**8주차 예비보고서**

전공 : 수학전공 학년 : 3학년 학번 : 20191244 이름 : 김현승

1. **Flow chart**

구현할 테트리스 프로그램은 크게 두가지의 독립적인 실행을 통해 프로그램이 동작한다. 하나는 사용자의 키 입력을 받고 처리하는 역할을 하는 play(), 다른 하나는 1초마다 호출되어 필드 내 떨어지고 있는 블록의 위치를 갱신하는 BlockDown() 함수이다. play() 함수에서 alarm(1); 함수 호출을 통해 1초마다 시그널을 보내고, 이 때 시그널 핸들러로 설정한 BlockDown() 함수가 실행되게 되는 구조이다. 먼저 main() 함수에서부터 play() 함수까지의 Flow chart를 보자.

텍스트, 스크린샷, 도표, 디자인이(가) 표시된 사진

자동 생성된 설명

main() 함수가 실행되고 여러 초기 설정 이후 menu() 함수를 호출해 사용자 입력을 받는다. 이 때 1을 입력한 경우 play() 함수가 실행되고, 4를 입력한 경우 프로그램이 종료된다. play() 함수 내부에서는 시그널 핸들링과 같은 처리를 수행한 후 InitTetris() 함수로 초기 설정을 진행한다. 이후 do-while loop에 진입해 게임의 메인 루프가 돌아가게 된다. 이 루프 안에 alarm() 함수를 배치함으로써 1초마다 BlockDown() 함수가 실행되게 되는 것이다. 이와 별개로 do-while 루프에서 GetCommand() 함수를 통해 사용자 입력을 받고 입력에 따라 지정된 command를 리턴해 ProcessCommand() 함수에서 해당 command를 처리하게 된다. 이 때, 처리된 값이 QUIT라면 프로그램을 종료하는 방식으로 play() 함수가 실행된다.

다음은 1초마다 호출되어 필드 내 내려가는 블록의 위치를 갱신하는 BlockDown() 함수의 flow chart를 살펴보자.

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

BlockDown() 함수가 실행되면 먼저 CheckToMove() 함수를 호출해 현재 블록이 한 칸 내려갈 수 있는지 확인한다. 만약 내려갈 수 있다면 블록의 y좌표를 1 증가시키고 그대로 종료한다. 내려갈 수 없다면 추가적으로 블록의 y좌표를 확인해야한다. 만약 블록의 y좌표가 -1이면 더 이상 화면 상에 쌓을 곳이 없다는 뜻이므로 gameOver 플래그를 True로 설정해 play() 함수의 do-while 루프를 탈출하도록 한다. -1이 아닌 경우에는 AddBlockToField() 함수를 호출해 필드에 블록을 쌓고 DeleteLine() 함수를 호출해 모두 채워진 라인이 있으면 제거하고 점수를 갱신한다. 다음 블록으로 설정되어있는 블록을 현재 블록으로 바꾸고, 다음 블록을 rand() 함수를 사용해 랜덤하게 결정한다. 마지막으로, 바뀐 현재 블록에 대한 위치 정보를 초기화 하고 BlockDown() 함수를 종료한다.

1. **Functions**

* char menu(void);

화면에 메뉴를 출력하고, wgetch(stdscr)을 호출해 사용자로부터 표준 입력을 받아 입력 값을 리턴한다.

* void play(void);

테트리스 게임의 메인 게임 루프를 실행하는 함수이다. 시그널 핸들링과 alarm(1) 을 통해 1초마다 시그널을 생성해 BlockDown() 함수를 호출하고, GetCommand()와 ProcessCommand()를 통해 사용자의 입력을 처리한다.

* void DrawField(int currentBlock, int blockRotate, int blockX, int blockY);

테트리스 게임 플레이 필드를 그린다. 정해진 HEIGHT와 WIDTH 만큼의 필드를 그린다.

* void DrawnextBlock(int \*nextBlock);

다음 블록으로 설정된 블록을 오른쪽 next block 화면에 그린다. tetris.h에 선언 및 초기화된 block 배열을 가져와 그린다. 이 때, rotate num은 0으로 그리게 된다. (회전하지 않은 초기 상태)

* void PrintScore(int score);

화면의 정해진 위치에 score를 출력한다.

* void DrawOutline(void);

화면의 테두리를 그린다. 필드 화면, 다음 블록을 보여주는 화면, 점수판의 테두리를 DrawBox() 함수를 사용해 그린다.

* void DrawBox(int y, int x, int height, int width);

(y, x) 위치를 왼쪽 위 모서리로 하고 높이가 height, 너비가 width인 직사각형을 그린다.

* int GetCommand(void);

플레이어의 입력을 대기하고 해당 입력에 대한 command를 리턴한다. 상/하/좌/우 방향키 값, QUIT, NOTHING가 command 값으로 리턴된다. 리턴된 값은 ProcessCommand() 함수에서 처리된다.

