<c++ tetris project>

과목 : c++ 프로그래밍

담당교수 : 최준수

학과 : 소프트웨어융합학부

학번 : 20143104

이름 : 조승현

테트리스 코드를 짜기 위해 10개의 cpp파과 1개의 헤더파일로 구성했다.

1개의 헤더파일에는 각각 cpp파일에 필요한 클래스와 함수들이 선언되어있다.

먼저 클래스부터 살펴보면,

Block class

멤버변수 : 각각의 좌표를 결정할 변수 8개와 블록의 회전 모양을 나타내는 state 각 각의 모양 특성을 결정할 trait가 있다.

int x1,x2,x3,x4,y1,y2,y3,y4; int state,trait;

멤버함수 : 가상함수로 board배열에 넣어주는 create 함수와, 블록이 움직일 수 있는 지 검사하는 check 함수,이동시키는 move,rotate 함수 및 접근자 함수가 있다.

virtual void createblock(int board[][20]);

virtual void Movedown(int board[][20]);

virtual bool Checkdown(int board[][20]);

class Sblock : public Block

모양클래스들은 Block 클래스를 상속하며 대표적으로 S-block을 예로들면

멤버변수 : 블록이 몇 개 사용되는 지를 나타내는 정적변수 num을 선언한다

static int num; int Sblock::num=0;

멤버함수 : 생성자에는 block 클래스에서 정의된 멤버변수들을 초기화 한다.

x1=4; x2=4; x3=5; x4=5;y1=0; y2=1; y3=1; y4=2; state=0; trait=3;

createblock : 생성자에서 설정된 좌표 8개를 통해 board배열에 넣어주는 함수이다.

board[x1][y1]=3; board[x2][y2]=3; board[x3][y3]=3; board[x4][y4]=3; num++;

Checkdown(board) : 각각의 이동하는 방향에 따라 이동할 수 있는지를 검사 하는 함수이다.

블록 회전모양에 따라 경우를 나눈다. 기본 상태인 state=0; 일 때

아래 방향인 경우 맨 밑 점들인 (x2,y2) (x4,y4) 밑에 블록이 있거나 board 배열을 벗어나면 안되므로 움직일 수 없으므로 아래와 같이 설계한다.

if(state==0){

if(y4+1<20 &&board[x4][y4+1]==0&& board[x2][y2+1]==0)

return true;

else

return false;

}

moveDown(board) : 방향에 따라 블록을 움직이는 함수이다.

블록 회전모양에 따라 경우를 나눈다. 기본 상태인 state=0; 일 때

아래 방향인 경우 각 점들을 한 칸씩 내리고 맨 위 점 들은 0으로 초기화한다.

if(state==0){

board[x1][y1+1]=board[x1][y1];

board[x2][y2+1]=board[x2][y2];

board[x3][y3+1]=board[x3][y3];

board[x4][y4+1]=board[x4][y4];

board[x1][y1]=0;

board[x3][y3]=0;

y1++; y2++; y3++; y4++;

}

Checkrotate(board) : 블록이 회전 되면서 이동할 점에 블록이 있거나 회전 되면서 배열을 벗어날 경우를 반환하는 함수이다.

if(state==0){

if(x1-1>=0&&board[x1-1][y1+1]==0&&board[x4][y4-2]==0)

return true;

else

return false;

}

rotate(board) : 블록을 회전시키는 함수이다. 블록 회전모양에 따라 경우를 나누고 회전시킨 다음 state를 1증가 시킨 후 블록의 회전 주기에 따라 %연산 해준다. 회전시킬 때는 기존 점들을 0으로 초기화 하고 이동할 점들에 해당 하는 위치에 다시 board배열에 설정 하고 점 위치를 변경한다.

if(state==0){

board[x1][y1]=0; board[x2][y2]=0; board[x3][y3]=0; board[x4][y4]=0;

board[x1-1][y1+1]=3; board[x2][y2]=3; board[x3-1][y3-1]=3; board[x4][y4-2]=3;

x1-=1; y1+=1; x3-=1; y3-=1; y4-=2;

