6주차 TETRIS1 결과 보고서

전공 : 수학 학년 : 4 학번 : 20151378 이름 : 진상우

1. 실습 시간에 작성한 프로그램의 함수들이 예비보고서에서 작성한 각 구현 함수들의 pseudo code와 어떻게 달라졌는지 설명하고, 시간 및 공간 복잡도를 보이시오.

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

<예비 보고서>

스크린샷이(가) 표시된 사진

자동 생성된 설명

<실습>

스크린샷이(가) 표시된 사진

자동 생성된 설명

이전 코드에서는 이중 for문에서 내부의 for문에 어떠한 경우라도 처음 반복문을 수행할 때 무조건 1이 return 되는 문제점이 있었다. 이를 수정하기 위해서 return 1을 맨마지막 괄호로 이동시켰고, 실제로 블록이 이동하기 전에 이동이 가능한지 확인하기 위하여 블록이 이미 쌓여있는지, 주어진 필드를 벗어나지 않는지를 수행하는 코드를 추가했다..

시간 복잡도 : O(BLOCKHEIGHT \* BLOCKWIDTH) 공간 복잡도 : O(1)

Void DrawChange(char field[HEIGHT][WIDTH], int command, int currentBlock, int blockRotate, int blockY, int blockX)

<예비보고서>스크린샷이(가) 표시된 사진

자동 생성된 설명

<실습>

스크린샷이(가) 표시된 사진

자동 생성된 설명

DrawChange 함수는 예비보고서와 많은 차이는 없는데, 이 함수는 블록을 좌우하단이동시키거나 회전시켰을 때, 기존에 그려진 블록을 지우고 새로운 부분으로 그려주는 함수이다. 다만, 블록이 정해진 크기의 정사각형 내에 그려진다는 점을 반영하여, 블록의 위치를 정확히 구해주기 위해서 이동하는 좌표를 pX와 pY로 받아 구체화하였다. 마지막으로는 필드에 출력하기 위해서 printf 함수가 아니라, printw 함수를 이용하였다.

시간복잡도 : O(BLOCKHEIGHT \* BLOCKWIDTH) 공간복잡도 : O(1)

Void BlockDown(int sig)

<예비보고서>스크린샷, 지도이(가) 표시된 사진

자동 생성된 설명

<실습>

스크린샷이(가) 표시된 사진

자동 생성된 설명

BlockDown 함수는 매초마다 블록을 한칸 아래로 내리는 역할을 하며, 예비보고서에 작성한 코드에서는 다음 블록이 막혀있는지를 확인함에 따라서 if문을 작성하였는데, 실습 때는 이전에 구현한 CheckToMove 함수를 사용하여 블록이 아래로 이동이 가능한지를 체크하였다. 블록이 아래로 그려질 수 있을때는 기존의 블록을 지우고 한칸 아래에 새로운 불록을 그리기 위해 DrawChange함수를 이용했다. 만약 블록이 아래로 내려갈 수 없을 때에는 현재 위치에 블록을 그려주고, 지워진 주는 만큼 점수를 추가해준다. 그리곤, 다음 블록으로 교체하기 위해 현재 블록을 다음 블록으로 교체시켜준다. 해당 블록의 기본정보들은 모두 초기화를 해주고, 이 블록의 다음 블록이 어떤 블록인지 오른쪽 창에 그려준다. 그리고 점수도 출력을 해주고 게임에 이용될 블록을 그려준다. 또한, 1초마다 블록이 한칸씩 내려올 수 있게 timed\_out 변수를 0으로 다시 설정해준다.

시간복잡도 : O(1) 공간복잡도 : O(1)

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

<예비 보고서>

스크린샷이(가) 표시된 사진

자동 생성된 설명

<실습>

스크린샷, 그리기이(가) 표시된 사진

자동 생성된 설명

AddBlockToField 함수는 예비보고서와 동일하다.

시간복잡도 : O(BLOCKHEIGHT\*BLOCKWIDTH) 공간복잡도 : O(1)

Int DeleteLine(char f[HEIGHT][WIDTH])

<예비 보고서>

스크린샷이(가) 표시된 사진

자동 생성된 설명

<실습>

스크린샷이(가) 표시된 사진

자동 생성된 설명

DeleteLine 함수는 꽉찬 줄이 있는 경우 삭제해주는 기능의 함수이다. 기존의 예비보고서에서 작성한 pseudo code는 단순히 각각의 줄에서 빈 공간이 있는지를 체크하고 빈공간이 없다면 줄을 삭제해주는 방식으로 작성하였다. 하지만 실습때 작성한 코드에서는 pseudo code에서 작성한 코드에, 줄을 지운 경우 그 줄의 윗부분에 해당하는 블록들을 한칸씩 아래로 내리는 과정을 추가하였다. 그리고 지운 줄만큼 점수를 추가해주기 위해서 지운줄^2 \* 100에 해당하는 값을 return 해주었다.

