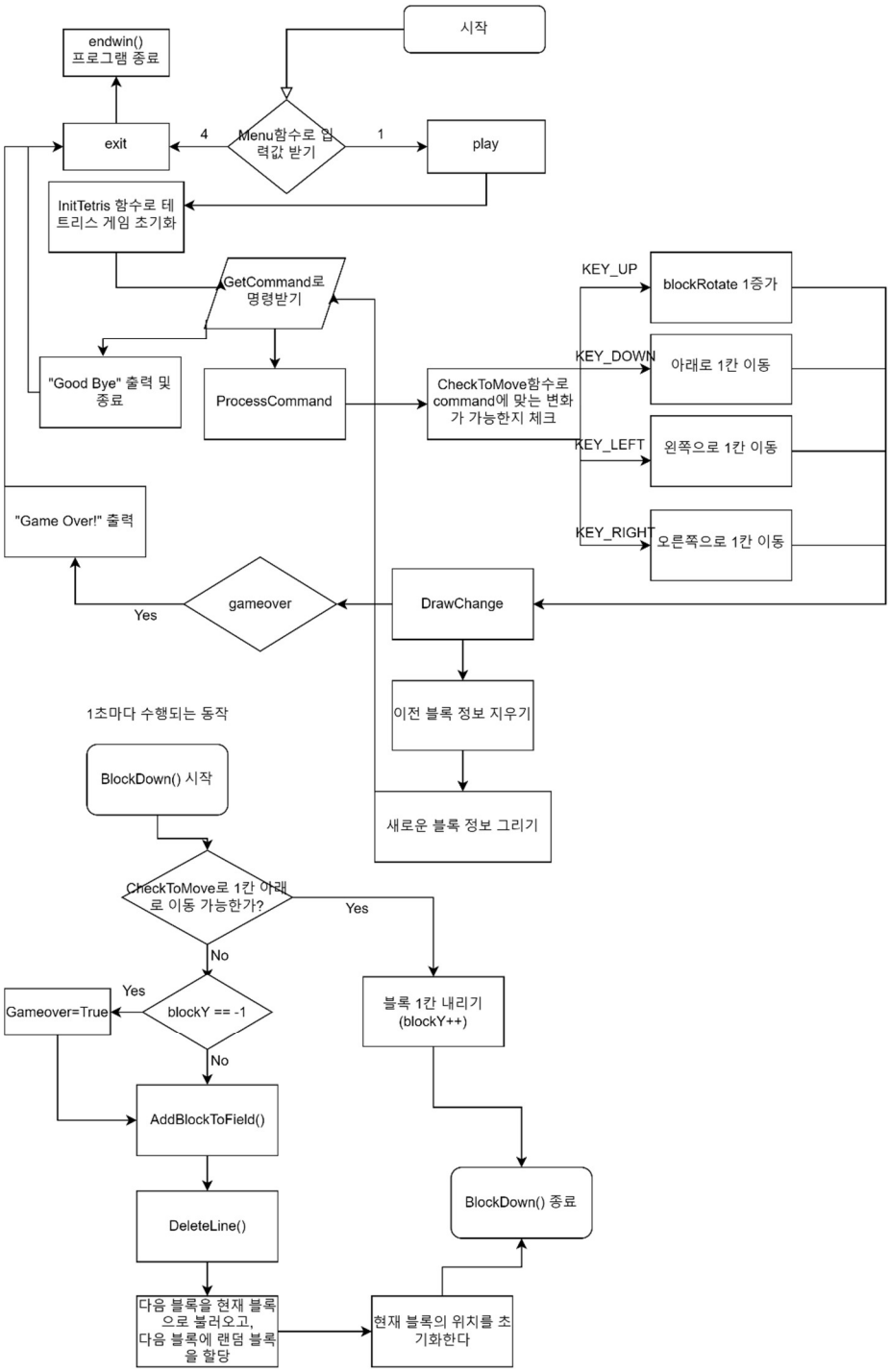
테트리스 1주차 예비보고서

전공: 철학과 학년: 3학년 학번: 20180032 이름: 남기동

1. 테트리스 게임의 flowchart



2. 테트리스 게임을 구성하는 각 함수에 대한 설명

1) void InitTetris()

