 board의 name에 복사해 저장한다.

```
void Tetris::play(){
        b.setName(name);
        srand(time(NULL));
        int input;
        bool gameOver = false;
        bool newBlock = false;
        int nextBlock = rand()%7 +1; // 다음 블록을 나타내는 변수
        int curBlock; // 현재 블록을 나타내는 변수
        updateScreen();
        input = getch();


        while(1){ // 새로운 블록을 생성하는 반복문
                curBlock = nextBlock;
                nextBlock = rand()%7+1; // 새로운 블록 랜덤 생성
                b.setNextBlock(nextBlock); // 다음 블록 넘겨줌
                b.eraseBlock(); // 일단 꽉찬블록 다 지워주고
                gameOver = !b.insertBlock(curBlock);
                updateScreen();
                if(gameOver){
                        input = getch();
                        break;
                }
```

2. Board.h, Board.cpp

* 파란색 함수 또는 클래스는 본 파일의 뒷부분에서 정의한다

2단계의 Board클래스는 OBlock의 움직임만을 Board에 그대로 구현하였었다. 3단계에서는 모든 테트로미노(블록)의 움직임을 구현하기 위해 Block.h 파일을 포함한다. 그리고 각 블록별로 숫자를 세고 다른 움직임을 보일 수 있도록 변수를 선언한다.

먼저, Board.h는 다음과 같다.

```
#ifndef _BOARD_H_
#define _BOARD_H_

#include "Block.h"
class Board{
	friend class StatPane;
	friend class InfoPane;
	friend class BoardPane;
	friend class NextPane;
	friend class Block;

private:
	int height;
	int width;
	Cell **cells; // 블록을 그릴 2차원배열
	int t_x, t_y;
	int nextblock; // 다음에 나올 블록을 저장할 변수
	Block *iblock, *oblock, *lblock, *jblock, *sblock, *zblock, *tblock;
                                    // 블록에 따라 다른 움직임(다형성)을 구현하기 위한 주소값 변수
	int fulllines;
	int Iblockcount, Oblockcount, Lblockcount, Jblockcount, Sblockcount, Zblockcount, Tblockcount;
	// 서로 다른 블록의 개수를 셀 변수
	int total; // 현재까지 insert된 블록의 총 개수를 셀 변수
	char name[100]; // 사용자의 이름을 저장할 변수
public:
	Board(int w, int h);
	~Board();
	bool insertBlock(int curblock); // 현재 블록을 받아 해당 블록을 insert
	bool moveBlockLeft(int curblock); // 현재 블록을 받아 해당 블록을 왼쪽으로 이동
	bool moveBlockRight(int curblock); // 현재 블록을 받아 해당 블록을 오른쪽으로 이동
	bool moveBlockDown(int curblock); // 현재 블록을 받아 해당 블록을 아래로 이동
	bool rotateBlock(int curblock); // 현재 블록을 받아 해당 블록을 반시계방향으로 회전
	void dropBlock(int curblock); // 현재 블록을 받아 해당 블록을 drop
	void eraseBlock();
	void setNextBlock(int nextblock); // 다음 블록 값을 받아 저장
	void setName(char *string); // 사용자로부터 입력받은 이름을 받아 저장
};

#endif
```

2단계에 비해 크게 변화된 Board클래스의 함수들만 설명하면,

- bool Board::insertBlock(int curblock);

아래와 같이 구현하여 curblock에 따라 다른 insertBlock을 호출 할 수 있도록 한다. 다른 move함수들과 rotateBlock도 이와 같이 다형성을 활용하여 구현한다.

```
total++;
if(curblock == 1){
        Iblockcount++;
        return ((IBlock *)iblock)->insertBlock(t_x,t_y,cells);}
if(curblock == 2){
        Oblockcount++;
        return ((OBlock *)oblock)->insertBlock(t_x,t_y,cells);}
if(curblock == 3){
        Lblockcount++;
        return ((LBlock *)lblock)->insertBlock(t_x,t_y,cells);}
if(curblock == 4){
        Jblockcount++;
        return ((JBlock *)jblock)->insertBlock(t_x,t_y,cells);}
if(curblock == 5){
        Sblockcount++;
        return ((SBlock *)sblock)->insertBlock(t_x,t_y,cells);}
if(curblock == 6){
        Zblockcount++;
        return ((ZBlock *)zblock)->insertBlock(t_x,t_y,cells);}
if(curblock == 7){
        Tblockcount++;
        return ((TBlock *)tblock)->insertBlock(t_x,t_y,cells);}
```

