6주차 예비보고서

전공: 수학과,컴퓨터공학과 학년: 3학년 학번: 20181256 이름: 김도현

1.1

전체적인 테트리스의 flow chart이다.

Start main()

Menu()

종료(EXIT==1 이라면)

MENU\_PLAY 라면

InitTetris()

GetCommand()

ProcessCommand()

QUIT==FALSE

ProcessCommand()

==Quit?

QUIT==TRUE play() 동작 끝

Terminate main()

Menu가 종료를 의미하면 그냥 바로 게임이 종료된다. 만약 Menu 가 play(1주차 실습)로 되면 Play 함수 실행이 된다. 우선 InitTetris에서 필요한 변수들을 초기화 한다. 이후 GetCommand()를 통해 자판의 방향키, ‘q’ 등을 input으로 받아 이에 해당하는 행동을 ProcessCommand()에서 결정한다. 이때 CheckToMove 함수로 블록이 움직일 수 있는 지 판단할 수 있고 가능할 시 해당 위치로 DrawChange 로 블록을 그린다. 이 과정을 반복적으로 할 수 있도록 다시 getCommand()로 돌아간다. 만약 command가 ‘q’ 일 경우 QUIT=true 가 되며 게임이 종료 된다.

Global 변수들은 대표적으로 rand한 지금과 다음 블록을 저장하는 배열 nextBlock, 블록 회전수를 알려주는 blockRotate, 블록 x좌표 blockX, 블록 y좌표 blocky, 점수 score, 게임 종료 판단 gameOver, 매초를 의미하는 변수인 time\_out 등을 생각할 수 있다.

게임이 실행되는 동안 사용자가 입력을 하던 하지 않던 매 시간마다 동작이 일어나는 BlockDown() flow chart이다.

Start BlockDown()

CheckToMove()

CheckToMove==TRUE

CheckToMove==FALSE

blockY == -1?

BlockY==-1(Yes), 게임 종료

blockY++(Block 한칸 내림)

No

블록 쌓기

gameOver=TRUE

AddBlockToField()

DeleteLine()

(line is full check,score update)

nextBlock[0]=nextBlock[1]

nextBlock[1]=rand()%7

Initializing process(like location of currentBlock, drawfield,drawnextblock)

BlockDown() 동작 끝

Terminate BlockDown()

게임은 매 순간마다 한 칸 씩 내려가야하기 때문에 CheckToMove를 호출해서(KEY\_DOWN) 가능 여부를 판단한다. 만약 가능하다면 그냥 blockY를 1 추가해준다. 즉 한 칸을 내린다는 의미이다. 그렇지 않다면 더이상 블럭이 내려올 수 없기 때문에 게임이 끝났는지 블럭이 쌓이는지 판단한다. blocky==-1 이면 게임이 종료 된다는 의미의 gameOver를 true로 한다. 아니라면 필드에 블럭이 쌓여야하기 때문에 AddblockTofield로 블럭을 field에 넣는다. 그리고 라인이 다 찾는지 확인해서 그 줄을 지울지 말지 DeleteLine을 호출해 판단 및 score 점수도 계산하여 기존의 score 에 더한다. 여기서는 score는 (지운 줄수)^2\*100으로 하였다. 이후 다시 블록을 초기화해주고 다음 블록이 나올 수 있도록 rand()를 이용해 랜덤하게 해준다. DrawNextBlock이나 DrawField 등으로 바뀐 정보들을 업데이트 해준다. 계속 이 과정을 반복해준다.

