6주차 예비보고서

전공: 수학/컴퓨터공학 학년: 3학년 학번: 20181288 이름: 윤성호

1. 테트리스 게임은 play 함수와 blockDown함수가 독립적으로 실행되면서 진행된다. Play함수는 키를 입력받아 해당하는 이동명령을 수행한다. 이 때 키 입력에 대한 동작은 입력 처리, 입력 대기 동작의 반복이다. 그리고 blockDown함수는 정해준 시간간격으로 명령이 자동으로 수행되도록 하는 함수이다. 이 때 자동으로 수행해야 하는 2가지 명령은 매초마다 블럭이 y축방향으로 +1만큼 이동이 가능하면 이동시키는 명령, 필드에서 한 줄이 완전히 채워졌을 때 그 줄을 삭제하고. Score에 점수를 반영해주는 명령이다. 두 함수가 독립적으로 동작하므로, 방향키를 계속 누르면서 블럭을 이동시키면서도(play함수) 1초가 지날때마다 블럭이 아래방향으로 1칸 떨어지게 된다.(blockDown함수)

Main함수의 flow chart는 아래와 같다.

Main() start

Menu() ---------------------------------

|

InitTetris() |

|

---------------> GetCommand() |

| |

| ProcessCommand() |

| |

-----no------ProcessCommand의 값 == QUIT |

Yes |

Main() end <--------------------------------

테트리스 project는 main함수로부터 시작된다. Main함수에서는 menu함수가 실행되며, 1,2,3,4를 입력하면 각각의 해당하는 게임을 시작하게 된다. 1주차 실습에서는 아직 1 : play 와 4 : exit 메뉴에만 들어갈 수 있도록 구현되어있다.

4를 누르면 main함수 실행을 종료하게 되고 다시 터미널로 돌아오게 된다.

1을 누르면 실제 테트리스 게임이 시작된다. 가장 먼저 InitTetris함수가 실행되는데, 이 함수는 테트리스의 모든 값들을 기본값으로 초기화해주는 함수이다. 먼저, 게임 진행에 필요한 global변수들을 기본값으로 초기화해준다. 여기서 초기화 해주는 global변수는 아래와 같다.

* nextBlock : 현재 블록과 다음 블록의 정보를 저장하고 있는 크기 2인 배열.

Rand()%7 를 통해 0부터 6사이의 임의의 수를 저장한다.

* blockRotate : 블록의 회전 횟수. 0으로 초기화한다.
* blockX : 4\*4 블록 배열의 (0,0) 위치의 x좌표. WIDTH/2 – 2 로 초기화한다.
* blockY : 4\*4 블록 배열의 (0,0) 위치의 y좌표, 0으로 초기화한다.
* score : 점수. 0으로 초기화한다.
* gameOver : 게임 종료 여부를 저장하는 변수. 0으로 초기화한다.
* timed\_out : blockDown함수를 매초마다 진행시키기 위한 변수. 0으로 초기화한다.

변수들을 초기화하고 난 후에는, 테트리스 게임 화면을 초기화해야한다. 초기화과정에서 이용되는 함수들은 DrawOutLine(), DrawField(), DrawBlock(blockY, blockX, blockID, blockRotate, ‘ ‘), DrawNextBlock(nextBlock), PrintScore(score) 이다. 자세한 함수의 설명 아래에서 설명할 예정이다.

키 입력은 GetCommand()함수를 통해 받는다. Command 값은 방향키와 스페이스바, 그리고 ‘q’, ‘Q’ 를 입력으로 받고 다음 키 입력을 받게되면 command 값을 NOTHING으로 설정해준다. 그리고 ProcessCommand(command) 함수를 통해 함수의 입력 값에 따라 할 행동을 정해준다. 방향키가 입력되었다면, 이동가능 여부를 CheckToMove함수를 통해 판단하여, 이동이 가능하면 블럭이 그 방향으로 이동된 것을 DrawChange()함수를 통해 그려준다. 그리고 다시 GetCommand()함수가 실행되는 위치로 돌아가 계속해서 키 입력을 받을 준비를 한다. 만약 QUIT이 입력되었다면 게임을 종료하라는 의미이므로, 필드 가운데 Good-bye!! 라는 문구를 출력해주며 더 이상의 입력을 받지 않고 게임을 종료한다.

