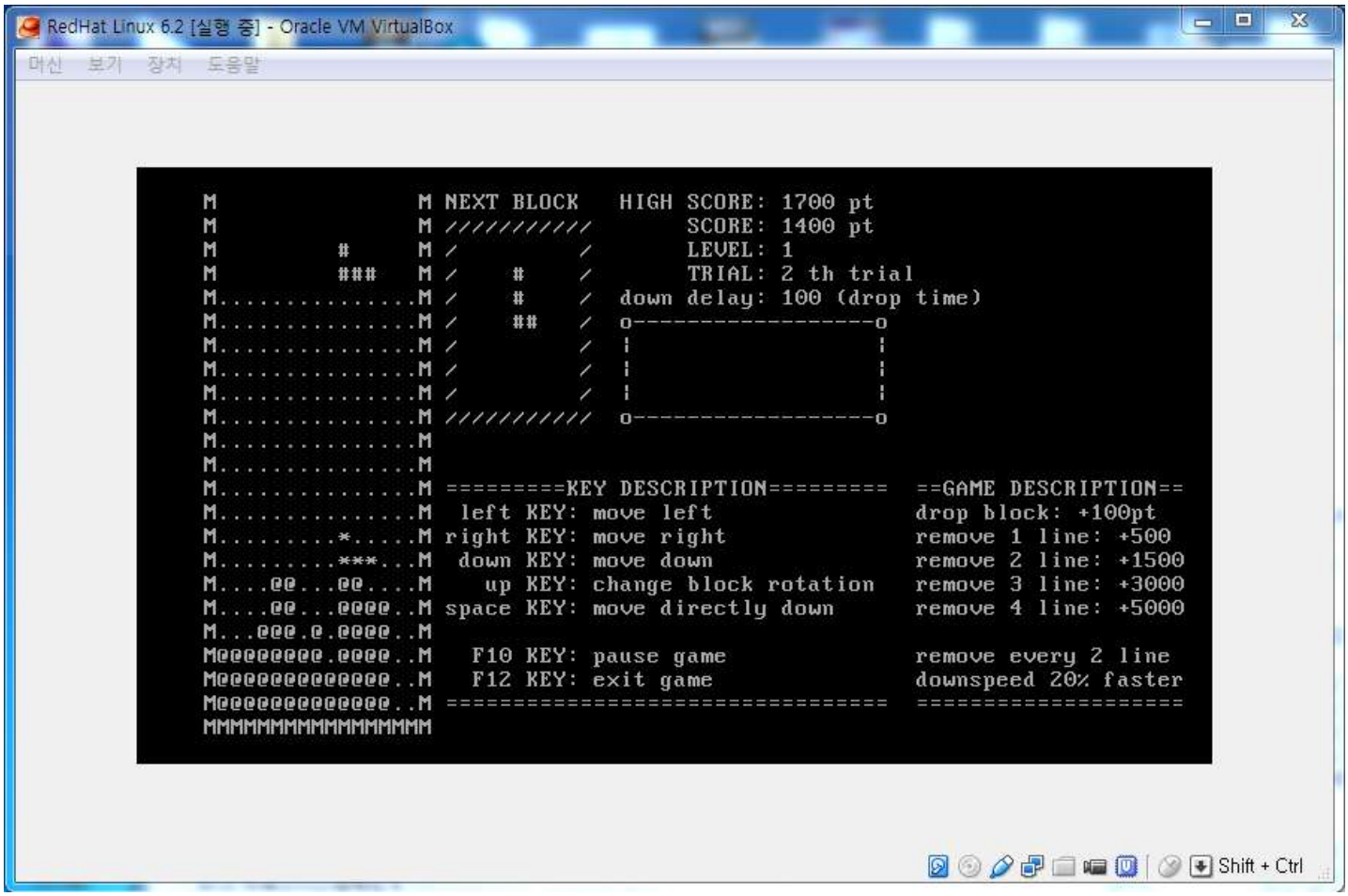
리눅스 환경(RedHat Linux 6.2) 에서 테트리스 게임 개발

양경석 (gaeng4@gmail.com)

<실행 화면>



<소스코드>

```
#include <ncurses.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

char block[7][4][4][2]={
        {
        {{ 0, 0},{-1, 0},{ 1, 0},{ 2, 0}},
        {{ 0, 0},{ 0,-1},{ 0, 1},{ 0, 2}},
        {{ 0, 0},{-2, 0},{-1, 0},{ 1, 0}},
        {{ 0, 0},{ 0, 1},{ 0,-1},{ 0,-2}}
        }, //block[0][][][] : long block

        {
        {{ 0, 0},{ 0, 1},{ 1, 0},{ 1, 1}},
        {{ 0, 0},{ 0, 1},{ 1, 0},{ 1, 1}},
        {{ 0, 0},{ 0, 1},{ 1, 0},{ 1, 1}},
        {{ 0, 0},{ 0, 1},{ 1, 0},{ 1, 1}}
        }, //block[1][][][] : rectangle block

        {
        {{ 0, 0},{ 0, 1},{-1, 0},{-2, 0}},
        {{ 0, 0},{-1, 0},{ 0,-1},{ 0,-2}},
        {{ 0, 0},{ 1, 0},{ 2, 0},{ 0,-1}},
        {{ 0, 0},{ 0, 1},{ 0, 2},{ 1, 0}}
        }, //block[2][][][] : L  block

        {
        {{ 0, 0},{-1, 0},{-2, 0},{ 0,-1}},
        {{ 0, 0},{ 1, 0},{ 0,-1},{ 0,-2}},
        {{ 0, 0},{ 0, 1},{ 1, 0},{ 2, 0}},
        {{ 0, 0},{ 0, 1},{ 0, 2},{-1, 0}}
        }, //block[3][][][] : reverse L block
```

```
            {
            {{ 0, 0},{ 1, 0},{ 0,-1},{-1,-1}},
            {{ 0, 0},{ 0, 1},{ 1, 0},{ 1,-1}},
            {{ 0, 0},{-1, 0},{ 0, 1},{ 1, 1}},
            {{ 0, 0},{ 0,-1},{-1, 0},{-1, 1}}
            }, //block[4][][][] : Y block

            {
            {{ 0, 0},{-1, 0},{ 0,-1},{ 1,-1}},
            {{ 0, 0},{ 0,-1},{ 1, 0},{ 1, 1}},
            {{ 0, 0},{ 1, 0},{ 0, 1},{-1, 1}},
            {{ 0, 0},{ 0, 1},{-1, 0},{-1,-1}}
            }, //block[5][][][] : reverse Y block

            {
            {{ 0, 0},{ 0, 1},{-1, 0},{ 0,-1}},
            {{ 0, 0},{ 1, 0},{ 0,-1},{-1, 0}},
            {{ 0, 0},{ 0, 1},{ 0,-1},{ 1, 0}},
            {{ 0, 0},{ 0, 1},{ 1, 0},{-1, 0}}
            }, //block[6][][][] : E block
              };

int x,y,b,r;
int b_next;
int shadow_y;
int strike_cnt;
int shadow_strike_cnt;
int strike_or_not;
int line_cnt[22]={0,};
int board[23][17]={0,};
int next_block_board[9][11]={0,};
int gap_var=0;
int game_cnt=1;
int score=0;
int line_score=0;
int high_score=0;
int ch;
int moved;
char gameover;
char yesno;
int i,j,k,l;
int pause_cnt;
int level_cnt;
int down_delay;
int time_start;
int time_end;
char game_msg[29];
char *tile[]={".", "#", "M", "@", " ", "*", "/"};

void board_arr(){
for(i=0;i<23;i++){
  for(j=0;j<17;j++){
   if( i>=0 && i<4 && j!=0 && j!=16){     board[i][j]=4; }
   else if( i==22 || j==0 || j==16){     board[i][j]=2; }
   else{                    board[i][j]=0; }
  }
}
}

void next_block_board_arr(){
for(i=0;i<9;i++){
  for(j=0;j<11;j++){
   if(i>0 && i<8 && j>0 && j<10){ next_block_board[i][j]=4; }
   else if(i==0 || i==8 || j==0 || j==10){ next_block_board[i][j]=6;}
  }
}
}

void board_print(){
for(i=0;i<23;i++){
  for(j=0;j<17;j++){
   move(1+i,5+j);
   addstr(tile[board[i][j]]);
  }
}

for(i=0;i<9;i++){
  for(j=0;j<11;j++){
   move(2+i,23+j);
   addstr(tile[next_block_board[i][j]);
  }
}
```

```
}


void showblock(){
for(i=0;i<4;i++){
