전공 : 컴퓨터공학과 학년 : 4 학번 : 20212020 이름 : 박민준

1. 제공된 테트리스 소스파일을 미리 읽어보고, 테트리스 게임의 flow chart를 자세히 작성하시오.그리고 각 함수의 기능에 대해 설명하시오.

도표, 텍스트, 평면도, 라인이(가) 표시된 사진

자동 생성된 설명

- menu(): 메뉴를 출력하고 사용자의 입력을 받아 반환한다.

- GetCommand(): 사용자로부터 입력을 받아 해당 명령을 반환한다. 방향키나 스페이스바 입력을 처리한다.

- ProcessCommand(): 받은 명령을 처리하고, 필요한 경우 블록의 이동을 확인하고 그에 따른 필드의 상태를 업데이트한다.

- BlockDown(): 블록을 아래로 움직이는 함수이다. 블록을 아래로 움직일 수 있는지 확인하고, 가능하면 블록을 이동시킨다.

- CheckToMove(): 블록이 이동 가능한지 확인한다. 블록의 회전, 이동이 가능한지 확인한다.

- AddBlockToField(): 블록을 게임 필드에 추가한다.

- DeleteLine(): 꽉 찬 줄이 있는지 확인하고, 있다면 그 줄을 삭제하고 점수를 계산한다.

- DrawChange(): 블록의 이동에 따른 필드의 변화를 그린다.

- play(): 테트리스 게임의 주요 루틴이다. 사용자 입력을 받아 게임을 진행하고, 게임 오버 여부를 확인한다.

- DrawField(): 게임 필드를 그린다. 현재 필드의 상태를 기준으로 화면에 블록을 그린다.

- DrawNextBlock(): 다음 블록을 표시하는 영역에 다음 블록을 그린다.

- PrintScore(): 점수를 출력한다.

- DrawOutline(): 테트리스 화면의 기본 테두리를 그린다.

- DrawBox(): 주어진 위치와 크기로 상자를 그린다.

- DrawBlock(): 주어진 위치에 주어진 블록을 그린다.

2. 실습 시간에 구현할 5가지 함수들에 대한 간단한 pseudo code를 제시하시오.

int CheckToMove(char f[HEIGHT][WIDTH],int currentBlock,int blockRotate, int blockY, int blockX){

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) { // 각 블록 셀에 대하여 다음을 수행

if (현재 셀이 블록의 일부라면)

if (블록이 이동하려는 위치가 게임 영역 밖이거나 이미 채워진 셀이라면)

return 0;

}

}

return 1;

}

void DrawChange(char f[HEIGHT][WIDTH],int command,int currentBlock,int blockRotate, int blockY, int blockX){

switch (command) {

case KEY\_UP:

이전 블록 방향으로 회전한다.

case KEY\_DOWN:

이전 블록 위치의 아래로 이동한다.

case KEY\_RIGHT:

이전 블록 위치의 오른쪽으로 이동한다.

case KEY\_LEFT:

이전 블록 위치의 왼쪽으로 이동한다.

default:

break;

}

for(int i = 0; i < 4; i++){

for(int j = 0; j < 4; j++){ // 각 블록 셀에 대하여 다음을 수행

if (이전 블록이 위치하던 칸이라면) {

이전 블록의 좌표로 이동한다.

printw(".");

}

}

}

새로운 블록 정보를 그린다.

}

void BlockDown(int sig){

if (블록을 아래로 이동시킬 수 있다면) {

블록을 한 칸 아래로 이동시키고 새로운 위치에 블록을 그린다.

}

else {

if (blockY == -1) gameOver = 1;

}

현재 블록을 필드에 추가하고 필드에서 완전한 라인을 지우고, 점수를 계산한다.

새로운 블록을 다음 블록으로 이동시키고 게임 상태를 초기화한다.

}

void AddBlockToField(char f[HEIGHT][WIDTH],int currentBlock,int blockRotate, int blockY, int blockX){

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++)

if (현재 셀이 블록의 일부라면)

각 블록 셀에 대해, 필드에서 블록이 차지하는 셀에 1을 설정한다.

}

}

int DeleteLine(char f[HEIGHT][WIDTH]){

int flag, dLine = 0;

for(int i = 0; i < HEIGHT; i++) { // 모든 라인을 확인한다.

flag = 1;

for(int j = 0; j < WIDTH; j++){ // 라인이 꽉 차 있는지를 확인한다.

if (f[i][j] == 0) {

flag = 0;

break;

}

}

if (라인이 꽉 차 있다면) {

해당 라인을 지우고 상단의 모든 블록을 한 칸씩 아래로 이동시킨다.

dLine++;

i--;

}

}

return dLine \* dLine \* 100; // 삭제된 라인의 개수에 따라 점수를 계산하여 반환한다.

}