8주차 결과보고서

전공: 컴퓨터공학과 학년: 4학년 학번: 20212022 이름: 이예준

**1.**

**1. int CheckToMove()**

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

2중 for문을 사용했고, block이 4x4 그리드안에 존재하기 때문에 for문의 범위를 2중 모두

0~4까지로 제한했다. for문을 돌면서 해당 그리드 안에 block이 존재하는 경우 block이

field를 벗어났는지 또는 다른 block과 겹쳤는지 확인을 해서 해당되면 0을 반환하고 아니하면

1을 반환한다.

시간 복잡도 : 언제나 일정하게 42번 반복문을 돌기 때문에 상수 시간 복잡도 O(1)이 된다.

공간 복잡도 : 함수 내에서 추가로 할당되는 공간이 없기 때문에 공간 복잡도는 O(1)이 된다.

**2. void DrawChange()**

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

먼저 현재 block의 회전정보, 위치를 임시로 저장한다. 그 다음 switch문을 이용하여

이전 block의 정보를 가져온다. command값에 따라서 ‘↑’키이면 이전 회전 정보를,

‘↓’키이면 한 칸 위 높이 정보를, ‘→’키이면 한 칸 왼쪽 위치 정보를 그리고 ‘←’키이면

한 칸 오른쪽 위치 정보를 불러와서 저장한다.

2중 for문을 이용하여 이전 block을 field에서 제거하고, DrawBlock 함수로 현재 block을 그린다

마지막으로 move함수를 이용하여 커서 위치를 초기화 시켜준다.

시간 복잡도 : 언제나 일정하게 42번 반복문을 돌기 때문에 상수 시간 복잡도 O(1)이 된다.

공간 복잡도 : 함수 내에서 추가로 할당되는 공간이 없기 때문에 공간 복잡도는 O(1)이 된다.

**3. void BlockDown()**

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

CheckToMove 함수를 통해 만약 현재 block이 한 칸 아래로 이동할 수 있다면,

blockY 변수 값에 1을 더한 뒤 DrawChange 함수로 다시 그린다.

만약 이동할 수 없다고 했을 때, blockY값이 -1이라면, 게임을 더 진행시킬 수 없기 때문에

gameOver 변수 값을 1로 설정하여 게임을 종료 시키도록 한다.

AddBlockToField 함수로 field에 현재 block을 추가시키고, DeleteLine 함수를 통해 만약 field에

block이 한 줄로 나열되어 있다면 그 줄을 제거하고 줄을 제거한 점수를 반환한다.

현재 block을 nextblock으로 교체하고 nextblock은 랜덤으로 설정하고,

현재 block의 회전, 높이, 좌우 위치를 초기화 시켜준다.

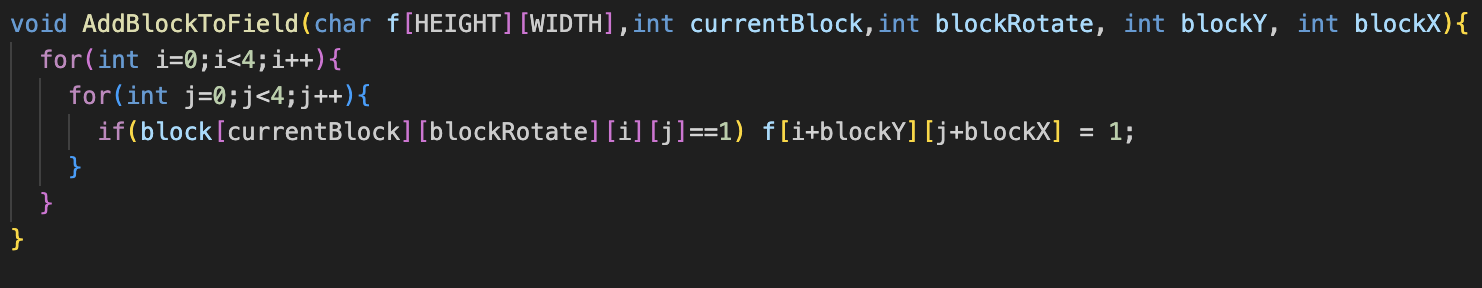
DrawNextBlock(), PrintScore(), DrawField(), Drawblock() 함수를 이용해서

각각 다음 block, 점수, 필드, 현재 block을 그린다.

시간 복잡도 : if문으로만 이루어져 있기 때문에 상수 시간 복잡도 O(1)이 된다.

공간 복잡도 : 추가적인 메모리를 사용하지 않기 때문에 공간 복잡도는 O(1)이 된다

**4. void AddBlockToField()**



2중 for문을 이용해서 block을 field 2차원 배열에 추가한다.

