8주차 예비보고서

전공: 기계공학과 학년: 3학년 학번: 20191820 이름: 김형준

**1.**

**1-1. 함수들의 기능**

int CheckToMove(char filed[HEIGHT][WIDTH], int currentBlock, int blockRotate, int blockY, int blockX) : 블록이 (blockY,blockX)로 움직일 수 있는지의 여부를 확인함

(가능하면 1을 리턴, 불가능하면 0을 리턴함)

void DrawChange(char field[HEIGHT][WIDTH], int command, int currentBlock, int blockRotate, int blockY, int blockX) : 블록의 이전 위치를 지우고, 명령(command)에 의해서 바뀐 위치에 블록을 다시 그린다.

void BlockDown(int sig) : 블록을 한 칸 내릴 수 있으면 내린다. 내릴 수 없다면 블록을 필드에 쌓고 Next 슬롯에 있는 다음 블록을 사용한다. (쌓아서 필드의 줄을 삭제한 경우,

줄을 삭제해서 얻은 점수가 현재 점수에 추가된다.)

void AddBlockToField(char field[HEIGHT][WIDTH], int currentBlcok,int blockRotate, int blockY, int blockX) : (blockY,blockX)에 블록을 필드에 추가시킨다. (해당하는 field 배열의 원소에 1을 대입함)

int DeleteLine(char field[HEIGHT][WIDTH]) : 필드 전체를 검사해서 꽉 찬 줄이 있는 경우 해당 줄을 지우고, 밑으로 한 칸씩 내린다. (지운 줄의 수)^2\*100를 리턴한다.

char menu(void) : 화면에 메뉴를 출력하고 입력 값을 받아서 리턴한다.

void play(void) : 테트리스 게임을 시작한다.

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

: 테트리스 게임의 필드를 그린다. 필드에 블록이 채워져 있는 칸은 색이 반전된 ‘ ‘(Space)로, 빈 칸은 ‘.’로 채운다.

void DrawNextBlock(int \*nextBlock) : 화면 우측의 Next 슬롯에 다음에 나올 블록을 그린다. (nextBlock[1]의 모양을 표시함)

void PrintScore(int score) : 점수를 표시한다.

void DrawOutline(void) : DrawBox()를 사용해 테트리스 필드, Next 슬롯, 점수 표기판의 테두리를 그린다.

int GetCommand(void) : 사용자의 입력 값을 받아서 해당하는 명령으로 리턴한다. (상하좌우 방향키, 스페이스바, Q키만 전달되고 나머지 키들은 NOTHING으로 변환되어 리턴됨, Q키의 경우 QUIT으로 변환되어 리턴됨)

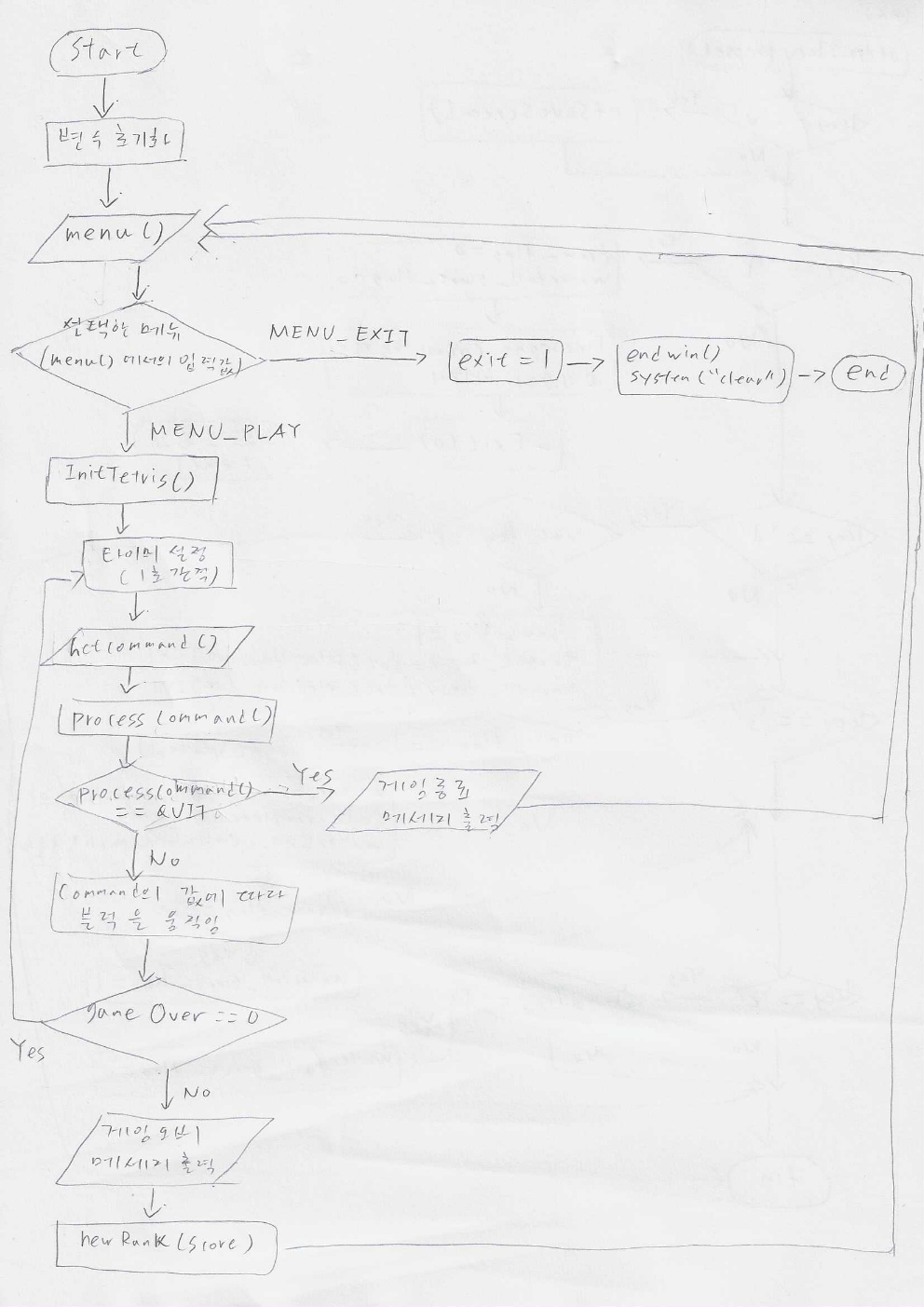
int ProcessCommand(int command) : 입력받은 명령(command)를 처리한다.

