2020년도 2학기 컴퓨터공학설계및실험I

6주차 결과 보고서

20170175 김태안

1. 실습 목적

누구에게나 친숙하고 유명한 게임인 테트리스(tetris)를 구현한다. 실험 1주차에서는 제공된 프레임(frame) 프로그램을 바탕으로, 블록의 이동, 블록의 회전, 블록을 필드에 쌓기, 줄 삭제, 점수 계산, 블록 미리 보여주기로 구성된 기본적인 기능들을 갖는 테트리스 게임을 구현하고, ncurses 라이브러리, 디버깅 방법, makefile을 만드는 방법 등을 익히도록 한다.

1. 실습 구현 내용
2. int CheckToMove(char f[HEIGHT][WIDTH], int currentBlock, int blockRotate,   
   int blockY, int blockX);

input에 해당하는 위치에 블록을 놓을 수 있는 지 판별하는 함수이다. 블록을 놓으려는 위치에 이미 블록이 쌓여 있는 경우, 블록의 위치가 필드 범위를 벗어난 경우 0을 반환하며 움직일 수 없음을 보이고, 놓을 수 있는 경우 1을 반환한다.

input: 테트리스 필드와 놓으려는 블록 정보 (모양, 방향, 위치)

output: 블록이 놓일 수 있는 지 여부 (0, 1)

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

방향키를 눌렀을 때 변경된 블록의 모양을 출력한다. 입력된 커맨드를 바탕으로 이전의 모양을 찾아 블록을 지운 뒤, 변경된 새 블록을 출력한다.

input: 테트리스 필드, 블록 정보, 입력한 커맨드

output: 변경된 블록 출력

1. void BlockDown(int sig);

매초 테트리스 블록을 한 칸씩 내리는 함수이다. 블록이 내려갈 수 있으면, 블록을 한 칸씩 내리고, 더 이상 내려갈 수 없다면 블록을 필드에 붙인다. 이때, 한 줄이 꽉 차 지울 수 있다면 블록을 지우고 점수룰 반환 받는다. 그 후, 새 블록을 떨어트리기 시작한다.

input: signal

output: 떨어지는 블록과 블록이 쌓여 변경된 필드를 출력

1. int AddBlockToField(char f[HEIGHT][WIDTH], int currentBlock, int blockRotate,   
   int blockY, int blockX);

테트리스 필드에 블록을 추가해 블록을 쌓는 함수이다.

input: 테트리스 필드, 쌓을 블록의 정보

output:

1. int DeleteLine(char f[HEIGHT][WIDTH]);

테트리스 필드의 한 줄이 채워졌는 지 확인하고 채워졌다면 해당 구간으로 필드 값을 내린다. 그리고, 지워진 줄에 따라 점수를 반환한다.

input: 테트리스 필드

output: 지워진 줄에 따른 점수

1. 실습 환경

cspro.sogang.ac.kr

OS: Ubuntu 16.04.2 LTS (GNU/Linux 4/4/0-184-generic x86\_64)

Compiler: gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.12)

1. 과제
2. 그림자 기능
3. void DrawShadow(int y, int x, int blockID, int blockRotate);

현재 블록이 떨어지면 위치할 곳을 표시하는 그림자를 그리는 함수이다. 현재 블록의 y좌표를 최대한 내려, 놓을 수 있는 위치에 ‘/’ 모양의 블록을 그린다.

input: 현재 블록의 정보

output: 그림자 출력

1. void DrawBlocksWithFeatures (int y, int x, int blockID, int blockRotate);

DrawBlock과 DrawShadow를 동시에 호출해 블록을 더 편하게 그릴 수 있도록 돕는 함수이다.

input: 그릴 블록 정보

output: -

1. 다음 블록 보여주기 추가
2. void InitTetris();

2번째 다음 블록을 추가하기 위해 nextBlock의 크기를 1 늘리고, InitTetris에서 해당 블록도 rand 함수를 통해 값을 할당해주도록 변경하였다.