이 함수는 테트리스를 플레이할 때 필요한 기초 작업을 해주는 역할을 하는 함수이다. 일반 global 변수들을 초기화해주는데, 현재 블록(nextBlock[0])에 무작위로 블록을 설정하고 다음 블록(nextBlock[1])에도 무작위로 블록을 할당한다. 블록의 회전수는 0으로 초기화하고 y좌표는 -1로 필드 보다 위를 의미한다. x좌표는 width/2-2로 너비의 중간을 나타내는데 -2는 블록이 4x4의 크기를 갖기 때문에 2칸 전부터 그려야 블록이 중간에 그려지기 때문에 위와 같이 설정된 것이다. 점수를 할당하는 score도 0으로 초기화하고 gameover라는 변수는 0으로 초기화되는데 후에 1이 되면 gameover를 화면에 띄우고 게임이 종료되게 하는 것을 나타내 주는 변수이다.

변수를 초기화한 이후에는 drawoutline함수를 통해 테두리를 그리고 drawfield를 통해 필드(테투리 안에서 블록이 위치하는 공간)를 그려주며 drawblock함수를 통해서는 현재의 블록을 직접 그리고 drawnextblock을 통해서는 다음 블록을 그려준다. Printscore는 현재 점수가 얼마인지를 해당하는 공간에 가서 출력해준다.

2) void DrawOutline()

이 함수는 테두리를 그리는 함수이다. 보다 구체적으로는 이후에 설명할 Drawbox함수를 통해서 3개의 공간의 테두리를 그리는 역할을 한다. 첫 번째는 블록이 떨어지는 공간의 테두리이며, 두 번째는 next block을 보여주는 공간의 테두리, 그리고 마지막으로 점수를 보여주는 공간의 테두리를 그리는 역할을 수행한다.

3) int Getcommand()

이 함수는 키보드를 통해 입력된 값을 command로 받아들이기 위한 함수이다. wgetch(stdscr)을 이용하여 상하좌우 화살표, 그리고 q나 Q를 command로 받게 된다. 그 외의 값에 대해서는 Nothing(0)으로 command를 설정하고 설정한 command를 반환하고 함수를 종료한다.

4) int ProcessCommand(int command)

이 함수는 위의 Getcommand함수를 통해 반환된 command 값을 읽고 그에 해당하는 작업을 수

행하는 함수이다. q나 Q가 들어온 경우에는 Quit을 반환하고 이는 후에 play함수에서 gameover를 출력하게 하는 조건으로 이용된다. 이외의 화살표 입력들에 대해서는 CheckToMove함수를 통해 그러한 입력이 요구하는 행동을 할 수 있는지 검사한 이후에 가능하다면 DrawChange함수를 통해 변화를 적용하고 그렇지 않다면 변화를 적용하지 않는 방향으로 작동한다. 화살표 아래, 오른쪽, 왼쪽은 모두 블록의 이동에 관한 것이라 특별한 것은 없는데 화살표 위는 블록의 방향을 바꾸는 것을 수행하는 역할이다. blockrotate값을 1증가시키는데, 1마다 90도의 변화를 의미한다. 회전이 가능하다면 blockrotate값을 1 증가시키고 그것을 drawchange함수를 통해 적용하게 된다.

5) void DrawField()

이 함수는 블록이 움직이고 쌓이는 공간인 field를 그리는 함수이다. HEIGHT와 WIDTH 값에 맞추어 커서를 이동하면서 attorn(A_REVERSE) → 출력 내용 → attroff(A_REVERSE)를 통해 공간이 차있다면 빈칸에 색을 반전하여 색깔이 칠해진 칸이 보이도록 출력을 하고 그렇지 않으면 .을 찍어서 빈칸임을 보여준다. 이때 빈칸과 채워진 칸의 구분은 field[][]의 값이 0인가 혹은 1인가로 구분된다

6) void Printscore(int score)

이 함수는 score를 위해 initTetris함수가 호출한 drawoutline함수에 의해서 그려진 테두리 안으로 이동하여 할당된 score의 값을 출력하는 역할을 수행한다

7) void DrawNextBlock(int *nextBlock)