3. Block.h, Block.cpp

Block.h에서는 사용할 모든 블록들이 공통적으로 가지는 변수와 함수들을 묶어서 Block 클래스로 만든다. Block클래스는 함수의 헤더만 존재하고 몸체는 없는 순수 가상함수를 포함하는 추상 클래스의 형태로 구현한다. 자세한 구현은 아래와 같다.

```
#ifndef _BLOCK_H_
#define _BLOCK_H_

enum Cell{VOID, I, O, L, J, S, Z, T}; // 블록의 형태를 정의

class Block{

protected:
        int rotate; // 회전 횟수를 세는 변수
        int height; // Cell을 나타낼 배열의 높이를 나타내는 변수
        int width; // Cell을 나타낼 배열의 너비를 나타내는 변수

public:
        Block();
        // 아래의 함수들은 Block을 상속하는 클래스들에서 구현할 수 있도록 순수 가상함수로 구현한다
        virtual bool insertBlock(int &x, int &y, Cell **cells) = 0;
        virtual bool moveBlockLeft(int &x, int &y, Cell **cells) = 0;
        virtual bool moveBlockRight(int &x, int &y, Cell **cells) = 0;
        virtual bool moveBlockDown(int &x, int &y, Cell **cells)= 0;
        virtual bool rotateBlock(int &x, int &y, Cell **cells)= 0;
        virtual void dropBlock(int &x, int &y, Cell **cells)=0;
};
```

4. I, O, L, J, S, Z, T Block.cpp

각각의 블록 클래스에서는 board 클래스로부터 현재 x값과 y값, cell의 주소를 받아서 블록 별로 서로 다른 형태로 구현될 수 있도록 좌표값을 찍는다. 또한, 같은 블록 내에서도 회전 횟수에 따라 서로 다른 움직임을 보여야 하므로, 회전 횟수 별로 서로 다른 좌표값을 찍어서 구현한다. IBlock의 예시를 보면 다음과 같이 구현된다.

```
#include "Block.h"
#include "Board.h"

IBlock::IBlock() : Block(){}

bool IBlock::insertBlock(int &t_x, int &t_y, Cell **cells){

	t_y = 4; t_x = 4; // 현재 맨 아래
	rotate = 0; // 최초 회전 횟수를 0으로 초기화

	if(cells[t_y][t_x] == VOID){
		cells[t_y][t_x] = I; // 한칸 위
		cells[t_y-1][t_x] = I; // 두칸 위
		cells[t_y-2][t_x] = I; // 세칸 위
		cells[t_y-3][t_x] = I; // 네칸 위
		return true;
	}
	else return false;
}

bool IBlock::moveBlockLeft(int &t_x, int &t_y, Cell **cells){

	if(rotate == 1 || rotate == 3){ // 회전 후 가로형태일경우
	// 회전 후 최초좌표는 블록의 제일 왼쪽
		if(cells[t_y][t_x-1] == VOID){
			if(t_x>0){
				cells[t_y][t_x-1] = I;
				cells[t_y][t_x+3] = VOID;
				t_x = t_x-1;
				return true;
			}
			else return true;
		}
	else return true;
	}

	else if(rotate == 0 || rotate == 2){ // 기본 세로형태일 경우
	// 현재 위치는 맨 아래
	if(cells[t_y][t_x-1] == VOID && cells[t_y-1][t_x-1] == VOID){
		if(cells[t_y-2][t_x-1] == VOID && cells[t_y-3][t_x-1] == VOID){
			if(t_x>0){
				cells[t_y][t_x] = VOID; // 원래 있던값들 지워준다
				cells[t_y-1][t_x] = VOID;
				cells[t_y-2][t_x] = VOID;
				cells[t_y-3][t_x] = VOID;
				cells[t_y][t_x-1] = I; // 왼쪽으로 이동
				cells[t_y-1][t_x-1] = I;
				cells[t_y-2][t_x-1] = I;
				cells[t_y-3][t_x-1] = I;
				t_x = t_x-1;
```

```
                        return true;
                        }
                else return true;
                }
                else return true;
                }
                else return true;
        }
}

bool IBlock::moveBlockRight(int &t_x, int &t_y, Cell **cells){
        if(rotate == 1 || rotate == 3){
                ......
        }
        else if(rotate == 0 || rotate == 2){
                .....
        }

bool IBlock::moveBlockDown(int &t_x, int &t_y, Cell **cells){
        if(rotate == 1 || rotate == 3){
                        ....
        }
        if(rotate == 0 || rotate == 2){
                        ....
        }

void IBlock::dropBlock(int &t_x, int &t_y, Cell **cells){
        if(rotate == 1 || rotate == 3){
                ......
        }
        else if(rotate == 0 || rotate == 2){
                        ....
        }

bool IBlock::rotateBlock(int &t_x, int &t_y, Cell **cells){
        rotate = (++rotate) % 4; // 회전 횟수를 증가시키고 0~3 사이의 값이 나올수 있도록 한다
        // 현재 I블록의 좌표는 맨 아래
        if(rotate == 1 || rotate == 3){
                .....
        }
        else{
                ......
        }
```

