6주차 예비보고서

전공: 수학/컴퓨터공학 학년: 3학년 학번: 20181294 이름: 임승섭

**1.**

Tetris 게임의 flow chart는 크게 두 가지로 설명할 수 있다. 코드 전반에 사용되는 함수에 대한 설명도 포함하였다.

1-1. play() 함수의 flow chart

play() 함수는 main에서 게임을 실행한다고 명령이 들어오면 수행을 시작한다. 우선 InitTetris() 함수를 실행시켜서 기본적인 변수와 자료구조를 초기화한다. 먼저 테트리스 필드를 모두 ‘0’으로 초기화한다. ‘0’은 필드가 채워지지 않음을 의미하고, 화면에는 ‘.’으로 표시된다. 반대로 ‘1’은 필드가 채워졌음을 의미한다. 다음은 현재 블록과 다음 블록의 id를 임의로 생성한다. 이들은 int nextBlock[BLOCK\_NUM] 에 저장된다. 초기에는 BLOCK\_NUM이 2로 설정되어 있다. 각 블록의 id는 rand함수를 이용해서 0부터 6까지의 수를 임의로 생성된다. 7개의 숫자는 각각 하나씩의 블록의 종류를 의미한다. 다음은 현재 블록의 상태를 초기화한다. 현재 회전수는 0, x좌표는 (width/2) - 2, y좌표는 -1로 초기화한다. 다음은 게임 전반에 필요한 score, gameover 변수를 0으로 초기화한다. score는 점수, gameoversms 게임의 종료 여부를 나타내며 게임 종료 순간 1로 변경된다. 다음은 BlockDown()에서 필요한 timed\_out 변수를 0으로 초기화한다. 마지막으로 테트리스 화면을 그려준다. 이 과정에서 DrawOuntline() (화면에서 나타나는 상자들의 테두리를 그림), DrawField() (테트리스 필드를 0 or 1로 그림), DrawBlock() (블록의 좌표, 모양, 회전수에 따라 정해진 위치에 블록을 그림), DrawNextBlock() (nextblock의 id를 바탕으로 다음 블록의 모습을 상자 안에 그림), PrintScore() (score 값을 상자 안에 그림) 의 함수를 사용한다.

다음은 GetCommand() 함수가 실행되서 사용자의 입력을 기다린다. 들어온 입력을 구분해서 그대로 return한다. 구분되는 입력은 KEY\_UP, KEY\_DOWN, KEY\_LEFT, KEY\_RIGHT, ‘Q’(‘q’) 로 총 5개 이다.

이 반환된 명령을 ProcessCommand() 함수가 입력으로 받는다. 각 입력에 따라 적절한 동작을 취한다. 이전 함수와 비슷하게 명령을 5개로 구분하여 각 명령에 따른 동작을 수행한다. ‘Q’(‘q’)가 들어오면 게임이 강제 종료되면서 tetris.h에 정의되어 있는 QUIT을 반환한다. 이후 “Good-bye!!”라는 메시지를 화면에 출력한다. KEY\_UP이 들어오면 CheckToMove() 함수를 이용해서 시계 반대방향으로 90도 회전할 수 있는지 여부를 체크한다. KEY\_DOWN이 들어오면 CheckToMove() 함수를 이용해서 블록을 아래로 이동할 수 있는지 여부를 체크한다. KEY\_LEFT가 들어오면 CheckToMove() 함수를 이용해서 블록을 왼쪽으로 이동할 수 있는지 여부를 체크한다. KEY\_RIGHT이 들어오면 CheckToMove() 함수를 이용해서 블록을 오른쪽으로 이동할 수 있는지 여부를 체크한다.

만약 ProcessCommand 함수의 결과가 QUIT이 아니면 다시 GetCommand() 함수로 돌아가서 다시 새로운 입력값을 기다린다.

결론적으로 play() 함수는 키보드의 입력을 받아, 적절한 동작을 수행하는 함수이다.

1-2. BlockDown() 함수의 flow chart

BlockDown() 함수는 일정한 시간마다 현재 블록의 위치를 아래로 한 칸 내리는 함수이다. 내릴 수 없는 상황에는 필요한 함수를 실행시켜 그 상황을 처리한다.

우선 CheckToMove() 함수를 통해 블록을 한 칸 내릴 수 있는지 확인한다. 내릴 수 있으면 blockY++ 을 해주어서 한 칸 내려주고 블록위 위치를 새로 그린 후 함수를 종료한다.

블록을 한 칸 내릴 수 없는 상황은 두 가지로 나누어 생각할 수 있다.