이 함수는 drawoutline에서 다음 블록을 위해 만들어둔 테두리 안으로 이동하여 Drawfield함수와 유사하게 attorn(A_REVERSE), attroff(A_REVERSE)를 이용하여 4x4 공간 안에 블록이 있는 자리는 색칠된 빈칸으로, 그렇지 않고 빈 공간은 공백으로 그린다. Drawfield와 다른 점은 빈 공간을 . 으로 표시하는게 아니라 공백으로 그린다는 점이다

8) void DrawBlock(int y, int x, int blockID, int blockRotate, char tile)

이 함수는 블록을 그리는 함수이다. 최대 4x4 공간 안에 정의된 블록의 id와 회전수를 받아오고 4x4공간을 한 칸씩 조사하면서 채워진 공간이면 tile에 색을 채워서 출력한다

9) void DrawBox(int y, int x, int height, int width)

이 함수는 DrawOutline에서 사용되는 함수로 주어진 인자에 맞추어 실질적으로 박스를 만드는 함수이다. 앞의 두 개의 인자인 y, x는 박스가 만들어지기 시작하는 좌측 상단의 공간을 의미한다. 따라서 주어진 위치로 이동하여 3번째 인자인 height와 4번째 인자인 width를 기준으로 크기에 맞는 박스를 그리게 된다. 박스를 그리는 순서는 일단 좌상단 모서리를 addch(ACS_ULCORNER)로 그리고 너비만큼 addch(ACS_HLINE)으로 박스의 상단을 그린후 좌측과 우측라인은 addch(ACS_VLINE)으로 동시에 그리면서 내려간다. 마지막으로 addch(ACS_LLCORNER), addch(ACS_HLINE), addch(ACS_LRCORNER)를 통해 하단의 모서리와 하단을 그려서 박스를 완성한다.

10) void play()

이 함수는 main함수에서 menu함수를 통해 1번 값을 입력받았을 때 호출하는 함수이다. play함수가 호출되면 InitTetris함수를 통해 테트리스를 위한 초기 값을 설정한 이후 getcommand를 통해 입력된 command를 확인하고 processcommand를 통해 해당하는 command에 적절한 행동을 취한다. 하지만 이 때 입력된 command가 q,Q라면 QUIT이 processcommand의 반환값이 될 것이고 이 경우에는 Good-bye라는 것을 화면의 중간에 출력하고 refresh를 통해 윈도우 상에 바뀐 내용을 반영한다. 이 작동을 gameover이 1이 되기 전까지 반복적으로 수행하게 된다. 그리고 gameover이 1이 되어 게임이 끝나게 되면 Gameover을 출력하고 똑같이 refresh를 통해 윈도우 상에 바뀐 내용을 반영하고 newRank함수를 통해 사용자의 이름을 입력받고 score와 함께 저장하게 된다.

11) char menu()

이 함수는 play, rank, recommended play exit이라는 4가지 옵션을 printw를 통해 윈도우 화면에 출력해주고 wgetch(stdscr)을 통해 어떤 값이 입력되었는지를 반환하는 역할을 수행한다.

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

이 함수는 ProcessCommand함수에서 키보드 화살표 입력을 받았을 때 각각의 입력에 따른 명령을 수행할 수 있는지 수행 전에 검사해보기 위한 역할을 수행하는 함수이다. 인수 중 f[][]는 테트리스 필드의 정보를 받아온 것이고 currentBlock은 현재 블록의 ID, blockRotate는 블록의 회

전수, blockY는 현재 블록의 y좌표, blockX는 현재 블록의 x좌표를 의미한다. ProcessCommand함수를 통해 서로 다른 입력들이 들어올텐데 들어온 입력에 대해서 좌우,아래 이동이라면 이동한 곳이 비어있는 공간인지를 체크하여 이동 가능 여부를 판단하고 화살표 위가 입력되면 rotate를 수행하기 위해 변형된 블록의 정보가 비어있는 공간에 위치할 수 있는지 판단하는 작업을 수행하게 될 것이다.

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

이 함수는 이전 블록 정보를 찾고 이전 블록의 정보를 지운 이후에 새로운 블록의 정보를 그리는 역할을 수행하는 함수이다. checktomove함수에서 command가 인자로 추가된 상태라는 점이 차이점이라고 할 수 있다.