위에서 예로 본 IBlock의 경우는, 1회 회전과 3회회전의 모양이 같고, 2회 회전을 했을 때 처음의 모양과 같다. 그래서 두 가지 경우로만 나누어 좌표를 찍었다. 하지만 L, J, T 블록은 회전을 하는 모든 경우의 모양이 다르기 때문에 4가지 경우를 모두 나누어 좌표값을 정해주어야 한다. S,Z 블록은 I 블록과 마찬가지이고, OBlock은 회전을 해도 모양이 변화하지 않기 때문에 따로 다른 좌표값을 찍을 필요가 없다.

5. NextPane draw(Board &b)

2단계 테트리스까지는 NextPane에 아무런 값도 들어있지 않았었다. 3단계에서는 다음에 나올 테트로미노(블록)을 board로부터 받아와 NextPane에 나타내어 플레이어가 미리 어떤 블록이 나올지 미리 알 수 있도록 한다.

```
void NextPane::draw(Board &b){
        init_pair(4, COLOR_CYAN, COLOR_BLACK);
        wattron(win_, COLOR_PAIR(4));
        box(win_,0,0);
        mvwprintw(win_, 0,width_/2-5, "N E X T");
        wattroff(win_, COLOR_PAIR(4));
        // 기존에 그려진 값들을 지워주기 위해 배경색과 똑같은 색으로 덮는다.
        wattron(win_, COLOR_PAIR(13));
        mvwhline(win_, 2, 1, ACS_CKBOARD, 20);
        mvwhline(win_, 3, 1, ACS_CKBOARD, 20);
        wattroff(win_, COLOR_PAIR(13));

        // 아래에서는 nextblock에 해당하는 블록의 그림을 그린다
        if(b.nextblock == 1){
                wattron(win_, COLOR_PAIR(11)); // 파란색 켠다
                mvwhline(win_, 2, 8, ACS_CKBOARD, 8); // 테트로미노
                wattroff(win_, COLOR_PAIR(11)); // 파란색 끈다}
        if(b.nextblock == 2){
                wattron(win_, COLOR_PAIR(6)); // 빨간색 켠다
                mvwhline(win_, 2, 10, ACS_CKBOARD, 4);
                mvwhline(win_, 3, 10, ACS_CKBOARD, 4);
                wattroff(win_, COLOR_PAIR(6)); // 빨간색 끈다}
        if(b.nextblock == 3){
                wattron(win_, COLOR_PAIR(7)); // 보라색 켠다
                mvwhline(win_, 2, 9, ACS_CKBOARD, 6); // 테트로미노 그린다
                mvwhline(win_, 3, 9, ACS_CKBOARD, 2); // 테트로미노
                wattroff(win_, COLOR_PAIR(7)); // 보라색 끈다}
        if(b.nextblock == 4){
                wattron(win_, COLOR_PAIR(12)); // 흰색 켠다
                mvwhline(win_, 2, 9, ACS_CKBOARD, 6);
                mvwhline(win_, 3, 13, ACS_CKBOARD, 2);
                wattroff(win_, COLOR_PAIR(12)); // 흰색 끈다}
        if(b.nextblock == 5){
                wattron(win_, COLOR_PAIR(9)); // 초록색 켠다
                mvwhline(win_, 2, 9, ACS_CKBOARD, 4);
                mvwhline(win_, 3, 7, ACS_CKBOARD, 4);
                wattroff(win_, COLOR_PAIR(9)); // 초록색 끈다}
        if(b.nextblock == 6){
                wattron(win_, COLOR_PAIR(10)); // 시안색 켠다
                mvwhline(win_, 2, 9, ACS_CKBOARD, 4);
                mvwhline(win_, 3, 11, ACS_CKBOARD, 4);
                wattroff(win_, COLOR_PAIR(10)); // 시안색 끈다}
        if(b.nextblock == 7){
                wattron(win_, COLOR_PAIR(8)); // 노란색 켠다
                mvwhline(win_, 2, 9, ACS_CKBOARD, 6); // 테트로미노
                mvwhline(win_, 3, 11, ACS_CKBOARD, 2); // 테트로미노
                wattroff(win_, COLOR_PAIR(8)); // 노란색 끈다}
        wrefresh(win_);
}
```