1. void DrawNextBlock(int \*nextBlock);

기존의 DrawNextBlock과 같은 방식으로 11+i, WIDTH+13에 2번째 다음 블록을 그린다.

1. void DrawOutline();

2번째 다음 블록을 그릴 박스를 추가로 그린다. 기존의 점수 박스의 위치를 수정한다.

1. void BlockDown(int sig);

블록이 필드에 쌓였을 때, nextBlock[0]에 nextBlock[1]을, nextBlock[1]에 nextBlock[2]를, nextBlock[2]에 새 랜덤 값을 저장한다.

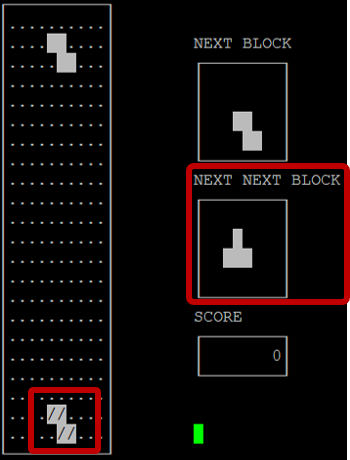
1. 새 점수 계산법
2. int AddBlockToField(char f[HEIGHT][WIDTH], int currentBlock, int blockRotate,   
   int blockY, int blockX);

블록을 쌓을 때, 블록의 y좌표+1이 필드의 채워진 부분이거나, 필드의 맨 아래 (HIEGHT)라면 touched를 증가시켜 블록의 밑과 필드가 닿는 면적을 계산한다. 그리고, touched\*10의 점수를 반환한다.

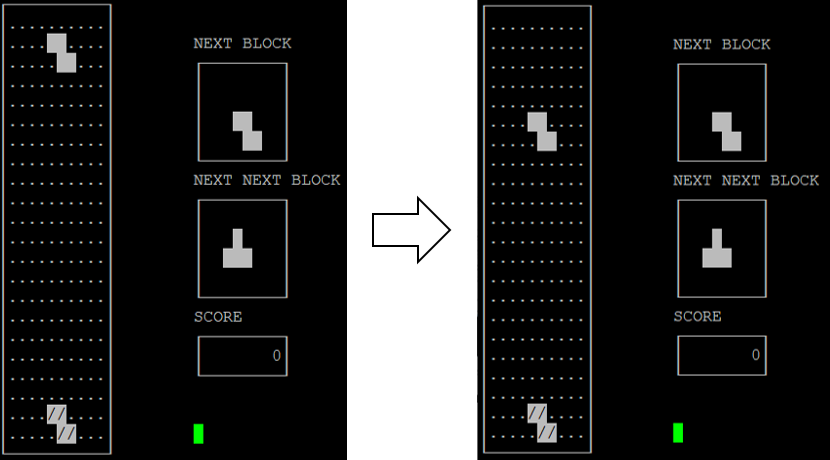
1. int DeleteLine(char f[HEIGHT][WIDTH]);

줄이 지워질 때 마다 지워진 줄의 수를 기억한 뒤, 지워진 줄 수2 X 100의 점수를 반환한다.