위는 입력을 통한 이동 과정이었고, 이 과정과 동시에 blockDown함수 또한 매초 실행이 된다. blockDown 함수의 flow chart는 아래와 같다.

BlockDown start

---------------------------------

CheckToMove |

no |

--------------- blockY == -1 |

| yes |

gameOver = TRUE no |

|

--------------> AddBlockToField blocky++

|

DeleteLine() |

|

currentBlock = nextBlock |

nextBlock = rand() |

|

Initialize Location of currentBlock |

|

BlockDown end <---------------------------

가장 먼저, ChechToMove 함수로 아래방향으로 한 칸 내려갈 수 있는 지 여부를 판단한다. 만약 내려갈 수 있다면 blockY++를 해주고 blockDown 함수를 종료시키고 다음 초를 기다린다. 내려갈 수 없는 경우에는 두가지 경우로 나뉜다. 첫번째 경우는 blockY == -1인 경우이다. 이는 가장 최상단 행까지 필드가 블록으로 찼다는 것을 의미하고 게임을 종료해야 하는 시점임을 알 수 있다. 따라서 gameOver 변수를 1로 바꿔주고 계속 진행한다. blockY != 1인 경우에는 블럭이 성공적으로 쌓일 수 있는 상황임을 알려주고, 쌓기 위한 과정을 진행한다. DrawChange로 이동한 블럭을 화면에 그려주고, 그려져 있는 정보를 AddBlockToField 함수로 field 배열에 옮겨준다. 그리고 DeleteLine 함수를 불러와 완전히 채워진 줄이 존재한다면 ( \* 100 만큼의 점수를 리턴해준다. 이 리턴값을 global 변수 score에 반영해준 다음, PrintScore함수를 이용해 score함수를 화면에 원하는 형식으로 표시해준다. 그리고 그다음 블럭을 현재 블록으로 설정해주고, 그다음 블럭은 rand()%7을 7가지 모양 중 임의로 모양을 지정해준다. 그리고 DrawNextBlock을 통해 다음 블럭을 원하는 형식으로 화면에 표시해주고, DrawField 함수로 AddBlockToField를 통해 field로 받아온 화면의 내용을 다시 가져와 화면에 표시해준다. 그리고 다음 블럭의 x좌표, y좌표, 회전횟수를 처음과 같이 초기화 해주고, 커서 위치를 화면 밖에로 이동시켜주면 blockDown 함수의 동작은 마무리된다.

아래는 테트리스 게임을 구성하는 함수들이다.

* Void InitTetris() : 테트리스 게임을 진행하기 위해 기본적인 변수들에 대한 초기화를 진행헤주는 함수. DrawOutLine(), DrawField(), DrawBlock(), DrawNextBlock(), PrintScore() 함수를 차례로 실행한다.
* Void DrawOutLine() : 테트리스 게임의 기본적인 틀을 화면에 그려주는 함수. 블럭이 쌓이는 박스, 다음 블럭이 무엇인지 표시해주는 박스, score 박스 총 3개의 네모칸을 그려준다. 실제로 박스를 그리는 과정은 DrawBox함수에서 진행된다.
* Void DrawField() : 블럭이 쌓이는 박스에 field 배열의 정보를 그대로 옮겨서 그려주는 함수. attorn(A\_REVERSE), printw(‘ ‘), attroff(A\_REVERSE)로 하얀색 네모칸을 칠해주지만, 현재는 게임 시작 직후의 상황이므로 아무것도 그려지지 않았다는 의미의 printw(“.”)를 그려준다.
* Void DrawBlock(int blockY, int blockX, int blockID, int blockRotate, char tile) : 첫번째로 등장할 블럭을 화면에 그려주는 함수. 이 때, field 배열에는 반영하지 않는다. 아직 이동 중인 블럭이기 때문이다.
* Void DrawNextBlock(\*nextBlock) : 다음 블럭이 무엇인지 표시해주는 박스에 다음 블럭의 모양을 그려주는 함수.
* Void PrintScore(int score) : score을 점수가 표시되는 박스에 표시해주는 함수.
* Int GetCommand() : 키 입력을 받는 함수. Q나 q가 입력되면 리턴값을 QUIT로 주고, 유효하지 않은 입력이 주어질 시 NOTHING을 리턴한다.
* Int ProcessCommand(int command) : 각각의 키 입력에 대해 어떠한 행동을 할 것인지 명시해주는 함수. 방향키를 입력받았을 때 CheckToMove함수를 통해 이동가능한지 판단하고, 이동가능하면 DrawChange함수를 통해 블럭이 쌓이는 공간에 그림을 업데이트해준다.
* Void DrawBox(int y, int x, int height, int width) : 상자 모양의 공간을 그려주는 함수. 이 프로젝트에서는 블럭이 쌓일 박스, 다음 블럭의 정보를 표시해주는 박스, 점수가 쓰일 박스 총 3군데에 사용된다.
* Void play() : 테트리스 게임의 전반적인 루프가 포함된 함수. 방향키가 입력되면 조건에 따라 이동하고, QUIT가 입력되면 게임을 종료하고 Godd-Bye 라는 문구를 화면에 출력해준다.
* void menu() : a.out을 실행시키면 화면에 출력할 내용을 담고있다. 1,2,3,4 중 하나를 입력받고, 입력받은 수를 리턴한다.
* Int main() : ncurses를 이용하기 위한 초기화 과정을 진행해주는 initscr함수, 사용자가 입력한 내용을 화면에 출력하지 않게 하기 위한 noecho함수, 방향키와 특수문자까지 입력받기 위한 keypad(stdscr, TRUE)함수를 실행하고, menu를 실행한다.