6. 키보드로부터 사용자 이름 입력받기

입력 파일 없이 프로그램을 구동하는 경우, 게임 시작 전에 사용자의 이름을 입력 받을 수 있도록 구현한다. 사용자 이름을 입력하고 Enter를 누르면 게임이 시작된다. 그리고 사용자의 이름은 InfoPane에 출력된다. 사용자의 이름을 입력받기 위해 main.cpp를 아래와 같이 바꿔준다.

```
#include <ncurses.h>
#include <fstream>
#include <iostream>
#include "Tetris.h"
using namespace std;

int main(){
        char string[100];
        for(int i=0; i<100; i++) string[i] = '\0';
        ifstream instream;
        instream.open("input1.txt");
        if(instream.fail()){
                cout << "테트리스 게임을 시작합니다" << endl;
                cout << "사용자의 이름을 입력하세요" << endl;
                cout << "이름 : "; cin >> string;
                Tetris t;
                t.setname(string);
                t.play();
        }
        Tetris t;
        t.replay(instream);
        return 0;
}
```

input file이 없어서 instream에 실패했을 경우에, 사용자의 이름을 string에 저장하고 이 이름을 Tetris t의 setname으로 넘겨준다. 그리고 Tetris 클래스에서 board클래스로 이 name을 넘겨준 후에 이 name을 InfoPane에서 받아서 출력한다. 아래는 InfoPane의 구현이다.

```
#include <ncurses.h>
#include "Pane.h"
        InfoPane::InfoPane(int x, int y, int w, int h) : Pane(x,y,w,h){}
        void InfoPane::draw(Board &b){ // Pane클래스의 가상함수인 draw() 구현
                init_pair(1, COLOR_CYAN, COLOR_BLACK); // 색깔 정의. (색상짝 정의 숫자, 전경색, 후경색)
                init_pair(2, COLOR_YELLOW, COLOR_BLACK);

                wattron(win_, COLOR_PAIR(1));
                box(win_,0,0);
                mvwprintw(win_, 0, width_/2-5, "I N F O"); // win_에서의 커서를 0,0으로 옮기고 INFO PANE 출력
                wattroff(win_, COLOR_PAIR(1));
                mvwprintw(win_, 1, 1, "Your Name : ");
                mvwprintw(win_, 1, width_-10, b.name); // 받아온 이름값을 출력한다.
                mvwprintw(win_, 2, 1, "Full Lines : ");
                mvwprintw(win_, 2, width_-5, "%d", b.fulllines);
                mvwprintw(win_, 3, 1, "SCORE :");
                wattron(win_, COLOR_PAIR(2)); // 색깔 속성을 켠다
                mvwprintw(win_, 3, width_-5, "%d", b.fulllines*10);
                wattroff(win_, COLOR_PAIR(2)); // 색깔 속성을 끈다
                wrefresh(win_); // win_를 출력
        }
```

