**CSE3013 컴퓨터공학설계및실험I: 테트리스 프로젝트**

**담당교수: 서강대학교 컴퓨터공학과 김승욱**

**1. 설계 문제 및 목표**

ncurses 라이브러리가 제공하는 리눅스 터미널 상에서의 GUI를 이용하여 테트리스 게임 프로그램을 제작하고 여기에 랭킹 시스템과 블록 배치 추천 기능을 추가 구현한다.

테트리스 게임은, 블록을 90도씩 반시계 방향으로 회전하거나 세 방향(좌, 우, 하)으로 움직여 필드에 쌓으면, 빈틈없이 채워진 줄은 지워지고 그에 따른 스코어를 얻어 결과적으로 가장 높은 최종 스코어를 얻는 것이 목적이다.

프로그램을 실행하면 사용자는 메뉴를 선택할 수 있다. 1번을 선택하면 테트리스 게임을 플레이할 수 있고, 2번을 선택하면 랭킹 정보를 확인할 수 있고, 3번을 선택하면 블록 배치 추천 기능을 따라 플레이되는 모드로 테트리스 게임이 실행되고, 마지막으로 4번을 선택하면 프로그램이 종료된다.

1주차 실험에서는 테트리스 게임을 구현한다. 미리 제공된 프레임 프로그램을 바탕으로 블록의 이동과 회전, 쌓기, 한 줄이 쌓인 줄의 삭제, 점수의 계산, 다음 블록 보여주기 등 기본적인 기능을 구현한다. 추가로, 게임의 편의를 위해 블록이 놓일 위치를 알려주는 그림자 기능도 구현한다.

2주차 실험에서는 플레이한 테트리스의 점수를 기록하는 랭킹 시스템을 구현한다. 테트리스 게임이 종료되면 사용자의 이름과 점수를 등수순으로 기록하고, 기록된 정보를 조건에 따라 확인/삭제할 수 있는 기능을 구현한다. 기록된 점수는 프로그램이 종료되어도 삭제되는 것이 아니라, 유지되도록 구현한다.

3주차 실험에서는 사용자를 위해 테트리스의 블록이 놓일 위치를 추천해주는 추천 시스템을 구현한다. 추천해 주는 위치는 현재 블록과 다음 블록을 통해 계산하며, 가장 높은 점수를 얻을 수 있는 위치를 추천한다. 추가로, 추천 시스템이 자동으로 테트리스를 플레이하는 추천 플레이 기능을 구현한다.

**1.1 현실적 제한조건**

실험을 통해 구현한 테트리스는 몇 가지 제한 조건이 존재한다. 먼저, 테트리스 게임은 Windows 환경에서는 사용할 수 없는 Ncurses 라이브러리를 통해 구현되었다. 따라서 유닉스 운영체제에서만 실행 가능하다. 그리고, Text User Interface (TUI)를 사용하였기 때문에 프로그램의 외관에 한계가 존재한다. 화려한 그래픽을 사용한 게임 화면은 기대할 수 없으며, CP437 인코딩 방식을 기준으로 텍스트를 통해 게임 화면을 표시한다.

**2. 요구사항**

**2.1 설계 목표 설정**

1주차 실험에서는 테트리스 게임의 동작을 목표로 한다. 테트리스 게임이 동작하기 위해선 다음의 기능을 구현해야 한다.

|  |  |
| --- | --- |
| 블록의 이동 | 방향키가 입력되면 블록인 좌우, 아래로 움직이거나 회전해야 한다. |
| 내려오는 블록 | 블록은 매 초 한 칸 씩 내려와야 한다. |
| 블록 쌓기 | 블록이 쌓인 블록과 닿아 더 이상 내려갈 수 없다면, 블록을 해당 위치에 쌓고 새 블록이 내려오기 시작해야한다. 쌓인 블록이 너무 높아져 더 이상 쌓을 수 없다면 게임을 종료해야 한다. |
| 블록이 모두 쌓인 줄 삭제 | 한 줄에 블록이 모두 쌓이면 해당 줄을 삭제하고, 줄 위의 블록을 한 칸 씩 내려야한다. |
| 점수 계산 | 줄이 삭제될 때마다 적절한 점수를 획득한다. 추가로, 블록이 쌓일 때, 바닥에 닿은 면의 크기만큼 점수를 추가한다. |
| 다음 블록 출력 | 게임 화면 옆에 다음에 내려올 블록을 미리 보여준다. |
| 그림자 | 현재 위치에서 블록이 떨어져 놓일 경우 놓일 위치를 보여준다. |

2주차의 랭킹 시스템에서는 랭킹을 기록하고 확인/삭제할 수 있도록 아래의 기능을 구현하여야 한다.