  move(1+block[b][r][i][0]+y, 5+block[b][r][i][1]+x);
  addstr(tile[1]);
}
}


void show_next_block(){
for(i=0;i<4;i++){
  move(6+block[b_next][0][i][0], 28+block[b_next][0][i][1]);
  addstr(tile[1]);
}
}


void shadow(){
shadow_strike_cnt=0;
shadow_y=3;

while(shadow_strike_cnt!=1){
if( board[shadow_y+block[b][r][0][0]+1 ][x+block[b][r][0][1] ] ==2 ||
     board[shadow_y+block[b][r][1][0]+1 ][x+block[b][r][1][1] ] ==2 ||
     board[shadow_y+block[b][r][2][0]+1 ][x+block[b][r][2][1] ] ==2 ||
     board[shadow_y+block[b][r][3][0]+1 ][x+block[b][r][3][1] ] ==2 ||

     board[shadow_y+block[b][r][0][0]+1 ][x+block[b][r][0][1] ] ==3 ||
     board[shadow_y+block[b][r][1][0]+1 ][x+block[b][r][1][1] ] ==3 ||
     board[shadow_y+block[b][r][2][0]+1 ][x+block[b][r][2][1] ] ==3 ||
     board[shadow_y+block[b][r][3][0]+1 ][x+block[b][r][3][1] ] ==3 ){

  shadow_strike_cnt++;
  }
else(shadow_y++;}
}
if(shadow_strike_cnt==1){
  for(i=0;i<4;i++){
   move(1+block[b][r][i][0]+shadow_y, 5+block[b][r][i][1]+x);
   addstr(tile[5]);
  }
}
shadow_strike_cnt=0;
}


void move_down(){

if( board[y+block[b][r][0][0]+1 ][x+block[b][r][0][1] ] ==2 ||
     board[y+block[b][r][1][0]+1 ][x+block[b][r][1][1] ] ==2 ||
     board[y+block[b][r][2][0]+1 ][x+block[b][r][2][1] ] ==2 ||
     board[y+block[b][r][3][0]+1 ][x+block[b][r][3][1] ] ==2 ||

     board[y+block[b][r][0][0]+1 ][x+block[b][r][0][1] ] ==3 ||
     board[y+block[b][r][1][0]+1 ][x+block[b][r][1][1] ] ==3 ||
     board[y+block[b][r][2][0]+1 ][x+block[b][r][2][1] ] ==3 ||
     board[y+block[b][r][3][0]+1 ][x+block[b][r][3][1] ] ==3 ){

else{ y++; }
}


void strike_block(){

if( board[y+block[b][r][0][0]+1 ][x+block[b][r][0][1] ] ==2 ||
     board[y+block[b][r][1][0]+1 ][x+block[b][r][1][1] ] ==2 ||
     board[y+block[b][r][2][0]+1 ][x+block[b][r][2][1] ] ==2 ||
     board[y+block[b][r][3][0]+1 ][x+block[b][r][3][1] ] ==2 ||

     board[y+block[b][r][0][0]+1 ][x+block[b][r][0][1] ] ==3 ||
     board[y+block[b][r][1][0]+1 ][x+block[b][r][1][1] ] ==3 ||
     board[y+block[b][r][2][0]+1 ][x+block[b][r][2][1] ] ==3 ||
     board[y+block[b][r][3][0]+1 ][x+block[b][r][3][1] ] ==3 ){

   strike_cnt++;
}
}


void striked_block(){
```

```
int this_score;
if(strike_cnt==2){

    board[y+block[b][r][0][0] ][x+block[b][r][0][1] ] =3;
    board[y+block[b][r][1][0] ][x+block[b][r][1][1] ] =3;
    board[y+block[b][r][2][0] ][x+block[b][r][2][1] ] =3;
    board[y+block[b][r][3][0] ][x+block[b][r][3][1] ] =3;

    this_score=100
    score=score+this_score+line_score;
    if(high_score<=score){ high_score=score; }
    x=8;y=3;
    strike_cnt=0;
    strike_or_not=1;
}
}


void auto_move_down(){
time_end=clock();

if( (time_end-time_start) / 10000 ==down_delay ){
time_start=clock();
move_down();
gap_var=-1;
strike_block();
striked_block();
board_print();
showblock();
}
refresh();
}


void move_left(){

if( board[ y+block[b][r][0][0] ][x+block[b][r][0][1]-1 ] ==2 ||
    board[ y+block[b][r][1][0] ][x+block[b][r][1][1]-1 ] ==2 ||
    board[ y+block[b][r][2][0] ][x+block[b][r][2][1]-1 ] ==2 ||