}

state+=1; state%=2;

isGameover : 초기 블록이 해당하는 지점에 블록이 있다면 true를 반환하는 함수이다.

if(board[x1][y1]!=0 || board[x2][y2]!=0 || board[x3][y3]!=0 || board[x4][y4]!=0)

return true;

else

return false;

tetris class

멤버함수 :

tetris():

initscr();

start\_color();

curs\_set(0);

noecho();

각각의 pane 클래스 실행

incolor() : 테트리스게임에서 필요할 색들을 전부 지정해주는 함수이다.

init\_pair(1, COLOR\_BLUE, COLOR\_BLUE);

updateScreen() : 게임 초기화면을 만들어 주는 함수이다.

infoPane\_->draw();

helpPane\_->draw();

nextPane\_->draw();

boardPane\_->draw();

statPane\_->draw();

update0000pane() : 게임 진행시 각각 pane을 갱신해 주는 함수이다.

ex)statPane\_->printnum(block);

play(board,\*gameblock,inkey) : 사용자로부터 입력 받은 키 및 파일로 블록을 이동시키는 함수이다.

main 함수로부터 inkey 로부터 받은 키로 동작한다. case 3 은 파일로부터

입력되었을 때 changefilekey 로 통해 정수로 치환되어 있는 수다. 대표적인 예로

아래 방향은 아래와 같이 설계되어있다.

switch(inkey){

case KEY\_DOWN:

case 3:

if(block.Checkdown(board))

block.Movedown(board);

updateboardpane(board);

break;

}

isFullline(board,sc) : 블록이 놓이게 되었을 때 1줄이 다 차있는지 검사하는 함수이다.

board 배열에서 각 줄을 돌면서 0이 아니면 sum를 증가연산 해서 sum==10이 되면 remove 함수와 moveGravity 함수를 실행한다.

그 후에 score객체에 선언된 설정자 함수들을 실행한다.

for(int j=0; j<20; j++){

int sum=0;

for(int i=0; i<10; i++){

if(board[i][j]!=0)

sum++;

}

if(sum==10){

count++;

remove(board,j);

moveGravity(board,j);

}

}

for(int i=0; i<count; i++)

sc.upScore();

sc.setLevel();

sc.setNumblock();

statPane\_->printscore(sc);

remove(board,row) : isFullline 함수에서 블록이 제거되는 함수이다. 그 줄을 0으로 초기화 한다.

for(int i=0; i<10; i++){

board[i][row]=0;

}

moveGravity(board, row)) : isFullline 함수에서 블록이 제거된 후 블록을 내려오게 하는 함수이다.

우선, 맨 위 줄을 0으로 초기화 한 후 지워지는 줄 부터 위로 올라가면서 한 칸씩 아래로 당겨온다.

for(int i=0; i<10; i++){

board[i][0]=0;

}

for(int i=0; i<10; i++){

for(int j=row; j>1; j--){

board[i][j]=board[i][j-1];

}

}

randomblock(gameblock,a) : seed 값에 따라 지정된 수에 따라 랜덤으로 블록 모양을 결정하는 함수이다.

ex)else if(a%7==3)

gameblock = new Sblock();

score class

멤버함수 :

socre() : socre(0),level(1),numblock(0)

getScore

getLevel

getNumblock :　접근자

upScore : 없어진 라인 수 대로 score+=2

setLevel : 50점당 레벨 1씩 증가한다 score/50+1

setNumblock : 없어진 block 수이다 score/2\*10

pane class

멤버변수 : 각각의 window의 폭과 높이를 지정하는 변수와 기본 위치를 지정하는 변수 윈도우변수

int width\_, height\_; int x\_, y\_; WINDOW \*win\_;

멤버함수 :

Pane(int x, int y, int w, int h); // 생성자

virtual ~Pane(); // 소멸자

virtual void draw(); // 각각의 게임 초기 화면을 그려주는 함수이다.