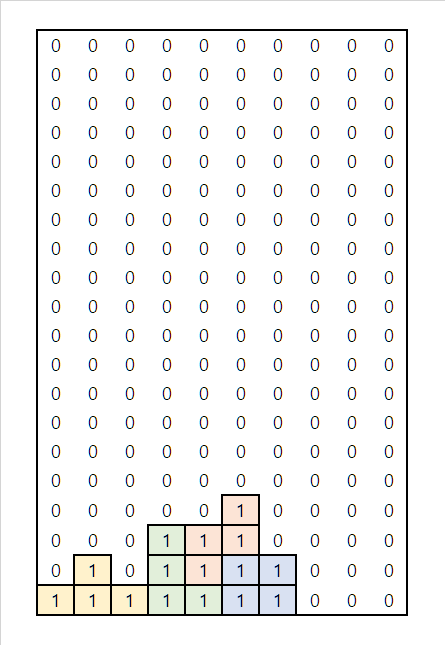
|  |  |
| --- | --- |
| 새 랭킹 등록 | 게임이 종료될 때마다 이름과 점수를 기록해야 한다. |
| 랭킹 검색 | 기록한 점수를 랭킹의 범위에 따라 검색해 출력한다. |
| 기록 삭제 | 저장한 점수를 삭제한다. |
| 프로그램 종료 시에도 유지되는 기록 | 프로그램이 종료되더라도 기록한 점수 데이터는 유지되어 다음 실행 때도 점수를 불러올 수 있어야한다. |

3주차의 추천 시스템에서는 다음 블록이 놓이면 좋을 위치를 사용자에게 추천한다. 이를 위해 다음 기능을 구현하여야 한다.

|  |  |
| --- | --- |
| 추천 위치 계산 | 현재 블록과 다음 블록을 통해 최대 점수를 얻을 수 있는 위치를 계산한다. |
| 추천 위치 출력 | 계산한 추천 위치를 테트리스 게임 화면에 출력한다. |
| 자동 플레이 기능 | 추천 위치에 따라 블록이 자동으로 쌓이며 게임이 진행되는 기능을 구현한다. |