    board[ y+block[b][r][3][0] ][x+block[b][r][3][1]-1 ] ==2 ||

    board[ y+block[b][r][0][0] ][x+block[b][r][0][1]-1 ] ==3 ||
    board[ y+block[b][r][1][0] ][x+block[b][r][1][1]-1 ] ==3 ||
    board[ y+block[b][r][2][0] ][x+block[b][r][2][1]-1 ] ==3 ||
    board[ y+block[b][r][3][0] ][x+block[b][r][3][1]-1 ] ==3 ){}

else { x--; }

}


void move_right(){

if( board[ y+block[b][r][0][0] ][x+block[b][r][0][1]+1 ] ==2 ||
    board[ y+block[b][r][1][0] ][x+block[b][r][1][1]+1 ] ==2 ||
    board[ y+block[b][r][2][0] ][x+block[b][r][2][1]+1 ] ==2 ||
    board[ y+block[b][r][3][0] ][x+block[b][r][3][1]+1 ] ==2 ||

    board[ y+block[b][r][0][0] ][x+block[b][r][0][1]+1 ] ==3 ||
    board[ y+block[b][r][1][0] ][x+block[b][r][1][1]+1 ] ==3 ||
    board[ y+block[b][r][2][0] ][x+block[b][r][2][1]+1 ] ==3 ||
    board[ y+block[b][r][3][0] ][x+block[b][r][3][1]+1 ] ==3 ){}

else { x++; }

}

void rotation(){

if( board[ y+block[b][(r+1)%4][0][0] ][x+block[b][(r+1)%4][0][1] ] ==2 ||
    board[ y+block[b][(r+1)%4][1][0] ][x+block[b][(r+1)%4][1][1] ] ==2 ||
    board[ y+block[b][(r+1)%4][2][0] ][x+block[b][(r+1)%4][2][1] ] ==2 ||
    board[ y+block[b][(r+1)%4][3][0] ][x+block[b][(r+1)%4][3][1] ] ==2 ||

    board[ y+block[b][(r+1)%4][0][0] ][x+block[b][(r+1)%4][0][1] ] ==3 ||
    board[ y+block[b][(r+1)%4][1][0] ][x+block[b][(r+1)%4][1][1] ] ==3 ||
    board[ y+block[b][(r+1)%4][2][0] ][x+block[b][(r+1)%4][2][1] ] ==3 ||
    board[ y+block[b][(r+1)%4][3][0] ][x+block[b][(r+1)%4][3][1] ] ==3 ){}

else { r=(r+1)%4; }

}
```

```
void line_check(){
int remove_line;

for(i=0;remove_line=0;i<22;i++){
  line_cnt[i]=0;
  for(j=0;j<16;j++){
   if( board[i][j]==3 ){ line_cnt[i]++; }
   if( line_cnt[i]==15{
    level_cnt++;
    remove_line++;
     if(level_cnt%2==0){down_delay*=0.8;}
     time_start=clock();
     if(remove_line==1){
          score+=500;
          mvprintw(7,38," remove 1 line! ");
          mvprintw(8,38,"  you got bonus ");
          mvprintw(9,38,"  +500 point!!! ");
          refresh();}

     if(remove_line==2){
          score+=1000;
          mvprintw(7,38," remove 2 line! ");
          mvprintw(8,38,"  you got bonus ");
          mvprintw(9,38," +1500 point!!! ");
          refresh();}

     if(remove_line==3){
          score+=1500;
          mvprintw(7,38," remove 3 line! ");
          mvprintw(8,38,"  you got bonus ");
          mvprintw(9,38," +3000 point!!! ");
          refresh();}

     if(remove_line==4){
          score+=2000;
          mvprintw(7,38,"*****TETRIS*****");
          mvprintw(8,38,"* BONUS  +5000 *");
          mvprintw(9,38,"*****************");
          refresh();}

     if(high_score<=score){ high_score=score; }
          for(l=i;l>=4;l--){
          for(k=1;k<16;k++){
            board[l][k]=board[l-1][k];
            if(l==4){ board[l][k]=0;} //make top line "." not " "
          }
          }
     }
  }
}
refresh();
}


void initiate_tetris(){
clear();
score=0;
ch=0;
x=8;
y=3;
shadow_y=3;
srand(time(NULL));
b=rand()%7;
b_next=rand()%7;
r=0;
strike_cnt=0;
shadow_strike_cnt=0;
pause_cnt=0;