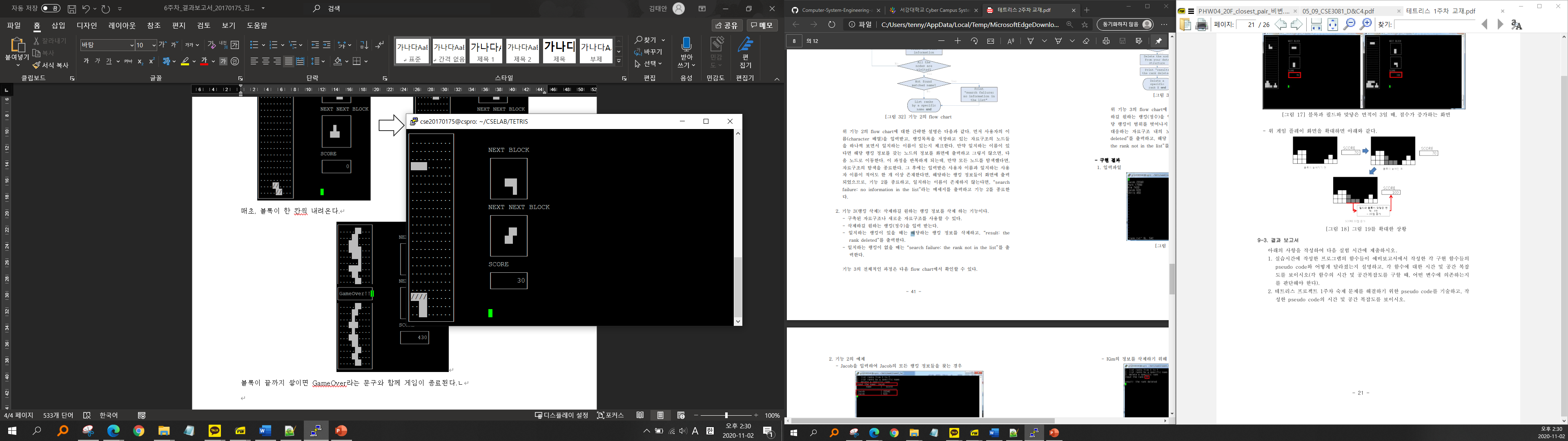
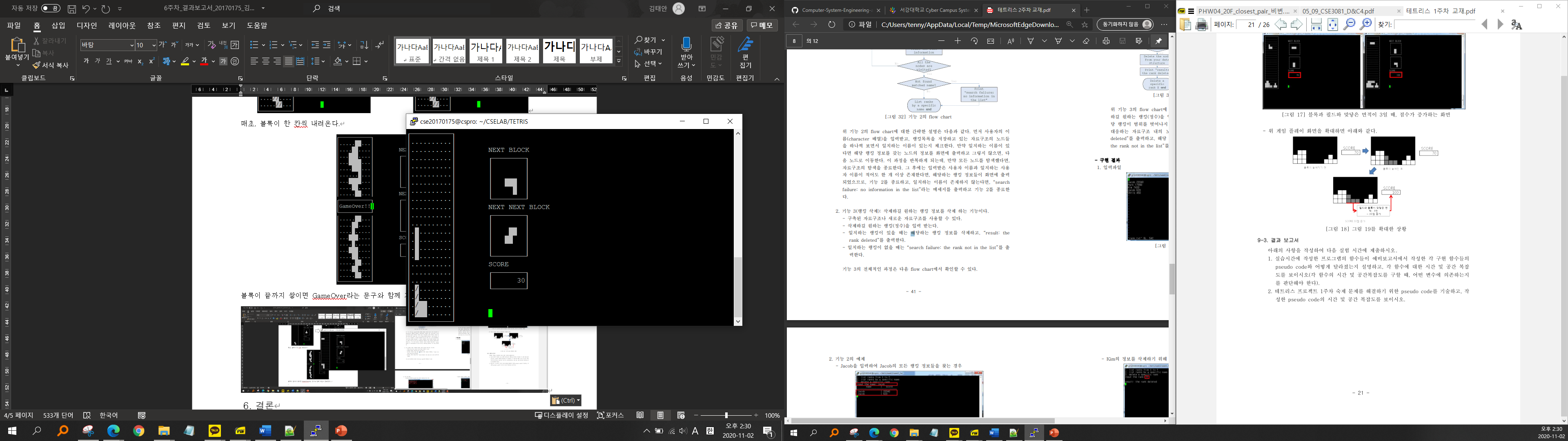
1. 결과 및 분석



