컴퓨터공학실험1 6주차 결과보고서

전공: 컴퓨터공학과 학년: 2 학번: 20191559 이름: 강상원

1. 실습 시간에 작성한 프로그램의 함수들이 예비보고서에서 작성한 각 구현 함수들의 pseudo code와 어떻게 달라졌는지 설명하고, 각 함수에 대한 시간 및 공간 복잡도를 보이시오(각 함수의 시간 및 공간복잡도를 구할 때, 어떤 변수에 의존하는지를 판단해야 한다).

Pseudo code와 실제 구현 코드를 비교해 보자면 일부 중첩 if 문을 && 연산자를 써서 표현한 점, x, y의 변수 이름이 blockX, blockY로 달라진 점 외에는 큰 차이는 존재하지 않는다.

(1) CheckToMove: 나올 수 있는 블럭의 배열 크기가 4*4이고, 각 행, 열마다 연산을 수행하므로 시간 복잡도는 O(1)이다. (O(WIDTH*HEIGHT)), 공간복잡도: O(1)

```
CheckToMove(f, currentBlock, blockRotate, blockY, blockX)
    for i = 0 to 3
        for j = 0 to 3
            if block[currentBlock][blockRotate][i][ j] == 1
                x = blockX+ j
                y = blockY + i
                if (0 ≤ x < WIDTH && 0 ≤ y < HEIGHT) ≠ TRUE
                    return TRUE
                if f[y][x]==1
                    return FALSE
    return TRUE
```

```
int CheckToMove(char f[HEIGHT][WIDTH],int currentBlock,int blockRotate, int
blockY, int blockX){
    // user code
    for(int i=0; i<4; i++){
        for(int j=0; j<4; j++){
            if(block[currentBlock][blockRotate][i][j]){
                int xpos=blockX+j;
                int ypos=blockY+i;

                if(f[ypos][xpos] || (xpos>=WIDTH) || (xpos<0) || (ypos>=HEIGHT)
|| (ypos<0))
                    return 0;
            }
        }
    }
    return 1;
}
```

(2) DrawChange: Command에 의해 변경될 부분만 칸을 변경한다. DrawBlock에 O(1)만큼의 시간 복잡도가 걸린다. 공간복잡도는 O(1)이다. Pseudo code와 달라진 부분은 switch문 안의 각 경우의 수에 대하여 변수를 변화시킨 뒤 Delete_Block함수를 호출하는 부분이다.

```
DrawChange(f, command, currentBlock, blockRotate, blockY, blockX)
    if (command == KEY_UP) blockRotate = (blockRotate+3)%4
    if (command == KEY_DOWN) blockY =- 1
    if (command == KEY_LEFT) blockX += 1
    if (command == KEY_RIGHT) blockX -= 1
    //ERASE BLOCK by Calling Delete_Block function
    //Draw new block
    DrawBlock(blockY, blockX, currentBlock, blockRotate, ' ')
```

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

    //1. 이전 블록 정보를 찾는다. ProcessCommand 의 switch 문을 참조할 것
    //2. 이전 블록 정보를 지운다. DrawBlock 함수 참조할 것.

    switch(command) {
        case KEY_UP:
            Delete_Block(blockY, blockX, currentBlock, (blockRotate+3)%4);
            break;
        case KEY_DOWN:
            Delete_Block(blockY-1, blockX, currentBlock, blockRotate); break;
        case KEY_LEFT:
            Delete_Block(blockY, blockX+1, currentBlock, blockRotate); break;
        case KEY_RIGHT:
            Delete_Block(blockY, blockX-1, currentBlock, blockRotate); break;
        default:
            break;

    }

    //3. 새로운 블록 정보를 그린다.
    DrawBlock(blockY, blockX, currentBlock, blockRotate, ' ');
}
```

(3) BlockDown: CheckToMove의 시간복잡도 O(1), DrawChange의 시간복잡도 O(HEIGHT*WIDTH), DrawField의 시간복잡도 O(HEIGHT*WIDTH)이므로 총 시간복잡도는 O(HEIGHT*WIDTH)이다. 공간복잡도도 O(1)이다. blockY=blocky+1을 blockY++로 바꾸는 등 사소한 것 이외에는 차이가 없다.

```
BlockDown(sig)
    if CheckToMove(f, currentBlock, blockRotate, blockY +1, blockX)
            blockY = blockY + 1
        DrawChange(f, command, currentBlock, blockRotate, blockY, blockX)
```

```
        else
            if blockY == -1
                gameOver = TRUE
            else
                score += DeleteLine(field)
                for i = 0 to 1
                    nextBlock[i] = nextBlock[i+1]
                nextBlock[2] = (Random integer in 0 ··· 6)
                DrawNextBlock(nextBlock)
                blockY = -1
                blockX = WIDTH / 2 - 2
                blockRotate = 0
                PrintScore(score)
        DrawField()
    timed_out = TRUE
```

```c
void BlockDown(int sig){
    // user code
    if(CheckToMove(field, nextBlock[0], blockRotate, blockY+1, blockX)){
        blockY++;
        DrawChange(field, KEY_DOWN, nextBlock[0], blockRotate, blockY, blockX);
    }
    else{
        AddBlockToField(field, nextBlock[0], blockRotate, blockY, blockX);
        if (blockY== -1) gameOver=1;
        else{
            score+= DeleteLine(field);
            for(int i=0; i<2; i++)
                nextBlock[i]=nextBlock[i+1];
            nextBlock[2]=rand()%7;
            DrawNextBlock(nextBlock);

            blockY= -1;
            blockX= WIDTH/2 -2;
            blockRotate=0;
            PrintScore(score);
        }
        DrawField();
    }
    //강의자료 p26-27 의 플로우차트를 참고한다.
    timed_out=0;
}
```

(4) AddBlockToField: 올 수 있는 블럭의 크기 행, 열 각각 4개씩 O(1) 작업이 진행되므로 시간 복잡도는 O(1)이다. 공간 복잡도는 O(1)이다.