2.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CheckToMove 의 코드 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

// user code

/////////////////////////////////////////////////////////////////

// 현재 블록을 움직일 수 없는 경우 //

// 1. 예상된 현재 블록이 필드 바깥으로 나갈 경우 //

// 2. 예상 현재 블록이 쌓여 있는 블록과 겹치는 경우 //

/////////////////////////////////////////////////////////////////

// 각각의 블록에 대하여

for (int i = 0 ; i < 4; i++){

for (int j = 0; j < 4 ; j++){

// 4X4 block prototype에서 블록이 있는 지점에 대해서

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

// 예상된 현재 블록이 필드 바깥으로 나갈 경우 움직일 수 없으므로 0을 리턴한다.

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

return 0;

// 예상 현재 블록이 쌓여 있는 블록과 겹치는 경우 움직일 수 없으므로 0을 리턴한다.

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

return 0;

}

}

}

// 예상된 현재 블록이 모두 필드 안에 있고 쌓여 있는 블록과 겹치지 않은 경우 움직일 수 있으므로 1을 리턴한다.

return 1;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* DrawChange 의 코드 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

// user code

//1. 이전 블록 정보를 찾는다. ProcessCommand의 switch문을 참조할 것

//2. 이전 블록 정보를 지운다. DrawBlock함수 참조할 것.

//3. 새로운 블록 정보를 그린다.

////////////////////////////// 1. 이전 블록 정보를 찾는다. ////////////////////////////

// 이전 블록의 위치, 회전횟수를 얻기 위해 이전 command를 살펴본다.

// preBlockRotate : 이전 블록의 회전횟수

// preBlockX : 이전 블록의 x좌표

// preBlockY : 이전 블록의 y좌표

int preBlockRotate; // 이전 블록의 회전 횟수

int preBlockX; // 이전 블록의 x좌표

int preBlockY; // 이전 블록의 y좌표

switch(command){

// 윗 방향키이면 블록이 반시계방향으로 90도 회전했다는 것을 의미하고,

// 이전 방향은 시계방향으로 90도 회전 시키면 된다.

case KEY\_UP:

preBlockRotate = (blockRotate-1) % 4;

preBlockX = blockX;

preBlockY = blockY;

break;

// 아래 방향키이면 블록이 아래로 한칸 내려왔다는 것을 의미하고,

// 이전 위치는 y좌표가 1 줄어든 위치이다. x좌표는 변함이 없다.

case KEY\_DOWN:

preBlockRotate = blockRotate;

preBlockX = blockX;

preBlockY = blockY - 1;

break;

// 오른쪽 방향키이면 블록이 오른쪽으로 한칸 이동했다는 것을 의미하고,

// 이전 위치는 x좌표가 1 줄어든 위치이다. y좌표는 변함이 없다.