14) void BlockDown(int sig)

블록이 한칸 내려갈 수 있는지 checktomove함수를 통해 확인하고 내려갈 수 있으면 drawchange함수를 통해 블록을 아래로 한 칸 내리고 함수를 종료한다. 만약 한 칸 내려갈 수 없는 상황이라면 다음 작업을 수행하는데 블록의 y좌표가 -1인 경우와 그렇지 않은 경우에 다른 작업을 수행하게 된다. 만약 y좌표가 -1인 경우에는 블록이 시작하는 지점이라는 것을 의미하며 이는 이미 블록이 꼭대기까지 쌓였음을 의미한다. 따라서 게임이 종료될 수 있도록 flag를 True로 설정하여 blockdown함수가 끝난 이후 프로그램이 종료될 수 있도록 한다. 만약 그렇지 않다면 블록을 필드에 쌓는 AddBlockToField함수를 호출한다. 그 이후에 완전히 꽉 채워진 줄이 있다면 지워주고 점수를 갱신하고 출력해야 한다. 이 과정에는 DeleteLIne과 PrintScore함수가 사용된다. 그 다음에는 다음 시행을 위해 NextBlock[1]을 nextblock[0]에 대입하여 다음 블록을 현재 블록에 넣고 rand()%7을 통해 다시 nextblock[1]에 새로운 블록을 무작위 생성 후 할당한다. 마지막으로는 drawField함수로 현재 블록의 위치를 초기화하고 함수를 종료한다.

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

이 함수는 Block이 추가된 영역의 필드값을 바꾸는 역할을 하는 함수이다. 위의 blockdown함수를 통해 이미 block이 내려갈 수 없고 field에 부딪히는 상황에서 호출되기 때문에 그 당시의

block의 정보, 회전수, x,y좌표를 가지고 field에 추가해주는 작업을 수행하면 된다.

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

필드를 처음부터 끝까지 이중 루프(height와 width만큼)를 통해 조사하면서 각각의 height마다 width의 모든 값에 해당하는 field의 값이 1로 채워져있다면 그 때 해당 height의 모든 필드 값을 0으로 바꾸어 지우고 그 위의 필드에 해당하는 필드값들을 한칸씩 아래로 내린다. 또한 이 때 지운 line의 개수에 대한 점수를 score 리턴하고 이것은 score 함수에 할당될 것이다.

3. 실습시간에 구현할 5가지 함수들에 대한 간단한 pseudo code 작성

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

```
for i=0 to HEIGHT-1
        for j=0 to WIDTH -1
                if ( block[currentblock][blockRotate][i][j] == 1)
                        if( i+blockY >= HEIGHT || j+blockX >= WIDTH )
                        return 0 //이동 불가능
                        if ( f[i+blocky][j+blockX] == 1 )
                        return 0
                else return 1 //이동 가능
```

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

```
Switch ( command )
For i=0 to HEIGHT
For j =0 to WIDTH
If( Block[previousBlock][blockRotate][i][j] == 1 )
        F[i][j] = 0
```

DrawBlock ( )

Move(HEIGHT, WIDTH + 10)

3) void BlockDown(int sig)

Checkflag = 0

If (checkflag = CheckToMove(field, nextBlock[0], blockRotate, blockY+1, blockX)) blockY++

If(blockY == -1) gameover =1;

Else

AddBlockToField( )

DeleteLine( )

nextBlock[0] = nextBlock[1]

nextBlock[1] = rand()%7

blockRotate = 0

blockY = -1

blockX = WIDTH /2 -2

DrawNextBlock( )

PrintScore ( )

DrawField ( )

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

for i=0 to HEIGHT

for j=0 to WIDTH

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

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

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

```
Int count
for i =0 to HEIGHT
        deleteflag=1
        for j=0 to WIDTH
                if (f[i][j] == 0) deleteflag =0
        if(deleteflag == 1) //삭제될 줄이 있다면
                count++
                for j=0 to WIDTH f[i][j] = 0
                for i to 0
                        for j=0 to WIDTH
                                f[i][j] = f[i-1][j] //위의 줄을 아랫줄로 내린다
return ( pow(count, 2) * 100 )
```