8주차 예비보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20231515 이름: 김다은

1-1. 테트리스 게임의 Flow Chart

**1-1-1. main() 함수의 Flow Chart**

텍스트, 도표, 그림, 기술 도면이(가) 표시된 사진

자동 생성된 설명

이때, InitTetris(), GetCommand(), ProcessCommand(), ProcessCommand()는 play 함수에서 호출되어 실행된다.

Menu 함수에서 1~4번 옵션 중에서 사용자가 Play를 선택한다면 -> InitTetris함수로 Tetris 초기화 -> GetCommand함수로 사용자가 누른 키 값 받아오기 ->ProcessCommand함수로 입력된 키값에 대한 과정 처리 -> ProcessCommand함수의 반환값이 Quit인 경우 -> 종료 후 Menu 함수를 실행한다.

**1-1-2. BlockDown함수의 Flow Chart**

텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

CheckToMove함수로 테트리스 블록이 한 칸 아래로 내려가는 것이 가능한지 판단한다.

-> 가능하면 블록을 한 칸 아래로 이동시킨다. / 불가능하다면 gameOver를 True로 설정한다.

-> field에 테트리스 블록을 나타낸다. -> 꽉 찬 줄이 있으면 삭제하고 점수에 더한다. -> nextBlock를 갱신한다. -> 현재 블록 정보들을 초기화한다.

1-2. 테트리스 게임에 사용되는 함수들 설명

**1-2-1. char menu(void);**

처음으로 게임을 실행했을 때 아래와 같은 메뉴 화면을 표시한다.

텍스트, 스크린샷, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

**1-2-2. void play(void);**

menu 함수의 return value가 MENU\_PLAY인 경우 실행된다. 테트리스 게임을 진행시킨다.

**1-2-3. void DrawField(int currentBlock, int blockRotate, int blockX, int blockY);**

field의 모습을 그린다. 블록이 없는 부분에는 ‘.’을 출력하고, 블록이 존재하는 부분에는 배경과 문자 색을 반전시켜 블록모양(배경색과 반전된 공백)을 출력한다.

**1-2-4. void DrawNextBlock(int \*nextBlock);**

다음 차례에 출력될 블록을 화면 오른쪽에 다음과 같이 표시한다.

텍스트, 스크린샷, 디자인, 폰트이(가) 표시된 사진

자동 생성된 설명

**1-2-5. void PrintScore(int score);**

현재 점수를 오른쪽 화면에 다음과 같이 출력한다.

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

**1-2-6. void DrawOutline(void);**

테트리스 게임이 이루어지는 공간의 테투리와 next block이 표시되는 부분의 테두리, score가 표시되는 부분의 테두리를 그리는 함수이다. DrawBox 함수를 호출하여 테두리를 그린다.

**1-2-7. int GetCommand(void);**

사용자의 키보드 입력값을 받아 return 한다.

**1-2-8. int ProcessCommand(int command);**

argument로 주어진 command 값에 따라 CheckToMove 함수를 사용하여 블록을 이동가능한지 판별한다. 이후 가능하다면 blockX 또는 blockY 값을 수정한다.

**1-2-9. void DrawBox(int y, int x, int height, int width);**

테두리 모양을 그려주는 함수이다.

**1-2-10. void DrawBlock(int y, int x, int blockID, int blockRotate, char tile);**

argument로 주어진 (y, x) 좌표에 해당 blockID와 blockRotate를 가지는 블록 모양을출력한다.

**1-2-11. int CheckToMove(char filed[HEIGHT][WIDTH], int currentBlock, int blockRotate, int blockY, int blockX);**

블록이 argument로 주어진 위치로 이동가능한지 여부를 반환한다.

**1-2-12. void DrawChange(char field[HEIGHT][WIDTH], int command, int currentBlock, int blockRotate, int blockY, int blockX);**

command의 값으로 알아낸 이전 블록의 모양을 찾아 지운다. 이후 새로운 블록을 그린다.

**1-2-13. void BlockDown(int sig);**

매초마다 프로그램 자체에서 호출되는 함수이다. 호출될 때마다 블록을 한 칸씩 아래로 이동시킨다.