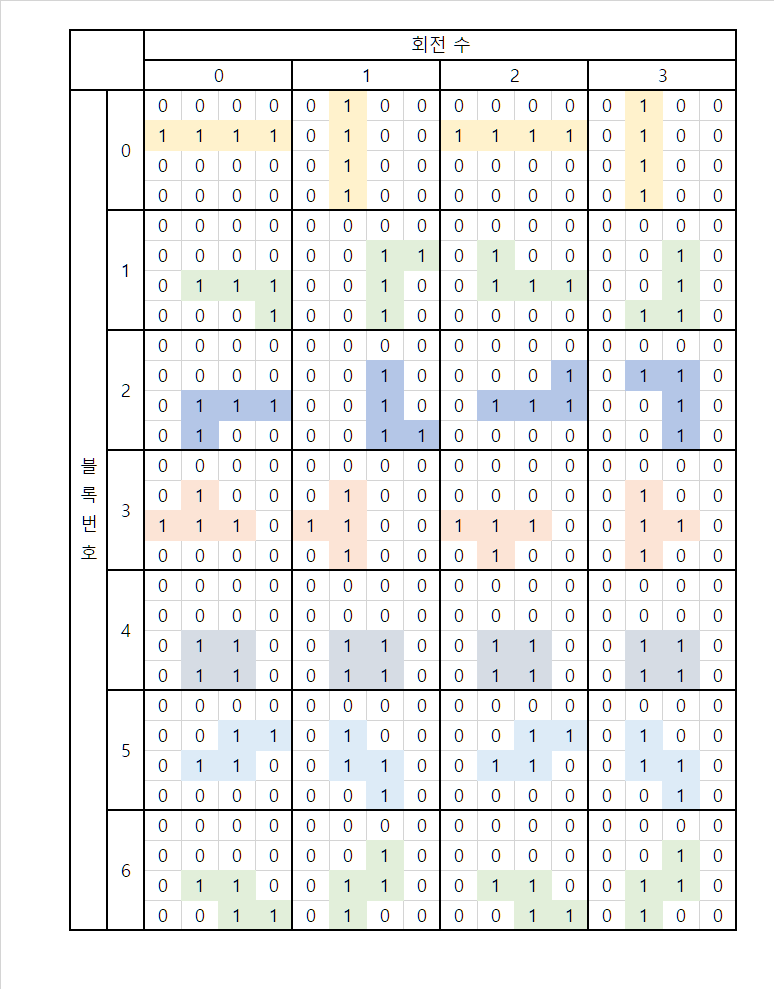
**2.2 합성**

테트리스 게임을 구현하기 위해, 테트리스의 필드와 블록 등의 정보를 저장해야 한다. 테트리스 필드를 구현하기 위해 문자로 이루어진 HEIGHT\*WIDTH의 2차원 배열을 사용하였다. 0은 블록이 쌓이지 않은 공간, 1은 블록이 쌓인 공간이다.



필드의 자료구조

블록은 4\*4의 배열로 나타낼 수 있다. 그리고 각 블록은 블록의 번호와 회전에 따라 다른 모양을 가진다. 총 4가지의 정보를 저장하기 위해 4차원 배열을 통해 블록을 구현하였다.



블록의 구조

테트리스 게임은 동시에 두가지 행동을 하여야 한다. 먼저, 매초마다 블록을 하나씩 떨어트린다. 블록이 바닥까지 떨어져 더 내려갈 수 없다면 블록을 쌓는다. 그리고, 필드의 한 줄이 모두 1인지 확인한 뒤, 1이라면 해당 줄을 지우고 블록을 한 칸 씩 내린다. 그리고, 새 블록을 떨어트린다. 동시에, 방향키와 q의 입력을 기다리며, 해당 입력이 있다면 입력에 맞는 커맨드를 실행해야 한다. 방향키가 입력되었다면 블록의 모양이나 위치를 바꾸고, q가 입력되었다면 게임을 종료해야 한다.

랭킹 시스템을 구현하기 위해, 사용자의 이름과 점수를 저장할 자료구조가 필요하다. 이를 위해 연결 리스트를 사용하였다. 연결 리스트는 기록의 개수가 프로그램 실행 중에 변하더라도 손쉽게 새 기록을 추가/삭제할 수 있다.

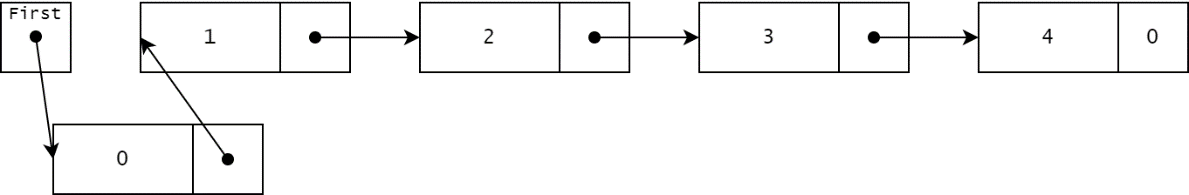
|  |
| --- |
| typedef struct \_ScoreNode {  int score;  char name [NAMELEN];  struct \_ScoreNode \*next;  } ScoreNode; |

기록을 저장하기 위한 구조체 ScoreNode

C:\Users\Tenny\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Untitled Diagram.png