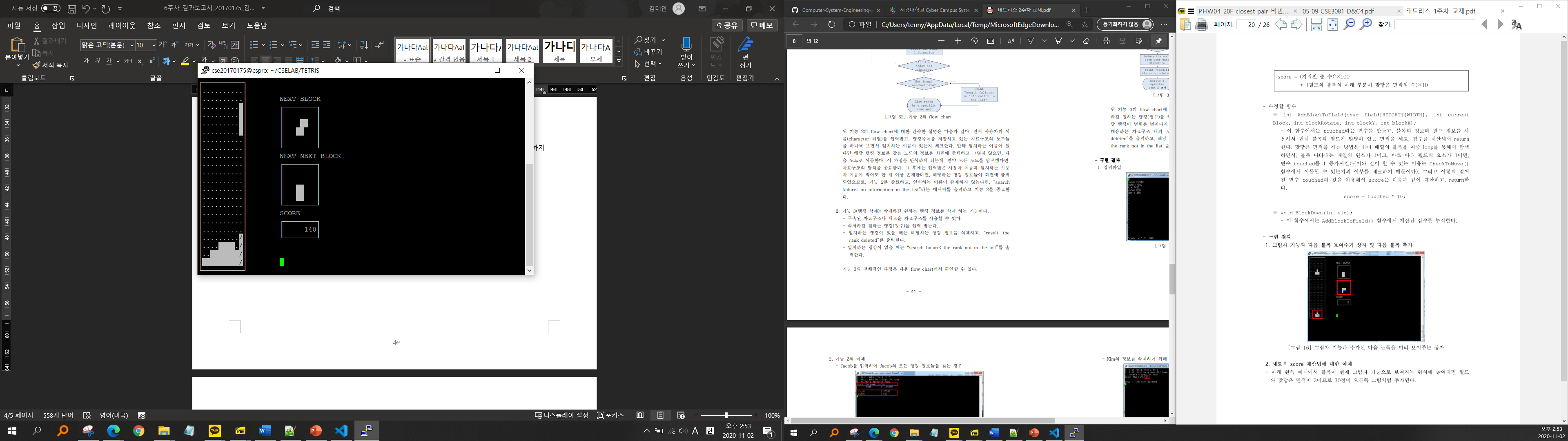
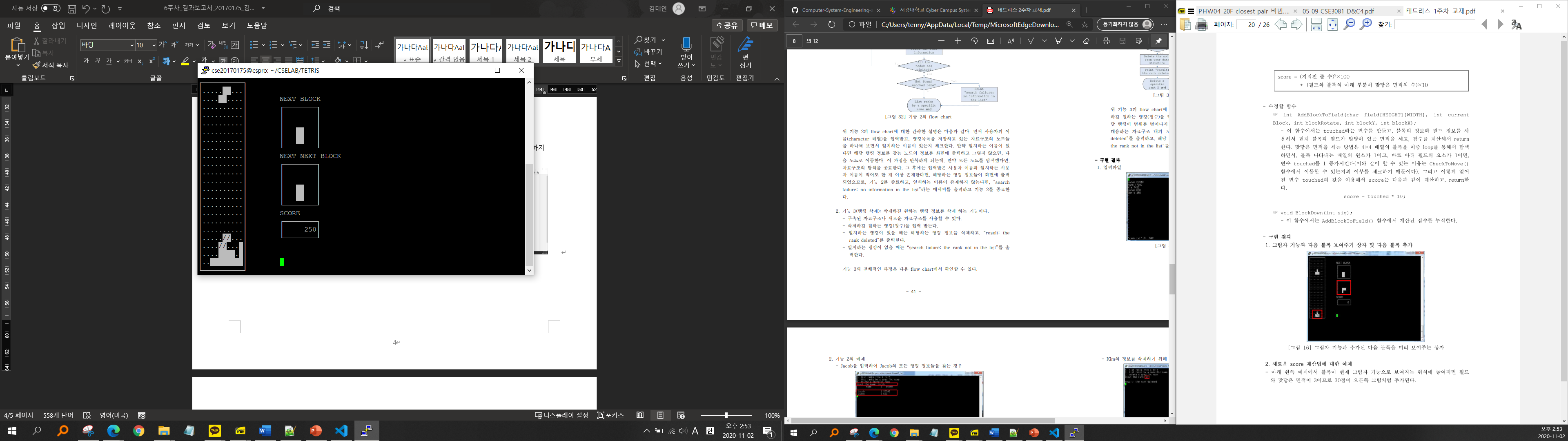
화면에 NEXT NEXT BLOCK과 그림자가 추가된 것을 확인할 수 있다.