**1-2-14. void AddBlockToField(char field[HEIGHT][WIDTH], int currentBlcok,int blockRotate, int blockY, int blockX);**

field에 추가된 블록의 위치 정보를 저장한다.

**1-2-15. int DeleteLine(char field[HEIGHT][WIDTH]);**

field를 탐색하여 column 전체가 꽉 찬 줄이 있는지 확인한다. 꽉 찬 구간이 있다면 해당 줄을 지우고 삭제한 줄 위의 정보들을 한 칸 아래로 이동시킨다.

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

**2-1. int CheckToMove(char filed[HEIGHT][WIDTH], int currentBlock, int blockRotate, int blockY, int blockX);**

for(i=0, j=0을 i<BLOCK\_HEIGHT, j<BLOCK\_WIDTH 일 때까지 i, j에 +1을 하면서 반복) {

블록의 위치가 게임판을 벗어나거나, 이미 그 위치에 다른 블록이 있다면 return 0;

아니라면 return 1;

}

**2-2. void DrawChange(char field[HEIGHT][WIDTH], int command, int currentBlock, int blockRotate, int blockY, int blockX);**

command의 값을 통해 이전 블록의 위치와 rotation 정보 알아냄

for(int i=0; i<BLOCK\_HEIGHT; i++) {

for(int j=0; j<BLOCK\_WIDTH; j++) {

if(이전 블록이 존재하는 위치이고 i+blockY가 0과 같거나 크다면) {

커서를 옮기고, 해당 위치의 그림 지우기

}

}

}

새로운 블록 정보를 그린다

**2-3. void BlockDown(int sig);**

if(블록이 아래로 움직일 수 있다면) {

blockY를 +1

아래로 움직인 블록을 그린다.

}

else {

if(blockY가 -1이라면) {

gameOver = 1;

}

블록을 field에 저장한다.

DeleteLine 함수를 실행하고 반환 값을 score에 더한다.

nextblock[1]을 nextblock[0]에 저장한다.

nextblock[1]에 0~6 사이의 랜덤 값을 저장한다.

blockRotate를 0으로 한다.

blockX, blockY를 초기화한다.

다음 차례의 블록, 점수를 화면에 그린다.

field를 그린다.

}

**2-4. void AddBlockToField(char field[HEIGHT][WIDTH], int currentBlcok,int blockRotate, int blockY, int blockX);**

for(int i=0; i<BLOCK\_HEIGHT; i++) {

        for(int j=0; j<BLOCK\_WIDTH; j++) {

if( (i, j) 좌표에 해당 블록이 존재하는 경우) {

(i, j) 좌표의 field 값을 1로 설정한다.

}

    }

}

**2-5. int DeleteLine(char field[HEIGHT][WIDTH]);**

int delete\_ypos[HEIGHT+1]; // 시간복잡도를 최소화하기 위해 추가 배열 사용

int delete\_cnt = 0; // 삭제한 줄의 수

int isfull; // 한 줄이 블록으로 꽉 찼는지 여부를 나타냄

for(int i=HEIGHT-1; i>=0; i--) {

        isfull = 1;

        for(int j=0; j<WIDTH; j++) {

            if(f의 (i, j) 좌표에 블록이 존재하지 않는 경우) {

                isfull = 0;

                break;

            }

        }

        if(isfull) { // 한 줄이 모두 채워진 경우

delete\_ypos[delete\_cnt]에 i 저장

            delete\_cnt 1 증가

        }

}

delete\_ypos[delete\_cnt] = 0;

// 시간 효율을 위해 꽉 찬 줄이 발견될 때마다 한 칸씩 아래로 이동시키는 것이 아니라,

// 지워야 할 줄을 delete\_ypos[]에 저장해 두었다가, 한 번에 아래로 이동시킨다.

for(int k=0; k<delete\_cnt; k++) {

        int target = delete\_ypos[k];

        for(int i=target-1; i>delete\_ypos[k+1]; i--) {

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

                f[i+k+1][j] = f[i][j];

            }

        }

        for(int j=0; j<WIDTH; j++) {

            f[0][j] = 0;

        }

}

return 삭제한 줄의 수\* 삭제한 줄의 수\*100