virtual void printscore(Score &sc){} // statpane에서 score 클래스 관련된 변수를 나나태는 함수이다,

virtual void printframe(int board[][20]){} // boardpane에서 board배열울 나타내는 함수이다.

virtual void drawinfo(char \*name){} // infopane에서 사용자로부터 입력받은 이름을 나타내는 함수이다.

virtual void printnextpane(Block &block){} //nextpane에서 다음 블록을 나타내는 함수이다.

virtual void printEnd(Score &sc){} // 게임이 끝났을 때 끝난 화면을 나타내는 함수이다.

virtual void printnum(Block &block){} // statpane에서 블록들이 얼마나 사용됬는지 나타내는 함수이다.

printframe(board) : board 배열을 돌면서 0이면 검은색을 출력하고 아니면 위에서 블록은 각각 다른 숫자들을 가지고 있으므로 각각의 숫자에 맞는 색을 출력하게 된다.

for(int i=0; i<10; i++){

for(int j=0; j<20; j++){

if(board[i][j]==1){

wattron(win\_,COLOR\_PAIR(5));

mvwprintw(win\_,j+1,2\*i+1,"■ ");

wattroff(win\_,COLOR\_PAIR(5));

}

else if(board[i][j]==2){

wattron(win\_,COLOR\_PAIR(2));

mvwprintw(win\_,j+1,2\*i+1,"■ ");

wattroff(win\_,COLOR\_PAIR(2));

}

}

}

main 함수

int main(int argc, char\* argv[]){

ifstream instream;

char \*name = new char[100];

unsigned int seed;

bool isgameover=true;

if(argc>1){ //파일 입력이 있으면 이름과 seed 값을 받는다

instream.open(argv[1]);

instream>>name>>seed;

}

else{ // 파일 입력이 없으면 사용자로부터 이릅과 seed 값을 시간에 따라 설정한다.

cout<<"Enter your name"<<endl;

cin>>name;

seed = (time(NULL));

}

Tetris t; // 게임을 진행할 tetris 객체 선언

Score sc; // score 객체 선언

int board[10][20]={0,}; //게임이 진행될 board배열 초기화

t.updateScreen(); //초기 게임화면 나타낸다.

t.updateinfopane(name); // 위에서 입력 받은 이름을 화면에 입력하는 함수이다.

int startkey = getch(); // 사용자로부터 키를 입력받는다.

if(startkey==10){ //enter키가 입력 되야 시작 된다

srand(seed);

Block \*gameblock; // 게임을 진행할 block 객체를 선언한다.

Block \*nextblock; // 다음 블록을 보여줄 block 객체를 선언한다.

int a = rand();

int b = rand();

t.randomblock(gameblock,a); // 난수 a에 의해 gameblock이 결정된다.

t.randomblock(nextblock,b); // 난수 b에 의해 nextblock이 결정된다

gameblock->createblock(board); // 결정된 gameblock을 board배열에 나타낸다.

t.updateboardpane(board); //board 배열에 나타낸 gameblock을 화면에 나타낸다

t.updatenextpane(\*nextblock); // 결정된 nextblock을 nextpane 화면에 나타낸다

t,updatestatpane(\*gameblock); // 결정된 gameblock의 개수를 statpane 화면에 나타낸다.

while(isgameover){

if(gameblock->Checkdown(board)==false){ //블럭이 내려갈 수 없을 때-> 블록이 놓이게 되는 경우다.

t.isFullline(board,sc);//줄이 다 차있는지 검사하는 함수를 실행한다.

gameblock = nextblock; // gameblock 주소를 nextblock 으로 설정한다.

a=rand();

t.randomblock(nextblock,a); // 새로 변경된 난 수 a를 통해 nextblock을 결정한다.

if(gameblock->isGameover(board)){ // 새로운 gameblock을 통해 Gameover 인지 검사한 후 각 pane을 갱신한

gameblock->createblock(board); 후 break 문을 통해 게임을 끝낸다.

t.updateboardpane(board);

t.updatestatpane(\*gameblock);

t.updatenextpane(\*nextblock);

break;