* int ProcessCommand(int command);

GetCommand() 에서 입력 받은 command에 대한 처리를 수행한다. 받은 command가 QUIT인 경우 QUIT를 리턴, 방향키인 경우 CheckToMove() 함수를 이용해 회전이나 이동이 가능한지 확인하고 회전/이동을 수행한다. 회전/이동이 수행된 경우 drawFlag가 1로 설정되어 DrawChange() 함수를 통해 이전 상태의 블록을 지우고 새로운 상태로 갱신한다.

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

(y, x) 위치에 blockID의 모양을 갖고 blockRotate의 회전수를 갖는 블록을 화면에 tile로 채워 그린다.

* int CheckToMove(char field[HEIGHT][WIDTH], int currentBlock, int blockRotate, int blockY, int blockX);

currentBlock이 움직일 수 있는지 확인한다. 만약 현재 블록이 한 칸 내려갈 수 있는지 확인하려면 CheckToMove(field, nextBlock[0], blockRotate, blockY+1, blockX) 를 호출한다. 움직일 수 있는 경우 1을, 없는 경우 0을 리턴한다.

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

인자로 받은 command를 통해 블록의 이전 상태를 찾아 이전 블록을 지운다. 그리고 변경된(인자로 받은 블록 상태들) 블록을 새로 화면에 그린다.

* void BlockDown(int sig);

1초마다 호출되어 블록을 한 칸 내린다. CheckToMove() 함수를 이용해 블록을 내릴 수 있는지 확인하고, 내릴 수 있다면 blockY를 1 증가시킨다. 만약 내릴 수 없다면 blockY가 -1인지 확인해 필드에 블록이 쌓일 수 있는 공간이 있는지 확인한다. 공간이 없다면 패배한 것이므로 gameOver 플래그를 True로 설정하고, 공간이 있다면 AddBlockToField() 함수로 블록을 필드에 쌓는다. 이후 다음 블록에 대한 처리를 진행하고, 현재 블록의 위치를 초기화한다.

* void AddBlockToField(char field[HEIGHT][WIDTH], int currentBlock, int blockRotate, int blockY, int blockX);

인자로 받은 상태의 블록을 field에 추가한다. 해당 블록이 채워진 곳의 field 값을 1로 설정한다.

* int DeleteLine(char field[HEIGHT][WIDTH]);

필드에 가로로 모두 채워진 줄이 있는지 확인하고, 있다면 해당 구간을 지우고 위의 필드 값을 한 칸씩 내리고 점수를 갱신한다.

1. **Pseudo code**

* CheckToMove()

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

**loop for** int i : 0 ~ 3

**loof for** int j : 0 ~ 3

**if** block[currentBlock][blockRotate][i][j] == 1

**if** i+blockY is out of field or j+blockX is out of field

**return** 0

**if** field[i+blockY][j+blockX] == 1

**return** 0

**return** 1

* DrawChange()

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

**switch** command

**case** KEY\_UP

oldRotate = (blockRotate-1)%4

oldY, X = blockY, X

**case** KEY\_DOWN

oldRotate, X = blockRotate, X

oldY = blockY-1

**case** KEY\_RIGHT

oldRotate, Y = blockRotate, Y

oldX = blockX-1

**case** KEY\_LEFT

oldRotate, Y = blockRotate, Y

oldX = blockX+1

**loop for** int i is 0 to 3

**loop for** int j is 0 to 3

**if** block[currentBlock][oldRotate][i][j] == 1

move(i+oldY, j+oldX)

printw(“.”)

DrawBlock(blockY, blockX, currentBlock, blockRotate, tile)

* BlockDown()

void BlockDown(int sig)

**if** CheckToMove(field, nextBlock[0], blockRotate, blockY, blockX)

blockY++

**return**

**if** blockY == -1

gameOver = 1

AddBlockToField(field, nextBlock[0], blockRotate, blockY, blockX)

DeleteLine(field)

nextBlock[0] = nextBlock[1]

nextBlock[1] = rand() % 7

Initialize blockRotate, blockY, blockX

DrawNextBlock(nextBlock)

PrintScore(score)

DrawField()

* AddBlockToField()

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

**loop for** int i is 0 to 3

**loop for** int j is 0 to 3

**if** block[currentBlock][blockRotate][i][j] == 1

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

* DeleteLine()

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

deletedLine = 0

**loop for** int i is 0 to HEIGHT-1

flag = **True**

**loop for** int j is 0 to WIDTH-1

**if** f[i][j] == 0

flag = **False**

break

**if** flag is **True**

deletedLine++

pull down f[0~i-1] to f[1~i]

**return** (deletedLine)^2 \* 100