case KEY\_RIGHT:

preBlockRotate = blockRotate;

preBlockX = blockX - 1;

preBlockY = blockY;

break;

// 왼쪽 방향키이면 블록이 왼쪽으로 한칸 이동했다는 것을 의미하고,

// 이전 위치는 x좌표가 1 늘어난 위치이다. x좌표는 변함이 없다.

case KEY\_LEFT:

preBlockRotate = blockRotate;

preBlockX = blockX + 1;

preBlockY = blockY;

break;

default:

break;

}

//////////////////////////// 2. 이전 블록 정보를 지운다. ///////////////////////////

// DrawBlock 함수를 참조했는데, 바꾼 점은 tile parameter가 "."로 고정이 되어있다는 점이다. 공백을 출력해야 하기 때문이다.

int i, j;

for (i = 0 ; i < 4 ; i++){

for (j = 0; j < 4 ; j++){

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

move(i+preBlockY+1, j+preBlockX+1);

printw(".");

}

}

}

//////////////////////////// 3. 새로운 블록 정보를 그린다. /////////////////////////

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

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* BlockDown 의 코드 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void BlockDown(int sig){

// user code

//강의자료 p26-27의 플로우차트를 참고한다.

// 블록이 한 칸 내려갈 수 있는 경우

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

// 블록을 한 칸 내리고

blockY++;

// 한 칸 아래로 내려온 블록을 콘솔에 그려준다.

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

// 시간을 다시 초기화해준다.

timed\_out = 0;

return;

}

// 블록이 한 칸 내려갈 수 없는 경우

// 블록의 y좌표가 -1 이면 gameOver = 1 로 변경해준다.

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

// 블럭정보를 필드에 반영해준다.

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

// 점수를 업데이트 해준다.

score += DeleteLine(field);

// 점수를 화면에 출력해준다.

PrintScore(score);

// 현재블럭을 다음블럭으로 변경해주고, 다음블럭을 랜덤으로 하나 지정해준다.

nextBlock[0] = nextBlock[1];

nextBlock[1] = rand() % 7;

// 다음 블럭을 화면에 출력해준다.

DrawNextBlock(nextBlock);

// 블럭의 위치와 회전횟수 정보를 초기화해준다.

blockRotate = 0;

blockY = -1;

blockX = WIDTH/2 - 2;

// 필드의 내용을 화면에 그려준다.

DrawField();

// 커서를 화면 바깥으로 이동해준다.

move(HEIGHT, WIDTH+10);

// 시간을 0으로 초기화해준다.

timed\_out = 0;

return;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* AddBlockToField 의 코드 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

// user code

//Block이 추가된 영역의 필드값을 바꾼다.

// block이 있는 곳을 field에도 반영해준다.

for (int i = 0; i < 4; i++){

for (int j = 0; j < 4; j++){

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

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

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* DeleteLine 의 코드 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

// user code

//1. 필드를 탐색하여, 꽉 찬 구간이 있는지 탐색한다.

//2. 꽉 찬 구간이 있으면 해당 구간을 지운다. 즉, 해당 구간으로 필드값을 한칸씩 내린다.

int isFull; // 현재 탐색중인 줄이 완전히 채워져 있는지의 여부를 담는 변수

int deletedLines = 0; // 지운 line의 개수를 저장하는 변수

int addedScore; // 추가 될 점수를 저장하는 변수

for (int i =0; i < HEIGHT; i++){

isFull = 1;

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

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

isFull = 0;

break;

}

}

// 현재 탐색 중인 줄이 완전히 채워져 있다면

if (isFull){

// 지운 line의 수를 하나 늘려준다.

deletedLines++;

// i-1번째 줄을 i번째로 옮겨주고, 가장 위에 줄이면 빈칸을 할당해준다.

for (int k = i; k >= 0; k--){

if (k == 0){

for (int l = 0; l < WIDTH; l++)

f[k][l] = 0;

}

else{

for (int l = 0 ; l < WIDTH ; l++)

f[k][l] = f[k-1][l];

}

}

}

}

// 점수를 저장한다.

addedScore = deletedLines\*deletedLines\*100;

return addedScore;

}