}

gameblock->createblock(board);//Gameover가 아니라면 새로운 gameblock을 board 배열에 나타낸다.

t.updatestatpane(\*gameblock); //결정된 gameblock의 개수를 statpane 화면에 나타낸다.

t.updatenextpane(\*nextblock); // 결정된 nextblock을 nextpane에 화면에 나타낸다

t.updateboardpane(board); //board 배열에 나타낸 gameblock을 화면에 나타낸다

}

if(argc>1){ //파일입력이 있을 경우이다.

char cur\_key;

instream>>cur\_key;

if(cur\_key=='q') //q가 입력 되면 break 문을 통해 게임을 끝내게 된다.

break;

int change=changefilekey(cur\_key); //입력된 문자를 정수로 치환하는 함수이다.

t.play(board,\*gameblock,change); // 치환된 함수로 블록을 움직이게 한다.

usleep(600000); //계속해서 함수 실행에 delay를 주면서 블록이 움직이는 경로를 보게 해준다.

}

else{ // 파일입력이 없을 경우 ->사용자로부터 키를 입력받는 경우이다.

while(1){ // 키 입력이 없을 경우 계속해서 내려가기 위해서 무한 루프를 만든다.

halfdelay(5); //0.5초에 한번씩 사용자로부터 키 입력이 있는지 검사하는 함수이다. 키 입력이 없다면 ERR을 반환한다.

int ch = getch(); // 사용자로부터 키 입력을 받는다.

if(ch==ERR){ // 키보드로부터 입력이 없다면

if(gameblock->Checkdown(board)) // 위와 같이 내려갈 수 있는지 검사한 후 내려간다.

gameblock->Movedown(board);

else

break; //내려갈 수 없다면 ->블럭이 놓이게 된다면 break를 통해 다시 무한 루프를 빠져나와 맨 위의 반복문으로 이동한다.

t.updateboardpane(board); // 내려가는동안 boardpane 화면에 나타낸다.

}

if(ch == 113){ // 사용자로부터 q입력을 받는 경우이다.

isgameover = false;

break; //break문을 통해 맨 위의 반복문으로 이동해서 게임이 끝나게 된다.

}

t.play(board,\*gameblock,ch); //방향키가 입력 되었으면 이동하게 된다.

}

}

}

cbreak(); //halfdelay 함수를 통해 변경된 지연모드를 다시 초기화 해준다.

t.printEnd(sc); // 마지막 화면인 점수를 표시한다.

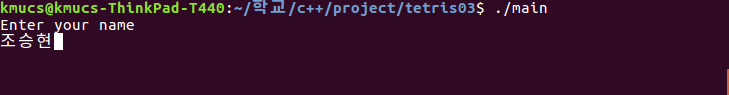
int outkey = getch(); // delay를 주기위해 활용했다.

}

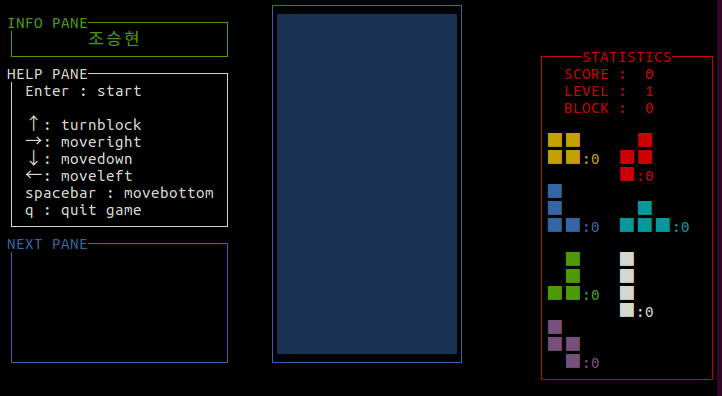
return 0;

}

1. 사용자로부터 이름 입력받는 모습



2. 게임 초기화면 모습



3. 게임이 진행되는 모습

4. 게임이 끝난 후의 모습