level_cnt=0;
down_delay=100;
board_arr();
next_block_board_arr();
strike_or_not=0; //0: not strike 1:strike
gap_var=0;
}


void message_box(){
refresh();
mvprintw(1,23,"NEXT BLOCK");
```

```
refresh();
mvprintw(13,23,"==========KEY DESCRIPTION=========");
mvprintw(14,23," left KEY: move left");
mvprintw(15,23,"right KEY: move right");
mvprintw(16,23," down KEY: move down");
mvprintw(17,23,"   up KEY: change block rotation");
mvprintw(18,23,"space KEY: move directly down");
mvprintw(20,23,"  F10 KEY: pause game");
mvprintw(21,23,"  F12 KEY: exit game");
mvprintw(22,23,"==================================");
refresh();

mvprintw( 2,36,"HIGH SCORE: %d pt",high_score);
mvprintw( 3,36,"     SCORE: %d pt",score);
mvprintw( 4,36,"     TRIAL: %d th trial",game_cnt);
mvprintw( 5,36,"down delay: %d (drop time)",down_delay);
mvprintw( 6,36,"o-----------------o",high_score);
mvprintw( 7,36,"|");mvprintw(7,55,"|");
mvprintw( 8,36,"|");mvprintw(8,55,"|");
mvprintw( 9,36,"|");mvprintw(9,55,"|");
mvprintw(10,36,"o-----------------o");
refresh();
mvprintw(13, 58,"==GAME DESCRIPTION==");
mvprintw(14, 58,"drop block: +100pt");
mvprintw(15, 58,"remove 1 line: +500");
mvprintw(16, 58,"remove 2 line: +1500");
mvprintw(17, 58,"remove 3 line: +3000");
mvprintw(18, 58,"remove 4 line: +5000");
mvprintw(20, 58,"remove every 2 line");
mvprintw(21, 58,"downspeed 20%% faster");
mvprintw(22, 58,"==================");
}

int main(){
ch=0;
initscr();
curs_set(0);
keypad(stdscr,TRUE);
time_start=clock();
nodelay(stdscr,TRUE);
initiate_tetris();

while(1){
message_box();
if(strike_or_not==1){ //0: not strike 1: strike
b=b_next;
b_next=rand()%7;
strike_or_not=0;
}
board_print();
shadow();
showblock();
show_next_block();
refresh();
if( (ch=getch() ) == ERR ) { auto_move_down(); }
else{
    switch(ch){
            case KEY_UP:
                    rotation();
                    refresh();
                    break;
            case KEY_DOWN:
                    move_down();
                    gap_var=-1;
                    strike_block();
                    striked_block();
                    refresh();
                    break;
            case KEY_LEFT:
                    move_left();
                    refresh();
                    break;
            case KEY_RIGHT:
                    move_right();
                    refresh();
                    break;
            case ' ':
                    while(strike_cnt!=1){
                        move_down();
                        strike_block();
                        }
                    strike_cnt++;
```

```
                    gap_var=0;
                    striked_block();
                    gap_var=-1;
                    refresh();
                    strike_cnt=0;
                    break;
            case KEY_F(12):
                    mvprintw(7,38,"    GAME OVER    ");
                    mvprintw(8,38,"                 ");
                    mvprintw(9,38,"    GOOD BYE!    ");
                    refresh();
                    sleep(1);
                    endwin();