```
AddBlockToField(f, currentBlock, blockRotate, blockY, blockX)
  for i = 0 to 3
      for j = 0 to 3
        if block[currentBlock][blockRotate][i][j]
                    if (0 ≤ i+blockY < HEIGHT && 0 ≤ j+blockX < WIDTH)
```

f[i+blockY][j+blockX] = 1

return TRUE

```
void AddBlockToField(char f[HEIGHT][WIDTH],int currentBlock,int blockRotate,
int blockY, int blockX){
    // user code
    for(int i=0; i<4; i++){
        for(int j=0; j<4; j++){
            if (block[currentBlock][blockRotate][i][j]){
                if (0<=i+blockY && i+blockY<HEIGHT && 0<=j+blockX &&
j+blockX<WIDTH)
                    f[i+blockY][j+blockX]=1;
            }

        }
    }
    //Block 이 추가된 영역의 필드값을 바꾼다.
}
```

(5) DeleteLine: 각 열마다 한 줄이 꽉 차있는지 확인하므로 시간 복잡도는 O(HEIGHT*WIDTH)이다. 공간 복잡도는 O(1)이다.

DeleteLine(f)
er_line_cnt = 0
all_fit = FALSE
for i = 0 to HEIGHT - 1
    all_fit = TRUE
    for j = 0 to WIDTH − 1
        if f[i][j] == 0
            all_fit = FALSE
    if all_fit == TRUE
        er_line_cnt = er_line_cnt + 1
        for y = i downto 1
            for x = 0 to WIDTH − 1
                f[y][x] = f[y−1][x]
        for x = 0 to WIDTH - 1
            f[0][x] = 0
        i = i - 1
return er_line_cnt * er_line_cnt * 100

```
int DeleteLine(char f[HEIGHT][WIDTH]){
    // user code

    //1. 필드를 탐색하여, 꽉 찬 구간이 있는지 탐색한다.
    int er_line_cnt=0;
    bool all_fit=0;

    for(int i=0; i<HEIGHT; i++){
        all_fit=1;
        for(int j=0; j<WIDTH; j++){
            if(!f[i][j]) all_fit=0;
        }
```

```
        if(all_fit){
            er_line_cnt++;
            for(int ypos=i; ypos>=1; ypos--){
                for(int xpos=0; xpos<WIDTH; xpos++)
                    f[ypos][xpos]=f[ypos-1][xpos];
            }
            for(int xpos=0; xpos<WIDTH; xpos++) f[0][xpos]=0;
            i--;
        }
    }
    //2. 꽉 찬 구간이 있으면 해당 구간을 지운다. 즉, 해당 구간으로 필드값을 한칸씩 내린다.

    return er_line_cnt*er_line_cnt*100;
}
```

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

(1) DrawBlock

```
DrawBlock(y, x, blockID, blockID, blockRotate, tile('/') )
for i=0 to 3
      for j = 0 to 3
            if block[blockID][blockRotate][i][ j] == 1 && i + y  ≥  0
                  move to (i+y+1, j+x+1)
                  print tiles
Move cursor to (HEIGHT , WIDTH + 10)
```

행과 열 각 4칸씩에 대해서 연산, 출력을 진행한다. 시간 복잡도는 O(1)이고, 공간 복잡도는 O(1)이다.

(2) FinalShapeDrop

```
FinalShapeDrop(y, x, blockID, blockRotate)
      while CheckToMove(field, nextBlock[0], blockRotate, y+1, x)
            y++
return y
```

블록이 최대한 내려갈 수 있는 길이를 측정한다. while문 안에서 CheckToMove 함수로 내릴 수 있을 때까지 값을 증가시킨다. 최악의 경우를 따져 보았을 때, 시간 복잡도는 O(HEIGHT)이다. (끝까지 내릴 수 있으므로) 공간복잡도는 O(1)이다.

(3) DrawShadow

```
DrawShadow(y, x, blockID, blockRotate)
    Ylength = FinalShapeDrop(y, x, blockID, blockRotate)
    DrawBlock(Ylength, x, blockID, blockRotate, '/')
```

'/'를 이용해 그림자를 그린다. 시간 복잡도는 O(HEIGHT), 공간 복잡도는 O(1)이다.

(4) DrawBlockWithFeatures

```
DrawBlockWithFeatures(y, x, blockID, blockRotate)
    DrawShadow(y, x, blockID, blockRotate)
    DrawBlock(y, x, blockID, blockRotate, ' ')
```

DrawShadow 함수를 호출하여 그림자를 그리고, DrawBlock 함수를 호출하여 블럭을 그린다. 시간 복잡도 O(HEIGHT), 공간 복잡도 O(1)이다.

- 기타 수정 사항

(1) InitTetris

nextBlock[0] = rand() % 7;

nextBlock[1] = rand() % 7; 로 바꾸고

DrawBlock(blockY, blockX, nextBlock[0], blockRotate, ' ');

DrawShadow(blockY, blockX, nextBlock[0], blockRotate);를

DrawBlockWithFeatures(blockY, blockX, nextBlock[0], blockRotate); 한 줄로 바꾸었다.

(2) DrawOutline

DrawBox(9, WIDTH + 10, 4, 8); 를 추가하여 두 번째 블럭의 표시 위치를 만들었다.

(3) AddBlockToField

return cnt * 10 을 통해 바닥면과 닿아서 발생한 점수를 반환한다.