7. input file 받아오기

instream을 이용하여 input file로부터 값들을 읽어와 넣는다. 기존과 달라진 부분은 빨간색으로 표시하였다.

```
void Tetris::replay(ifstream &instream){
        instream >> name; // input file로부터 사용자의 이름을 받아온다
        b.setName(name); // 사용자의 이름을 board의 name으로 넘겨준다
        int seed; // seed 값을 받아오기 위한 변수 선언
        instream >> seed; // seed 값을 input.txt 로부터 불러온다
        srand(seed); // seed에 불러온 값을 넣는다
        char input;
        bool gameOver = false;
        bool newBlock = false;
        int nextBlock = rand()%7+1;
        int curBlock;
        updateScreen();
        input = getch();

        while(1){ // 새로운 블록을 생성하는 반복문
                curBlock = nextBlock;
                nextBlock = rand()%7+1; // 새로운 블록 랜덤 생성
                b.setNextBlock(nextBlock); // 다음 블록 넘겨줌
                b.eraseBlock(); // 일단 꽉찬블록 다 지워주고
                if(gameOver){
                        input = getch();
                        break;
                }
                nextBlock = rand()%7+1;
                gameOver = !b.insertBlock(curBlock);
                updateScreen();
                if(gameOver){
                        input = getch();
                        break;
                }
                while(!newBlock){
                        instream >> input;
                        switch(input){
                                case 'l':
                                        newBlock = !b.moveBlockLeft(curBlock);
                                        break;
                                case 'r':
                                        newBlock = !b.moveBlockRight(curBlock);
                                        break;
                                case 'g':
                                        newBlock = !b.moveBlockDown(curBlock);
                                        break;
                                case 'd':
                                        b.dropBlock(curBlock);
                                        newBlock = true;
                                        break;
                                case 't': // rotate를 수행할 t의 case를 새로 추가한다.
                                        newBlock = !b.rotateBlock(curBlock);
                                        break;
```

```
                                case 'q':
                                        newBlock = true;
                                        gameOver = true;
                                        break;
                                }
                                updateScreen();
                        }
                        newBlock = false;
                }
        }
```

8. MakeFile 만들기

새로이 추가된 파일은 빨간색으로 표시하였다.

```
CC = g++
TARGET = tetris
SOURCES = Tetris.cpp \
            Pane.cpp \
            BoardPane.cpp \
            InfoPane.cpp \
            HelpPane.cpp \
            StatPane.cpp \
            NextPane.cpp \
            Block.cpp \
            OBlock.cpp \
            IBlock.cpp \
            LBlock.cpp \
            JBlock.cpp \
            SBlock.cpp \
            ZBlock.cpp \
            TBlock.cpp \
            Board.cpp \
            main.cpp
LDFLAGS = -lncurses
all:
$(CC) -o $(TARGET) $(SOURCES) $(LDFLAGS)
clean:
rm -rf *.o $(TARGET)
```