DeleteLine함수는 블럭을 필드에 추가하고 나서 만약 꽉찬 줄이 있다면 그 줄을 삭제해주는 함수로, 꽉 찬 줄이 있는지 체크하고, 줄을 삭제한뒤, 삭제한 줄 위의 모든 칸을 한 줄 내린다. 위에 코드에서는 꽉 찬 줄이 있는지를 체크하고, 줄을 삭제 하는 것에 이야기 하였지만, 삭제한 줄 위의 모든 칸을 한 줄 내리진 않았다. 따라서, 실습에서는 줄을 삭제하고, 삭제한 줄 위의 모든 칸을 한 줄 내리는 과정을 for문을 이용해 삭제는 하지 않고, 아랫줄에 덮어버려 동시에 처리해 주었다. 그리고 지운 줄 만큼 점수를 주기 위하여 count^2 \*100의 점수를 주었다.

시간복잡도 : O(HEIGHT \* WIDTH \* 삭제되는 줄 수) 공간복잡도 : O(1)

2. 테트리스 프로젝트 1주차 숙제 문제를 해결하기 위한 pseudo code를 기술하고, 시간 및 공간 복잡도를 보이시오.

[그림자 기능]

- DrawShadow 함수 수정(추가부분)

DrawBlock 함수는 문자를 parameter로 받아 블록을 그릴 때, 블록 안을 채워 그리는 기능을 한다.

Void DrawShadow(int y, int x, int blockID, int blockRotate){

While(CheckToMove 가 1){

y++;

If(CheckToMove == 0) break;

}

y--;

Call DrawBlock(‘/‘)

}

시간복잡도 : O(1) 공간복잡도 : O(1)

- DrawBlockWithFeatures 함수 추가

DrawBlock( ), DrawShadow( )함수를 이용하여, 움직임이 바뀔 때 마다 현재 블록과 그림자를 함께 그리도록 한다.

Void DrawBlockWithFeatures(int y, int x, int blockID, int blockRotate){

호출 DrawShadow;

호출 DrawBlock(‘ ‘)

}

시간 복잡도 : O(1) 공간 복잡도 : O(1)

[2개의 블록 미리 보여주기]

- Void InitTetris 수정

initTetris 에서는 nextBlock를 이용해서 두번째 nextBlock[2]를 초기화한다.

nextBlock[2] = rand( )%7; 을 nextBlock[1] = rand( )%7 밑에 추가하였다.

시간 복잡도 : O(HEIGHT\*WIDTH) 공간 복잡도 : O(HEIGHT\*WIDTH)

기존 함수와 동일

- void DrawNextBLock 수정

두번째 다음의 블럭을 그려주는데, if문 내부에 nextBlock[2]가 쓰인다.

For(i=0; i<4; i++){

Move(11+I, WIDTH+13)

For(j=0; j<4; j++){

If(block[nextBlock[2]][0][i][j] == 1){

Attron(A\_REVERSE); // with A\_REVERSE

Printw(“ “);

Attroff(A\_REVERSE);

}

Else printw(“ “);

}

}

위의 코드를 추가

시간 복잡도 : O(BLOCKHEIGHT \* BLOCKWIDTH) 공간 복잡도 : O(1)

기존 함수와 동일

- BlockDown 함수 수정

BlockDown 함수에서는 더 이상 아래로 움직일 수 없을 때, 블록 정보를 해준다. 블럭을 필드에 추가하고, 다음 블럭을 현재로 바꾸고, 두번째 다음 블럭을 새롭게 구하게 수정해준다.

nextBlock[0] = nextBlock[1] ; nextBlock[1] = rand() % 7; 를

-> nextBlock[0] = nextBlock[1] ; nextBlock[1] = nextBlock[2] ; nextBlock[2] = rand() % 7 로 교체한다.

시간복잡도 : O(1) 공간복잡도 : O(1)

[닿은 면적만큼 score 증가하기]

- void AddBlockToField 함수 수정

Ppt와는 다르게 블럭을 필드에 저장하였을 때 닿은 면적 \* 10 을 반환해야 하기에 int로 바꾸었다.

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

if(f[i+blockY+1][blockX+j] == 1) touched++;

else if(blockY+i+1 == HEIGHT) touched++;

Return 10 \* touched;

}

시간 복잡도 : O(BLOCKHEIGHT \* BLOCKWIDTH) 공간 복잡도 : O(1)

- void Blockdown 수정

Blockdown 함수에서는 AddBlockToField 함수로 계산되어 return된 score를 누적해야 하므로,

Score +=AddBlockToField(field, nextBlock[0], blockRotate, blockY, blockX) 코드를 추가해준다.

시간 복잡도 : O(1). 공간 복잡도 : O(1)