시간 복잡도 : 언제나 일정하게 42번 반복문을 돌기 때문에 상수 시간 복잡도 O(1)이 된다.

공간 복잡도 : 함수 내에서 추가로 할당되는 공간이 없기 때문에 공간 복잡도는 O(1)이 된다.

**5. int DeleteLine()**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

2중 for문을 이용해서 처음에 flag를 1로 초기화해준 뒤 field의 한 줄을 돌면서

비어 있는 곳이 있으면 flag를 0으로 설정해준다. 모두 돌고 나서 flag값이 여전히 1로

유지 되어 있다면 한 줄이 모두 차 있다는 것을 의미하므로 다시 2중 for문을 이용해서

그 줄을 제거하고 윗줄에 있는 모든 block을 한 칸 씩 내린다.

마지막에는 (제거한 줄의 개수)2\*100을 반환한다.

시간 복잡도 : 4중 for문을 이용했으며 field의 크기가 정해져 있지 않기 때문에

시간 복잡도는 O(n4)이다.

공간 복잡도 : 추가적인 메모리를 사용하지 않기 때문에 공간 복잡도는 O(1)이 된다

**2.**

**2-1. 그림자 기능**

**1.DrawShadow()**

while( 더 이상 내려갈 수 없을 때까지 ){

높이를 한 칸 내리기 ;

}

높이를 한 칸 올리기 ;

DrawBlock() ;

//시간 복잡도는 O(n)이며, 공간 복잡도는 O(1)이다.

**2.DrawBlockWithFeatures()**

DrawBlock() ;

DrawShadow() ;

//시간 복잡도는 O(1)이며, 공간 복잡도는 O(1)이다.

**3.DrawChange()**

DrawField() ; //추가

그림자는 field 배열에 추가하지 않았기 때문에 DrawField() 함수를 이용해서

이전에 그린 그림자를 삭제한다.

DrawBlockWithFeatures() ; //변경

DrawBlock() 함수가 있는 위치에는 DrawBlockWithFeatures() 함수로 교체하여

현재 block과 shadow를 그린다.

// 시간 복잡도는 O(1)이며, 공간 복잡도는 O(1)이다.

**2-2. 2개의 블록 미리 보여주기**

**1.InitTetris()**

nextBlock[2]=rand%7 ; //추가

2번째 next block을 생성한다.

// 시간 복잡도는 O(n2)이며, 공간 복잡도는 O(1)이다.

**2.DrawOutline()**

DrawBox(9,WIDTH+10,4,8); //추가

2번째 next block을 출력하기 위한 Line을 생성한다.

move(15,WIDTH+10); //변경

DrawBox(16,WIDTH+10,1,8); //변경

점수와 점수를 출력하기 위한 line도 그에 맞춰서 아래로 이동시켜 출력 및 그린다.

// 시간 복잡도는 O(1)이며, 공간 복잡도는 O(1)이다.

**3.PrintScore()**

move(17,WIDTH+11) ; //변경

점수를 나타내는 line이 이동했기 때문에 점수를 출력하는 위치도 바꿔준다.

// 시간 복잡도는 O(1)이며, 공간 복잡도는 O(1)이다.

**4.DrawNextBlock()**

for(0 ~ 4){

move(10+i,WIDTH+13);

for(0 ~ 4){

블럭 그리기 ;

}

}

원래 있던 2중 반복문을 복사한 뒤 출력하는 위치만 바꿔준다.

// 시간 복잡도는 O(1)이며, 공간 복잡도는 O(1)이다.

**5.BlockDown()**

nextBlock[1]=nextBlock[2]; //변경

nextBlock[2]=rand()%7; //추가

next block은 2번째 next block으로 바꿔주고, 2번째 next block은 랜덤으로 생성한다.

// 시간 복잡도는 O(1)이며, 공간 복잡도는 O(1)이다.

**2-3. 닿은 면적만큼 Score 증가하기**

**1.AddBlockToField()**

for(i = 0~4){

for(j = 0~4){

if(block[currentBlock][blockRotate][i][j]==1){

if(i+blockY+1 == HEIGHT || f[i+blockY+1][j+blockX]) touched++ ; //추가

}

}

}

만약 block이 field의 맨 밑까지 내려왔을 경우 맞닿는 부분에 대해 touched 값을 1 추가한다.

// 시간 복잡도는 O(1)이며, 공간 복잡도는 O(1)이다.

**2.BlockDown()**

score += AddBlockToField(field, nextBlock[0], blockRotate, blockY, blockX) ;

AddBlockToField(0 함수에서 반환되는 점수를 그대로 score 변수에 추가한다.

// 시간 복잡도는 O(1)이며, 공간 복잡도는 O(1)이다.