command가 QUIT일 경우 QUIT을 리턴하고, 다른 경우에는 1을 리턴한다.

void DrawBox(int y, int x, int height, int width) : (y,x)에 내부 공간이 width\*height 크기인 직사각형 테두리를 그린다. (직사각형의 내부는 채우지 않음)

void DrawBlock(int y, int x, int blockID, int blockRotate, char tile) : (y,x)에 tile 문자로 블록을 그린다.

**1-2. flow chart**



**2.**

**CheckToMove(f, currentBlock, blockRotate, blockY, blockX)**

for i = 0 to BLOCK\_HEIGHT-1

for j = 0 to BLOCK\_WIDTH-1

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

if (blockY+i < 0 || blockY+i >= HEIGHT || blockX+j < 0 || blockX+j >= WIDTH)

return 0

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

return 0

return 1

**DrawChange(f, command, currentBlock, blockRotate, blockY, blockX)**

prevRotate=blockRotate, prevblockX=blockX, prevblockY=blockY

switch(command)

case KEY\_UP:

prevRotate = (blockRotate>0) ? blockRotate-1 : NUM\_OF\_ROTATE-1

break

case KEY\_DOWN:

prevblockY = blockY-1

break

case KEY\_RIGHT:

prevblockX = blockX-1

break

case KEY\_LEFT:

prevblockX = blockX+1

break

for i = 0 to BLOCK\_HEIGHT-1

for j = 0 to BLOCK\_WIDTH-1

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

move(prevblockY+i+1,prevblockX+j+1)

printw(".")

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

move(HEIGHT,WIDTH+10)

**BlockDown(sig)**

if (CheckToMove(field,nextBlock[0],blockRotate,blockY+1,blockX))

blockY++

DrawChange(field,KEY\_DOWN,nextBlock[0],blockRotate,blockY,blockX)

else

if (blockY == -1) gameOver = 1

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

score += DeleteLine(field)

nextBlock[0] = nextBlock[1]

nextBlock[1] = rand()%7

blockRotate=0

blockY=-1

blockX=WIDTH/2-2

DrawNextBlock(nextBlock)

PrintScore(score)

DrawField()

timed\_out = 0

**AddBlockToField(f, currentBlock, blockRotate, blockY, blockX)**

for i = 0 to BLOCK\_HEIGHT-1

for j = 0 to BLOCK\_WIDTH-1

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

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

**DeleteLine(f)**

del = 0, i = HEIGHT

for count = 0 to HEIGHT-1

check = 1

for j = 0 to WIDTH-1

if (f[i][j] == 0) check = 0

if (check == 1)

for line = i-1 to 0

for x = 0 to WIDTH-1

f[line+1][x] = f[line][x]

for x = 0 to WIDTH-1

f[0][x] = 0

del++

else

i--

return del\*del\*100