**6주차 Tetris-1**

전공: 수학과 학년: 4 학번: 20161255 이름: 장원태

1. **테트리스 frame 프로그램 파일을 미리 읽어보고(부록1 ncurses 라이브러리 포함), 테트리스 게임의 flow chart를 자세히 작성하시오. 그리고 테트리스 게임을 구성하는 각 함수의 기능에 대해서 설명하시오.**

테트리스 게임의 flow chart는 다음과 같다.

End Blockdown()

Start Blockdown()

Start main

yes

yes

blockY == -1

CheckToMove()

menu()

no

no

Exit == 1

MENU\_PLAY

Gameover

== TRUE

blockY++

GetCommand()

InitTetris()

DeleteLine()

AddBlockToField()

Initialize

Location of currentBlock

nextBlock[0] = nextBlock[1]

nextBlock[1] = rand()%7

ProcessCommand()

ProcessCommand() == Quit

no

yes

End main

Main 함수는 게임을 시작하기 전 정보를 초기화하고, 입력받은 값으로 동작을 수행하는 함수이다. 그리고 BlockDown 함수는 게임 내에서 일정 시간마다 함수를 호출하여, 블록이 아래로 점점 떨어지도록 하는 함수이다.

테트리스 게임을 구성하는 함수들의 기능은 다음과 같다.

* InitTetris() : 테트리스 필드를 초기화하고, 현재 블록과 다음 블록의 모든 정보, 점수, 게임 종료 여부 모두 초기화한다. 그리고, timed\_out 변수도 0으로 초기화한다. 이렇게 결정된 정보를 바탕으로, 여러 함수 호출을 통해 초기 화면을 그린다.
* DrawOutline() : 테트리스 프로그램의 배경을 그리는 함수로, 테트리스 필드의 테두리와, 다음 블록이 미리 그려질 상자의 테두리, 그리고 점수를 나타내는 상자의 테두리를 그린다.
* GetCommand() : 테트리스 게임 중에 사용자의 입력을 받는 함수이다. 네 개의 방향키와 ‘q’(혹은 ‘Q’) 입력을 받아 명령을 리턴한다.
* ProcessCommand(int command) : GetCommand 함수로부터 입력받은 명령을 수행한다. 게임이 강제 종료되는 경우에만 QUIT을 반환하며, 그 외에는 1을 반환한다. 각 방향키에 대해서 그 방향키를 수행할 수 있는지 CheckToMove 함수를 이용하여 체크한다.
* DrawField() : 블록을 쌓을 테트리스 필드를 그리는 함수로, 그 정보는 0과 1로 구분된다. 0은 필드가 비어있음을 나타내어 ‘.’으로 표시된다. 1은 필드가 채워져 있음을 나타내어, 공백을 사용하여 화면에 표시한다.
* PrintScore(int score) : 점수를 표시하는 상자에 score값을 출력한다.
* DrawNextBlock(int \*nextBlock): nextBlock 정보를 바탕으로, 게임 화면 내의 다음 블록을 표시하는 상자에 해당 블록을 그린다.
* DrawBlock(int y, int x, int blockID, int blockRotate, char tile): 블록의 좌표, 모양, 회전수에 대한 정보를 이용해서 정해진 위치에 블록을 그린다. 이 때, 블록은 tile 모양으로 채워진다.
* DrawBox(int y, int x, int height, int width): 입력받은 위치에 입력받은 크기의 상자를 그린다.
* Play(): 테트리스 게임에 필요한 전반적인 과정을 관리한다.
* Menu(): 사용자가 입력한 메뉴 번호를 리턴한다. 1이 play, 2가 rank, 3이 recommended play, 4가 exit이다.
* CheckToMove(char field[HEIGHT][WIDTH], int currentBlock, int blockRotate, int blockY, int blockX): 현재 블록의 정보를 입력받아, 이동할 방향으로 이동할 수 있는지 판단한다. 이동할 수 없으면 0을, 이동할 수 있으면 1을 리턴한다.
* DrawChange(char field[HEIGHT][WIDTH], int command, int currentBlock, int blockRotate, int blockY, int blockX): 변경된 정보를 반영하여, 현재 블록이 이동한 블록을 다시 그린다.
* BlockDown(int sig): 현재 블록을 한 칸씩 아래로 떨어트린다. 블록이 더 이상 내려갈 수 없다면, 블록을 쌓고, 완전히 채워진 줄을 지우며, 그 후 다음 블록을 가져오며 점수를 화면에 출력해준다. 더 이상 블록을 넣을 공간이 없으면 게임 오버가 된다.
* AddBlockToField(char field[HEIGHT][WIDTH], int currentBlock, int blockRotate, int blockY, int blockX): 필드 내의 해당 위치에 블록을 쌓아준다.
* DeleteLine(char field[HEIGHT][WIDTH]): 완전히 채워진 라인을 찾아 지우고, 그만큼 점수를 계산하여 리턴한다.

1. **실습시간에 구현할 5가지 함수들에 대한 간단한 pseudo code를 제시하시오.**

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

for (i=0;i<BLOCK\_HEIGHT; i++){

for (j=0;j<BLOCK\_WIDTH;j++){

if (block[currentBlock][blockRotate][i][j] == 1) return 0;

else return 1;

}

}

}

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

switch(command){

case KEY\_UP:

반시계 방향으로 회전;

break;

case KEY\_DOWN:

y--;

break;

case KEY\_LEFT:

x++;

break;

case KEY\_RIGHT:

x--;

break;

}

for (i=0;i<BLOCK\_HEIGHT;i++){

for (j=0;j<BLOCK\_WIDTH;j++){

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

move(y, x);

printf(“.”);

}

}

}

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

}

void BlockDown(int sig){

if (블록을 내릴 수 있음){

blockY++

DrawChange(field, KEY\_DOWN, 다음 블록, blockRotate, blockY, blockX)

}

else{

if (blockY == -1) gameOver=1;

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

}

}

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

for (i=0;i<BLOCK\_HEIGHT;i++){

for (j=0;j<BLOCK\_WIDTH;j++){

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

field[blockY+i][blockX+j] = 1;

}

}

}

Int DeleteLine(char field[HEIGHT][WIDTH]){

flag = -1;

for (i=1;i<HEIGHT;i++){

for (j=0;j<WIDTH;j++){

if (field[i][j] == 0){flag=0; break;}

}

}

if (flag == 1){

온전히 채워진 라인 삭제;

점수 계산;

return 점수;

}

}