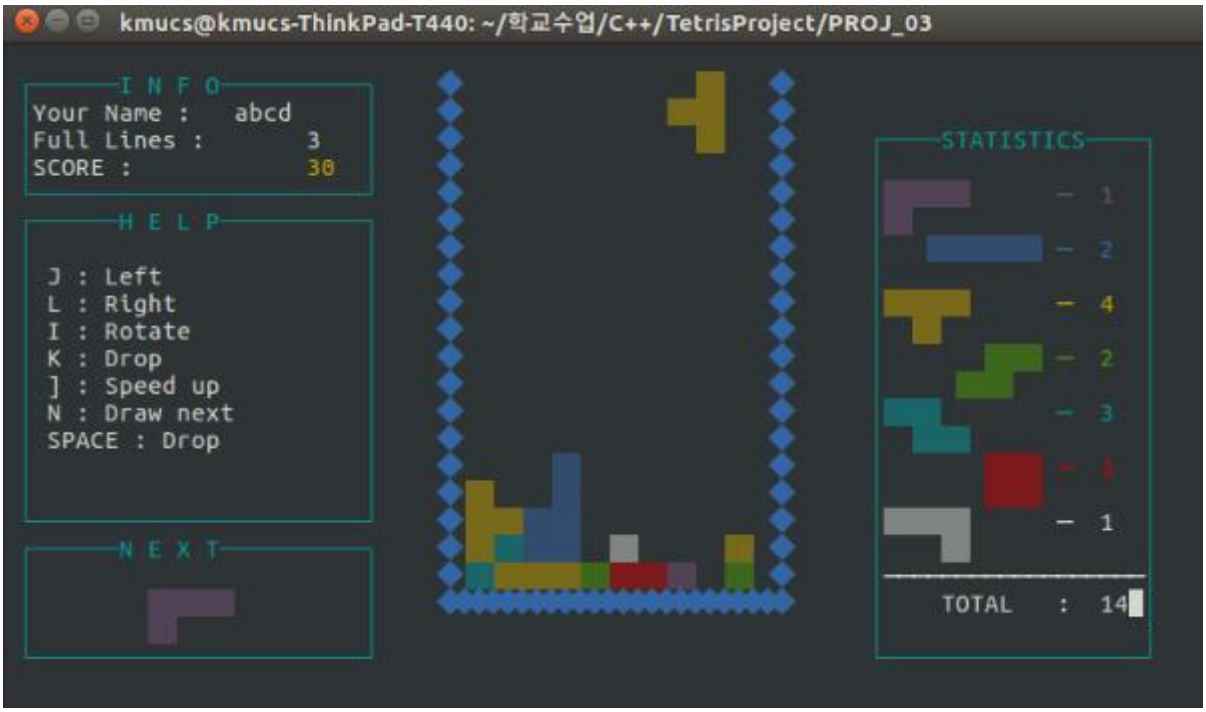
위와 같은 형태로 Makefile을 만들어준 후, 터미널 창에서 make 명령어를 통해 컴파일을 실시한다.