                    exit(0);
                    break;
            case KEY_F(10):
                    while( getch()!=KEY_F(10) ){
                    mvprintw(7,38,"    GAME PAUSE    ");
                    mvprintw(8,38,"  press F10 KEY   ");
                    mvprintw(9,38,"  to resume GAME  ");
                    refresh();
                    sleep(1);
                    }
                    mvprintw(7,38,"  GAME RESTART    ");
                    mvprintw(8,38,"    E N J O Y     ");
                    mvprintw(9,38,"   T E T R I S    ");
            default:

        }//end switch(ch)
    }//end else

    line_check();
    refresh();

/*    GAME OVER CHECK          */

for(i=0;i<16;i++){
if(board[3][i]==3){
  while(1){
   yesno==getch();
    refresh();
    mvprintw(7,38,"    GAME OVER    ");
    mvprintw(8,38,"                 ");
    mvprintw(9,38,"TRY AGAIN? (Y/N)");
    refresh();
     if(yesno=='y'){
     game_cnt++;
     initiate_tetris();
     gameover=1;
     break;
     }
     else if(yesno=='n'){
     refresh();
     mvprintw(7,38,"    GAME OVER    ");
     mvprintw(8,38,"                 ");
     mvprintw(9,38,"    GOOD BYE!    ");
     refresh();
     sleep(1);
     endwin();
     exit(0);
     }
  }
}
}
refresh();
}//end while
endwin();

return 0;
}
```

구현된 내용 및 구현할 내용.

```
#include <ncurses.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

char block[7][4][4][2]={
```

```
            {
            { { 0, 0}, {-1, 0}, { 1, 0}, { 2, 0} },
            { { 0, 0}, { 0,-1}, { 0, 1}, { 0, 2} },
            { { 0, 0}, {-2, 0}, {-1, 0}, { 1, 0} },
            { { 0, 0}, { 0, 1}, { 0,-1}, { 0,-1} }
            }, //block[0][][][]: 일자블록

            {
            { { 0, 0}, { 0, 1}, { 1, 0}, { 1, 1} },
            { { 0, 0}, { 0, 1}, { 1, 0}, { 1, 1} },
            { { 0, 0}, { 0, 1}, { 1, 0}, { 1, 1} },
            { { 0, 0}, { 0, 1}, { 1, 0}, { 1, 1} }
            }, //block[1][][][]: 네모블럭

            {
            { { 0, 0}, { 0, 1}, {-1, 0}, {-2, 0} },
            { { 0, 0}, {-1, 0}, { 0,-1}, { 0,-2} },
            { { 0, 0}, { 1, 0}, { 2, 0}, { 0,-1} },
            { { 0, 0}, { 0, 1}, { 0, 2}, { 1, 0} }
            }, //block[2][][][]: L자 블럭

            {
            { { 0, 0}, {-1, 0}, {-2, 0}, { 0,-1} },
            { { 0, 0}, { 1, 0}, { 0,-1}, { 0,-2} },
            { { 0, 0}, { 0, 1}, { 1, 0}, { 2, 0} },
            { { 0, 0}, { 0, 1}, { 0, 2}, {-1, 0} }
            }, //block[3][][][]: 거꾸로된 L자 블록

            {
            { { 0, 0}, { 1, 0}, { 0,-1}, {-1,-1} },
            { { 0, 0}, { 0, 1}, { 1, 0}, { 1,-1} },
            { { 0, 0}, {-1, 0}, { 0, 1}, { 1, 1} },
            { { 0, 0}, { 0,-1}, {-1, 0}, {-1, 1} }
            }, //block[4][][][]: Y자 블럭

            {
            { { 0, 0}, {-1, 1}, { 0,-1}, { 1,-1} },
            { { 0, 0}, { 0,-1}, { 1, 0}, { 1, 1} },
            { { 0, 0}, { 1, 0}, { 0, 1}, {-1, 1} },
            { { 0, 0}, { 0, 1}, {-1, 0}, {-1,-1} }
            }, //block[5][][][]: 거꾸로된 Y자 블록

            {
            { { 0, 0}, { 0, 1}, {-1, 0}, { 0,-1} },