1.2

다음은 테트리스 코드를 구현하기 위한 함수들을 정리해보았다.

int main() 에서는 프로그램을 실행 후 먼저 초기화 과정을 해 준다. 입력값 같은 경우 ncurses.h 을 이용하여 초기화 시키는 initscr(), 타이핑이 안보이도록 설정하는 noecho(), 화살표 같은 특수 자판을 입력가능한 keypad(stdscr, TRUE)가 있다. Menu를 선택하여 그 입력값에 맞는 코드를 실행한다. 현재 이번 실습시간에는 MENU\_PLAY와 MENU\_EXIT이 있다.

void InitTetris()

현재 게임에서 사용하는 변수 및 화면을 초기화한다. 우선 필드를 ‘0’으로 하고 x,y좌표도 기본 값(WIDTH/2-2, -1) 으로 둔다. score, gameover, time\_out 모두 0 으로 초기화를 해준다. 초기화를 한 이후 초기화 상태인 테트리스 화면을 아래 설명할 함수들을 이용해 보여준다.(그린다.)

void DrawOutline()

이름 그대로 전체적인 outline을 그리는 함수로 테트리스 게임 화면 겉 테두리, nextBlock을 보여줄 공간의 테두리, score를 나타낼 공간의 테두리를 그린다.

int GetCommand()

프로그램 실행 중 자판의 방향키 및 ‘q’ 등을 입력 받아 입력 값에 따라 그 해당 입력 값을 return 한다.

int ProcessCommand(int command)

GetCommand()의 return 값을 받아 해당 command에 따른 코드를 실행한다. 이때 ‘q’ 즉 QUIT을 받을 시 종료하라는 의미이기 때문에 QUIT을 return 하고 나머지는 1을 return 한다. 여기서 방향키에 따라 움직일 수 있는지 여부는 CheckToMove, drawFlag에 따라 DrawChange도 쓰였다.

void DrawField()

필드를 구성하는 함수로 0이면 빈 필드 ‘.’ , 1이면 채워진 필드로 attron(),attroff()로 ‘ ’으로 나타낸다.

void PrintScore(int score)

move(11,WIDTH+11) (점수 표시하는 공간) 에 게임의 score를 화면에 나타낸다.

void DrawNextBlock(int \*nextBlock)

nextBlock을 표시하는 공간에 다음 nextBlock(4x4)의 ID를 받아 이를 화면에 나타낸다.

void DrawBlock(int y, int x, int blockID, int blockRotate, char title)

받아오는 input 값들을 이용해서 (y,x)에 blockID나 blockRotate 를 토대로 해당 위치에 title 모양으로 채워 block을 그린다.

Void DrawBox(int y, int x, int height, int width)

받아오는 input 값에 따라 해당 위치에 (y,x)를 왼쪽 위 모서리로 두고 height, width를 이용해 상자를 그린다.

void play()

게임의 기본 값들을 초기화 시키고 전체적으로 게임을 play 하기 위한 과정을 수행하는 함수로 생각하면 된다.

char menu()

메뉴를 printw 하고 입력 값이 생기면 해당 메뉴 값을 return 한다. 1(play) 2(rank) 3(recommend play) 4(exit) 이 있고 이번 실습은 1번을 진행한다.

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

input 값들을 토대로 currentblock이 이동하고자 하는 곳으로 갈 수 있는지 없는지를 판단하여 가능하면 1, 불가능하면 0을 return 한다.

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

input 값들을(바뀐 정보) 토대로 기존의 위치를 찾아 block을 없애고 새 위치 또는 blockRotate를 보고 회전을 한 block을 그린다.

int BlockDown(int sig)

위의 flowchart에서도 보았듯이 일정 시간동안 하나 씩 block을 아래로 내리는데 내려갈 공간이 없으면 block을 쌓는다. 이때 block을 더 못 쌓으면(blockY==-1) 게임이 종료된다. 아니면 block을 필드에 넣고 줄이 다 차 있다면 해당 줄을 지우고 score를 계산한다. 이후 nextBlock을 가져오고 이런 정보들을 새롭게 하고 이런 값들을 화면에 보여준다.

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

현재 필드 위치에 currentblock을 쌓는 곳이다.

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

필드를 돌아다니며 한 줄이 꽉 채워져 있는지 확인한다. 만약 있다면 그 줄을 지운다. 그리고 지운 라인에 대해 점수를 측정해 return 해준다.