9. 결과물

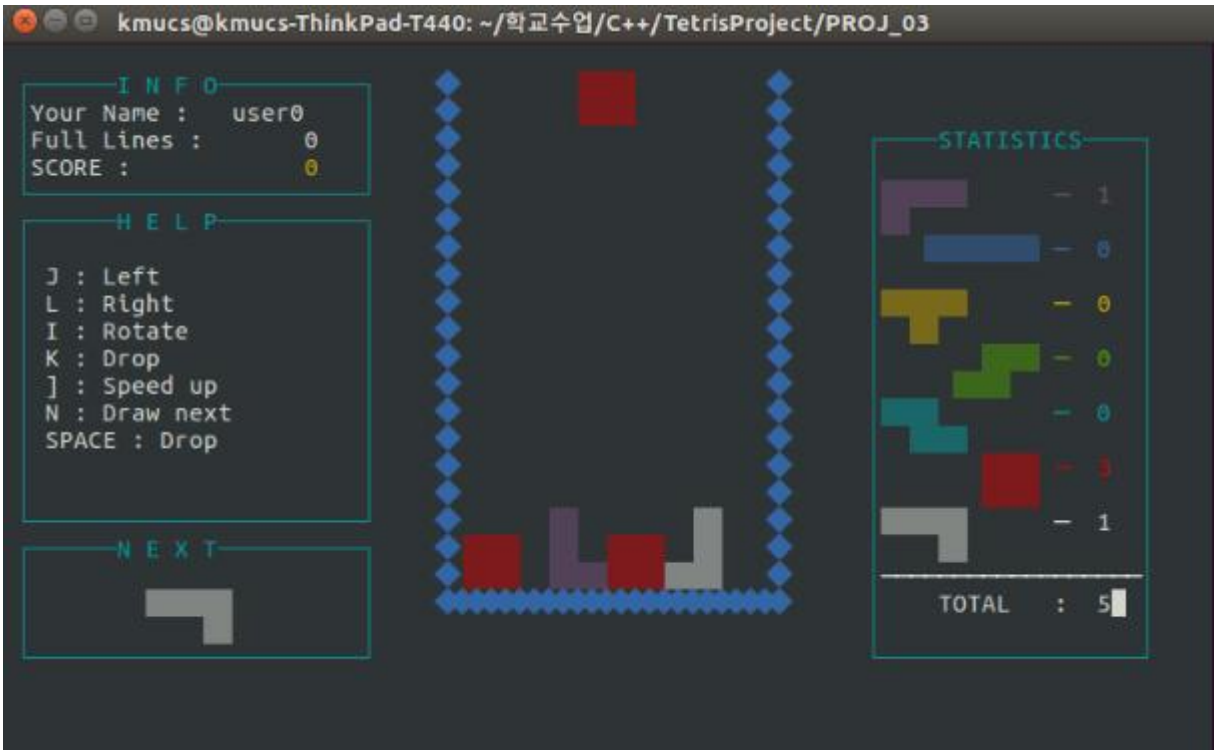
- play() 수행 시

  1) 시작 전에 사용자로부터 이름(abcd)을 입력받는다.

  2) 사용자로부터 이름을 입력받은 이후 게임을 수행하면, 사용자의 이름(abcd)이 InfoPane에 나타난다



- replay() 수행 시 : 사용자의 이름 변화(input.txt는 과제에 주어진 파일 활용)



10. 보완해야 할 사항 및 느낀점

3단계에서 블록을 rotate했을 때 바뀌는 좌표값의 범위가 BoardPane을 넘어가거나 이미 그려져있는 다른 블록의 범위를 침범했을 경우의 예외처리를 하지 못했다. 보완을 해보려고 노력해 보았으나, 어떻게 해야 보완을 할 수 있을지 해결책을 찾지 못해서 부득이하게 불완전한 상태로 제출하게 되었다. 현재는 시험기간이라 시간이 촉박하여 오랫동안 고민할 시간이 없어서 해결책을 찾지 못하였으나, 종강 후에 해결책을 고민하여 꼭 완벽한 테트리스를 구현하도록 할 것이다.

처음 테트리스 프로젝트를 받았을 때, ncurses라는 매우 생소한, 아니 처음으로 접하는 라이브러리로 인해 매우 혼란스러웠다. 도저히 어떻게 테트리스를 구현해야 할 지 감조차도 오지 않았었다. 하지만 임은진 교수님의 지도를 바탕으로 1단계부터 하나씩 차근차근 해 나가다 보니 불완전하긴 하지만 이러한 결과물을 만들어 낼 수 있었다. 사실 학기 초까지만 해도 과를 옮긴지라 리눅스가 무엇인지도 몰랐었고, C++의 객체지향 개념도 하나도 몰랐었는데, 내가 이런 결과물을 만들어 냈다는 사실이 믿기지가 않는다. 프로젝트를 통해 내가 얼마나 성장했는지를 체감할 수 있어서 매우 뿌듯하다. 프로젝트를 통해 얻은 교훈이 하나 있다면, 정말 어떻게 해야할지 감조차도 잡히지 않을만큼 어려워보이고 큰 문제일지라도, 작은 단계로 나누어 하나씩 차근차근 해결해 나간다면 결국에는 문제의 해결책에 근접할 수 있다는 사실이다. 프로그래밍에서 뿐만 아니라, 살아가면서 맞닥뜨리는 모든 문제에 적용될 수 있는 교훈이 아닐까 싶다. 이러한 프로젝트를 설계해주시고, 한 학기동안 좋은 수업을 통해 한층 성장할 수 있도록 도와주신 임은진 교수님과 김준호 교수님께 감사드린다.