            { { 0, 0}, { 1, 0}, { 0,-1}, {-1, 0} },
            { { 0, 0}, { 0, 1}, { 0,-1}, { 1, 0} },
            { { 0, 0}, { 0, 1}, { 1, 0}, {-1, 0} }
            }, //block[6][][][]: ㅗ블록
                };
block[7][4][4][2] //7가지 블록, 4가지 회전, 4칸의 블록, y축/x축 을 저장

int x,y,b,r; //계속 변화시키면서 적용해야 하는 변수들은 전역변수 선언
          //x: x축 위치, y: y축 위치, b: 블록의 종류(0~6까지 7가지 종류), r: 회전
int strike_cnt; // 바닥이나 블록에 부딪힌 횟수
int line_cnt[22]={0,}; //각 라인에서의 블록의 갯수 저장
int board[23][17]={0,}; // 테트리스의 배경 및 벽, 쌓인 블록 저장
int gap_var=0; //스페이스바 눌렀을 때에는 벽에 한번만 부딪혀도 쌓이도록 조정할때 활용
int i,j,k,l; //함수 내에서 자주 쓰는 변수이므로 그냥 전역변수로 한번만 선언해줌
char *tile[]={".", "#", "M", "@"}; //tile[]배열에 0번은 배경, 1번은 쌓이기 전 블록,
                           //2번은 벽, 3번은 쌓인 후 벽을 저장

void board_arr(){ //보드 배열에 저장만 함
for(i=0;i<23;i++){
  for(j=0;j<17;j++){
   if( i==22 || j==0 || j==16){  //y축 마지막줄, x축 첫번째 & 마지막 줄은 벽(M)으로 셋팅
    board[i][j]=2;
   }//END if
   else{
    board[i][j]=0; //나머지는 ..으로 세팅(저장만 함)
   }//END else
  }//END for(j)
}//END for(i)
}//END board_arr func


void board_print(){ //보드 배열에 저장된 것을 출력만 함
for(i=0;i<23;i++){
  for(j=0;j<17;j++){
   move(1+i,5+j); //1,5만큼 이동시킨 상태에서 board[i][j]에 저장된 타일 출력시킴
   addstr(tile[board[i][j]);
  }//END for(j)
}//END for(i)
```

```
}//END board_print func


void showblock(){
for(i=0;i<4;i++){
  move( 1+block[b][r][i][0]+y, 5+block[b][r][i][1]+x );
  addstr(tile[1]); //기본세팅인 1,5이동후 x축, y축 이동한 만큼 위치에 블록 출력
}//END for(i)
}//END showblock func


void move_down(){
if( board[ y+block[b][r][0][0] ][ x+block[b][r][0][1] ]==2 ||
    board[ y+block[b][r][1][0] ][ x+block[b][r][1][1] ]==2 ||
    board[ y+block[b][r][2][0] ][ x+block[b][r][2][1] ]==2 ||
    board[ y+block[b][r][3][0] ][ x+block[b][r][3][1] ]==2 ||

    board[ y+block[b][r][0][0] ][ x+block[b][r][0][1] ]==3 ||
    board[ y+block[b][r][1][0] ][ x+block[b][r][1][1] ]==3 ||
    board[ y+block[b][r][2][0] ][ x+block[b][r][2][1] ]==3 ||
    board[ y+block[b][r][3][0] ][ x+block[b][r][3][1] ]==3 ){}
else{ y++; } //아래로 이동중에 벽(2)이나 쌓인블록(3)이 없으면 계속 y축 이동
}//END move_down func


void strike_block(){
if( board[ y+block[b][r][0][0]+1 ][ x+block[b][r][0][1] ]==2 ||
    board[ y+block[b][r][1][0]+1 ][ x+block[b][r][1][1] ]==2 ||
    board[ y+block[b][r][2][0]+1 ][ x+block[b][r][2][1] ]==2 ||
    board[ y+block[b][r][3][0]+1 ][ x+block[b][r][3][1] ]==2 ||

    board[ y+block[b][r][0][0]+1 ][ x+block[b][r][0][1] ]==3 ||
    board[ y+block[b][r][1][0]+1 ][ x+block[b][r][1][1] ]==3 ||
    board[ y+block[b][r][2][0]+1 ][ x+block[b][r][2][1] ]==3 ||
    board[ y+block[b][r][3][0]+1 ][ x+block[b][r][3][1] ]==3 ){
//아래로 이동중에 한칸 아랫쪽에 벽(2)이나 쌓인블록(3)이 있으면 strike_cnt 증가시킴
  strike_cnt++;
}//END if
}


void striked_block(){

if(strike_cnt==2){
    board[ y+block[b][r][0][0]+gap_var ][ x+block[b][r][0][1] ]==3;
    board[ y+block[b][r][1][0]+gap_var ][ x+block[b][r][1][1] ]==3;
    board[ y+block[b][r][2][0]+gap_var ][ x+block[b][r][2][1] ]==3;
    board[ y+block[b][r][3][0]+gap_var ][ x+block[b][r][3][1] ]==3;
//바로 아랫줄에 블록이 있는 경우에 한번 더 아래 키를 누르면 쌓임
//스페이스바 누른 때에는 한번만 벽or쌓인블록 만나면 블록 쌓으면 됨.