2.1

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

{

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

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

if(block[currentBlock][blockRotate][i][j]==1) //블록이 해당위치에 있을 때 (4x4 중)

i+blockY >=HEIGHT, j+blockX >=WIDTH, j+blockX <0,f[y][x]==1 => return 0

//원하는 위치의 y,x 값이 필드보다 크거나 작거나 이미 필드가 차있다면 0 return

return 1 //아니면 1을 return 즉 블록이 다른 위치로 바뀔 수 있음을 의미

}

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

{

switch(command)

case KEY\_UP: //key\_up이면 이전은 rotate가 시계방향을 의미

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

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

if(block[currentBlock][prevblockRotate][i][j]==1 && i+blockY>=0)

move(i+blockY+1,j+blockX+1)

printw(“.”) //이전 블록을 지우기

break

case KEY\_DOWN: //key\_down이면 이전은 y-1 위치를 의미

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

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

if(block[currentBlock][blockRotate][i][j]==1 && i+(prevY)>=0)

move(i+prevY+1, j+blockX+1)

printw(“.”) //이전 블록 지우기

break

case KEY\_LEFT: //key\_left이면 이전은 x+1 위치를 의미

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

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

if(block[currentBlock][blockRotate][i][j]==1 && i+blockY>=0)

move(i+blockY+1, j+prevX+1)

printw(“.”) //이전 블록 지우기

case KEY\_RIGHT: //key\_right이면 이전은 x-1 위치를 의미

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

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

if(block[currentBlock][blockRotate][i][j]==1 && i+blockY>=0)

move(i+blockY+1, prevX+1)

printw(“.”) //이전 블록 지우기

break

DrawBlock(blockY,blockX,currentBlock, blockRotate, “ ”) //새로운 블록 그리기

move(HEIGHT, WIDTH+10) //커서를 필드 밖으로 이동

}

int BlockDown(int sig)

{

if(CheckToMove(f, nextBlock[0], blockRotate, blockY+1,blockX)) //checktomove로 blockY+1일 때 블록 한칸 내릴 수 있는지

blockY=blockY+1; //blockY값 1 추가

DrawChange(f, KEY\_DOWN, currentBlock, blockRotate, blockY,blockX) //key\_down으로 블록 그림

else

if (blockY==-1) gameOver=1 //블록 못 내리고 blocky==-1 이면 gameover

AddBlockToField(f, nextBlock[0],blockRotate, blockY,blockX) //아니면 필드에 블록 추가

nextBlock[0]=nextBlock[1] //블록 초기화 과정

nextBlock[1]=rand()&7;

blockRotate=0;blockY=-1,blockX=(WIDTH/2)-2

score=score+DeleteLine(f) //deleteline으로 꽉찬라인 지우고 score에 점수 추가

printScore(score) //스코어 출력

DrawNextBlock(nextBlock) //다음 블록을 해당 박스에 그리기

DrawField() //필드 업데이트

}

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

{

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

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

if(Block[currentBlock][blockRotate][i][j]==1) //해당 위치에 블록이 있을때(4x4)

field[blockY+i][blockX+j]=1 //필드에 추가(blockY+i,blockX+j)

}

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

{

bool checkline=false //flag checkline

int completedLine=0 //꽉 찬 line 개수 변수

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

checkline=true //checkline=true로 두고 false가 나오는지 확인

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

if f[i][j]==0 //필드 비었으면

checkline=false //checkline false

if(checkline) //checkline으로 꽉 찬 line 있으면

for(int k=i;k>=1;k--) //해당 y 축 그 위로 모든 블록들을 하나 아래로 내림

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

f[k][s]=f[k-1][s] //x축은 그대로 내려옴

completedLine++ //line 1추가

return completedLine\*completedLine\*100 //점수는 (꽉 찬 line 제곱 )\* 100

}