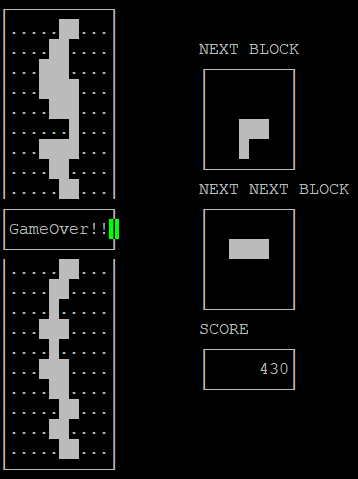
매초, 블록이 한 칸씩 내려온다.

방향키를 누르면 블록이 이동하거나 돌아가며, 이동할 수 없는 위치로는 이동하거나 회전하지 않는다.

한 줄이 꽉 차면 줄이 삭제되며, 삭제된 줄 수2 X 100 + 블록의 밑이 닿은 면 수 X 10의 점수를 얻는다.



블록이 닿으면 블록이 쌓이며, 블록이 끝까지 쌓이면 게임이 종료된다.

**함수 분석**

1. int CheckToMove(char f[HEIGHT][WIDTH], int currentBlock, int blockRotate,

int blockY, int blockX);

CheckToMove는 두 개의 for문을 통해 Block\_Height와 Block\_Width를 탐색하며 블록이 놓일 수 있는 지 판별한다. 따라서 O(Block\_Height X Block\_Width)의 시간 복잡도를 가진다.

1. void DrawChange(char f[HEIGHT][WIDTH],int command, int currentBlock,

int blockRotate, int blockY, int blockX);

DrawChange는 두 개의 for문을 통해 이전 블록의 출력을 지운다. 그리고, DrawBlock과 DrawShadow를 호출해 새 블록을 그린다. 따라서 O(3 X Block\_Height X Block\_Width +Height) = O(Block\_Height X Block\_Width)의 시간 복잡도를 가진다.

1. void BlockDown(int sig);

BlockDown 함수는 CheckToMove를 호출한 뒤, 움직일 수 없다면, AddBlockToField와 DeleteLine, DrawField, DrawBlocksWithFeatures 함수를 호출한다. AddBlockToField 함수는 두개의 for문을 통해 Block의 탐색을 진행하는 O(Block\_Height X Block\_Width)의 함수이다. DeleteLine은 필드의 탐색을 진행하는 O(Height X Width)의 함수이다. DrawField도 마찬가지로 필드를 탐색하는 함수로, O(Height X Width)의 시간 복잡도를 가지며, DrawBlocksWithFeatures는 O(Block\_Height X Block\_Width +Height)의 시간 복잡도를 가진다. 따라서 BlockDown은 O(2 X Height X Width +2 X Block\_Height X Block\_Width) = O(Height X Width)의 시간 복잡도를 가진 함수라고 할 수 있다.

각 함수는 사용한 로컬 변수 i, j 등 외의 메모리를 사용하지 않으므로 공간 복잡도는 O(1)이다. 전체 프로그램은 7개의 4개의 형태를 가진 블록을 사용하므로 O(NUM\_OF\_SHAPE X NUM\_OF\_ROTATE X BLOCK\_HEIGHT X BLOCK\_WIDTH)의 공간 복잡도를 가진 블록과, O(HEIGHT X WIDTH)의 공간 복잡도를 가지는 필드를 사용한다.

1. 결론

이번 실습에서는 ncureses 라이브러리를 활용한 테트리스 프로그램을 작성하였다. ncureses 라이브러리를 통해 텍스트 유저 인터페이스를 구현할 수 있으며, 문자의 출력으로 이루어진 테트리스 프로그램을 작성할 수 있었다. 또한, ncureses를 사용해 방향키와 숫자키에 새로운 기능을 매핑하여 게임을 즐길 수 있도록 구현하였다.