//이때에 y가 한칸 덜 증가하므로 gap_var를 하나 줄여줌으로써 ↓키 누를때와 구별해줌

x=8;y=3;b=rand()%7; //부딪친 후 새로운 블록 위치 및 모양을 셋팅해줌.
strike_cnt=0; //부딪친 횟수를 다시 체크해야 하므로 0으로 초기화
}//END if
}//END striked_block func


void move_left(){

if( board[ y+block[b][r][0][0] ][ x+block[b][r][0][1]-1 ]==2 ||
    board[ y+block[b][r][1][0] ][ x+block[b][r][1][1]-1 ]==2 ||
    board[ y+block[b][r][2][0] ][ x+block[b][r][2][1]-1 ]==2 ||
    board[ y+block[b][r][3][0] ][ x+block[b][r][3][1]-1 ]==2 ||

    board[ y+block[b][r][0][0] ][ x+block[b][r][0][1]-1 ]==3 ||
    board[ y+block[b][r][1][0] ][ x+block[b][r][1][1]-1 ]==3 ||
    board[ y+block[b][r][2][0] ][ x+block[b][r][2][1]-1 ]==3 ||
    board[ y+block[b][r][3][0] ][ x+block[b][r][3][1]-1 ]==3 ){}

else{ x--; } //왼쪽으로 이동중에 벽(2)이나 쌓인블록(3)이 없으면 계속 왼쪽으로 이동
}//END move_left func


void move_right(){

if( board[ y+block[b][r][0][0] ][ x+block[b][r][0][1]+1 ]==2 ||
    board[ y+block[b][r][1][0] ][ x+block[b][r][1][1]+1 ]==2 ||
    board[ y+block[b][r][2][0] ][ x+block[b][r][2][1]+1 ]==2 ||
    board[ y+block[b][r][3][0] ][ x+block[b][r][3][1]+1 ]==2 ||

    board[ y+block[b][r][0][0] ][ x+block[b][r][0][1]+1 ]==3 ||
```

```
    board[ y+block[b][r][1][0] ][ x+block[b][r][1][1]+1 ]==3 ||
    board[ y+block[b][r][2][0] ][ x+block[b][r][2][1]+1 ]==3 ||
    board[ y+block[b][r][3][0] ][ x+block[b][r][3][1]+1 ]==3 ){}

else{ x++; } //오른쪽으로 이동중에 벽(2)이나 쌓인블록(3)이 없으면 계속 오른쪽으로 이동
}//END move_right func


void rotation(){

if( board[ y+block[b][(r+1)%4][0][0] ][ x+block[b][r][0][1] ]==2 ||
    board[ y+block[b][(r+1)%4][1][0] ][ x+block[b][r][1][1] ]==2 ||
    board[ y+block[b][(r+1)%4][2][0] ][ x+block[b][r][2][1] ]==2 ||
    board[ y+block[b][(r+1)%4][3][0] ][ x+block[b][r][3][1] ]==2 ||

    board[ y+block[b][(r+1)%4][0][0] ][ x+block[b][r][0][1] ]==3 ||
    board[ y+block[b][(r+1)%4][1][0] ][ x+block[b][r][1][1] ]==3 ||
    board[ y+block[b][(r+1)%4][2][0] ][ x+block[b][r][2][1] ]==3 ||
    board[ y+block[b][(r+1)%4][3][0] ][ x+block[b][r][3][1] ]==3 ){}

else{ r=(r+1)%4; } //회전 방향의 다음 회전시에 벽이나 블록에 부딪치지 않으면 회전.
}//END rotation func


void line_check(){

for(i=0;i<22;i++){ link_cnt[i]=0;
  for(j=0;j<16;j++){
   if( board[i][j]==3 ){ line_cnt[i]++; } //각 라인의 블록수를 배열에 저장

   if( line_cnt[i]==15){ //0부터 증가시킨 어떤 라인의 블록수가 15개(한줄 전체)이면
    for(l=i;l>=1;l--){
     for(k=1;j<14;k++){
      board[l][k]=board[l-1][k]; //(어떤 라인의 한줄 위부터 어떤 라인으로 옮김)
                                 //어차피 어떤 라인은 지워져야 할 라인이므로
    }//END for(k)                //즉시 윗줄을 옮겨와도 상관없다.