첫 번째, 블록이 맨 꼭대기에서 내려갈 수 없으면 맨 위까지 블록들이 쌓여있는 상황이고, 이는 게임을 실패한 경우이다. 이를 blockY 가 -1 인지로 확인한다. 만약 blockY가 -1이라면 gameOver 변수를 1로 변경해서 게임을 종료하도록 한다.

두 번째, 블록이 최대한 아래로 내려가서 더 이상 내려갈 수 없는 상황이고, 이는 그 위에 블록을 쌓아주면 된다. 마찬가지로 blockY가 -1이 아니면 이 상황이라고 유추할 수 있고, 블록을 쌓아주는 과정은 AddBlockToField() 함수를 이용한다.  
 AddBlockToField() 함수는 입력인자로 필드의 정보, 현재 블록의 id, 회전수, y좌표, x좌표를 받는다. 각 블록은 4x4 크기이기 때문에 이중 loop을 사용해서 현재 블록의 상태(currentBlock)을 확인하고 블록이 채워져 있는 부분을 통해 테트리스 필드의 정보(field)를 업데이트한다. 현재 블록의 위치 + i(or j) 의 상태를 0에서 1로 바꾸어준다.

블록이 필드에 쌓였으면 각 줄의 빈칸 유무를 확인해서 빈칸이 없을 시 줄을 없애주는 DeleteLine() 함수를 실행한다. DeleteLine() 함수는 입력인자로 필드의 정보를 받는다. 여기도 마찬가지로 이중 loop을 이용해서 필드의 정보를 확인하고 다 채워진 줄이 있는지 확인한다. 빈칸이 하나라도 있으면 loop을 중단하고 다음 줄을 탐색하여 총 완성된 줄의 개수를 계산한다. 이 라인의 수를 이용해 점수를 계산하여 반환한다.

이후에는 다시 새로운 블록으로 업데이트해야 한다. 따라서 현재 블록(nextBlock[0])을 다음 블록(nextBlock[1])으로 바꿔주고 다음 블록(nextBlock[1])의 id를 다시 0부터 6까지의 숫자를 랜덤으로 뽑아 설정해준다. 또한 현재 블록에 대한 변수들(회전수, y좌표, x좌표)를 초기 상태로 초기화시킨다. 모두 초기화되면 상황을 화면에 보여주고 함수를 종료한다.

**2.**

2-1. int CheckToMove(char field[HEIGHT][WIDTH], int currentBlock, int blockRotate, int blockY, int blockX)

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

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

if currentblock[i][j] == 1

if (field[blockY + i][BlockX + j] == 1)

return 0;

else if (blockY + i >= HEIGHT)

return 0

else if (blockX + j < 0)

return 0

else if (blockX + j > WIDTH)

return 0

return 1

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

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

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

filed[blockY + i][blockX + j] = ‘.’

if (command == KEY\_UP)

DrawBlock(blockY, blockX, currentBlock, (blockRotate + 1)%4, 1)

else if (command == KEY\_DOWN)

DrawBlock(blockY+1, blockX, currentBlock, blockRotate, 1)

else if (command == KEY\_LEFT)

DrawBlock(blockY, blockX - 1, currentBlock, blockRotate, 1)

else if (command == KEY\_RIGHT)

DrawBlock(blockY, blockX + 1, currentBlock, blockRotate, 1)

2-3. int BlockDown(int sig)

if (CheckToMove(field[HEIGHT][WIDTH], currentBlock, blockRotate, blockY+1, blockX) == 0)

if (blockY == -1)

gameover = 1

else

AddBlockToField(field[HEIGHT][WIDTH], currentBlock, blockRotate, blockY, blockX)

score += DeleteLine(field)

PrintScore(score)

nextBlock[0] = nextBlock[1]

nextBlock[1] = rand() %7

currentBlock = nextBlock[0]

blockRotate = 0

blockY = -1

blockX = (WIDTH/2) - 2

DrawBlock(blockY, blockX, currentBlock, 0, 1)

DrawNextBlock(nextBlock[1])

DrawField()

else

blockY++

DrawChange(field, KEY\_DOWN, currentBlock, blockRotate, blockY, blockX)

2-4. int AddBlockToField(char field[HEIGHT][WIDTH], int currentBlock, int blockRotate, int blockY, int blockX)

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

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

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

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

2-5. int DeleteLine(char field[HEIGHT][WIDTH])

int DeadLine = 0

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

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

if (field[i][j] == 0)

break;

if (j == WIDTH-1) &&(field[i][j] != 0)

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

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

field[k][t] = field[k-1][t]

DeadLine++;

return (DeadLine)\*(DeadLine)\*100