   }//END for(l)                 //윗줄부터 아랫쪽으로 체크하므로 2줄 이상도 OK
  }//END if
 }//END for(j)
}//END for(i)
}//END line_check func


int main(){
int ch;

initscr();
curs_set(0);
keypad(stdscr,TRUE);

//start_color();
//init_pair(1,COLOR_RED,COLOR_RED);
//attron( COLOR_PAIR(1) );

x=8; //가운데
y=3; //회전해도 보드 밖으로 나가지 않도록

srand(time(NULL)); //rand함수의 규칙성을 시간함수로 불규칙하게 만듬
b=rand()%7; //7가지 중 랜덤한 모양 나오도록.
r=0; //회전 방향 기본셋팅->이후 반시계방향으로 회전

strike_cnt=0; //부딪친 횟수 0회로 초기화

board_arr(); //보드배열 기본세팅

while(1){

refresh(); //물리적 리프레시 but 안해줘도 작동은 됨.

board_print();
showblock();

ch=getch();

switch(ch){
        case KEY_UP:
                rotation();
                break;
        case KEY_DOWN:
                move_down();
                gap_var=-1;
                strike_block();
```

```
                striked_block();
                break;
        case KEY_LEFT:
                move_left();
                break;
        case KEY_RIGHT:
                move_right();
                break;
        case ' ': //space bar
                while(strike_cnt!=1){ //부딪친횟수 1회될때까지
                move_down();        //아래로 움직이며
                strike_block();     //부딪히는지 체크
                }
                strike_cnt=2; //부딪힌 횟수1이지만 strike_cnt가 2인 경우 블록이 쌓이므로
                gap_var=0;   //스페이스바 눌렀을때 그 줄에 블록 쌓이도록 처리
                striked_block(); //블록 쌓이는것 처리
                break;
        }//END SWITCH(ch)
line_check(); //지워지는 라인 있는지 체크
}//END WHILE(1)

endwin();
return 0;
}
```

구현한 것
-↑키 누르면 돌아감
-←→ 키 누르면 움직임
-↓키 누르고 벽 만나면 한번 더 누른 경우 쌓임
-상하좌우 키 누른 경우 벽이나 블록 만나면 회전x, 움직임x 처리
-스페이스바 누르면 벽이나 블록 만나면 바로 쌓임
-블록 랜덤으로 나오도록 세팅
-어떤 라인이 블록으로 다 채워진 경우 지워지고 윗 블록이 아래로 이동해서 쌓임
-테트리스게임이라는 타이틀 출력
-3번째 줄까지는 "."을 비워둠(경계선 체크)
-맨 윗칸이 넘어서 블록이 채워진 경우 게임오버 출력
-게임오버되면 GAME OVER출력하고 RETRY? 해서 Y 선택하면 다시 시작하기.
-게임 종료시키기 위해서 F12키를 누르면 종료된다는 메시지 출력
-F12키 누르면종료하기
-점수기능(블록 쌓을때마다 100점 / 한줄 지울때마다 1000점)
-최고기록 점수
-시간이 지나면 블록이 내려가는 기능. 레벨이 오를수록 더 빨리 내려가도록.
-아래에 떨어질 자리 표시하기.
-시도 횟수
-다음 블록은 무엇인지 예고하는 기능
점수 계산(1줄 +500 2줄 +1500 3줄 +3000 4줄 +5000)
pause기능
game message창에 메시지 띄우기.(1줄인지 2줄인지 3줄인지 4줄인지 출력 완료. pause 메시지.종료메시지 띄우기(8,38)
블록 다 쌓이면 정지
레벨표시
진행시간 표시

*구현시 문제가 되었던 점

getch()는 blocking function.
->키입력이 없는 경우 아래로 움직이고 키 입력 대기 하지 않는 non-blocking이 되어야 함.
http://hughm.cs.ukzn.ac.za/~murrellh/os/notes/ncurses.html
위 사이트에서 ncurses programming guide 발견.
nodelay(stdscr,TRUE)해주면 blocking되지 않고, 키입력이 없다면 ERR을 반환한다.

sleep(1)은 시스템 자체를 일정 시간동안 멈추는 함수임.
따라서 sleep함수를 만나면 키보드 입력은 가능하나 이를 저장했다가 sleep시간이 지난 후에 움직임
->시간은 따로 흘러가고 그 사이에도 키보드가 움직일 수 있어야 함
->how?
http://programmersheaven.com/discussion/365393/delay-function-in-gcc
ID: sarfrajkhan
you can use just an empty for loop
like below,
for(i=0;i<10000;i++)
{}
it will create the delay in your program

time.h 인클루드 하여 clock() 함수를 통해 시간1과 시간2의 차이를 구함. 이 시간 차이가 100이 되면 아래로 움직이게 함.
빠른 진행을 위해 라인 2줄 제거시마다 20%의 딜레이 감소를 통해 속도 증가시킴.
time()함수는 1초 단위밖에 처리하지 못함. clock()함수는 0.01초 단위까지 컨트롤 가능하므로 clock함수를 활용함.