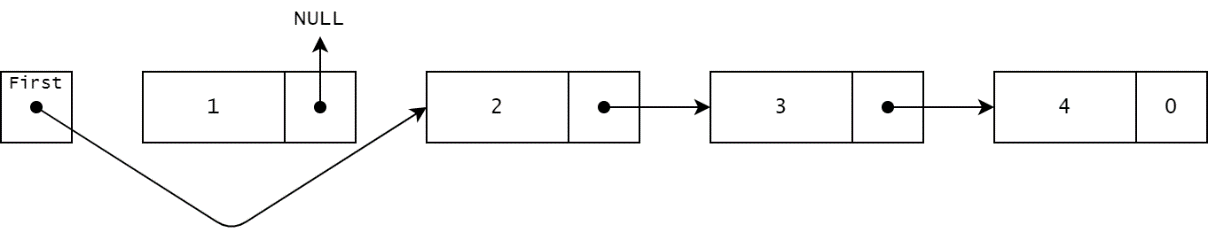
연결 리스트의 구조

연결 리스트는 리스트 중간에 원소를 손쉽게 삽입할 수 있다. 새로운 node를 삽입할 위치를 찾고, 앞의 node가 가리치는 link를 새 node에 연결한 뒤, 새로운 node의 link는 다음 node에 연결하여 node를 삽입한다. 맨 앞에 node를 추가하는 경우, 현재 연결 리스트를 임시 변수에 저장한 뒤, 새 node를 first로, first의 link를 임시 변수로 연결한다.



연결 리스트의 삽입

연결 리스트의 원소를 삭제할 경우, 삭제할 node의 앞 node의 link가 삭제할 node가 가리키는 node를 가리키게 한 뒤, 삭제할 node는 NULL을 가리키게 해 연결을 끊는다.

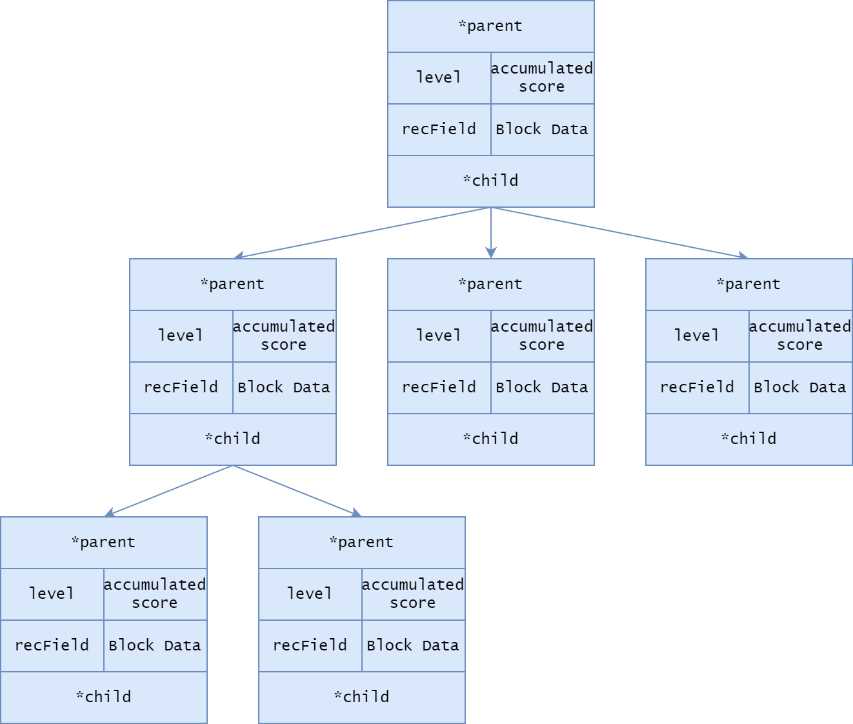


연결 리스트의 삭제

추천 시스템을 구현하기 위해, 현재 블록을 놓을 수 있는 모든 경우의 수를 계산하고, 경우의 수에 대해 다음 블록을 놓을 수 있는 모든 경우의 수를 계산해야 한다. 이를 위해, 트리 자료구조를 사용하였다. 각 노드는 최적의 위치를 계산하기 위해 필드 데이터, 현재 블록ID, 추천 위치, 깊이, 점수 등의 데이터를 저장해야 한다.

|  |
| --- |
| typedef struct \_RecNode{  int lv,score;  char f[HEIGHT][WIDTH];  struct \_RecNode \*c[CHILDREN\_MAX];  int curBlockID;  int recBlockX, recBlockY, recBlockRotate;  } RecNode; |

트리 노드 구조체



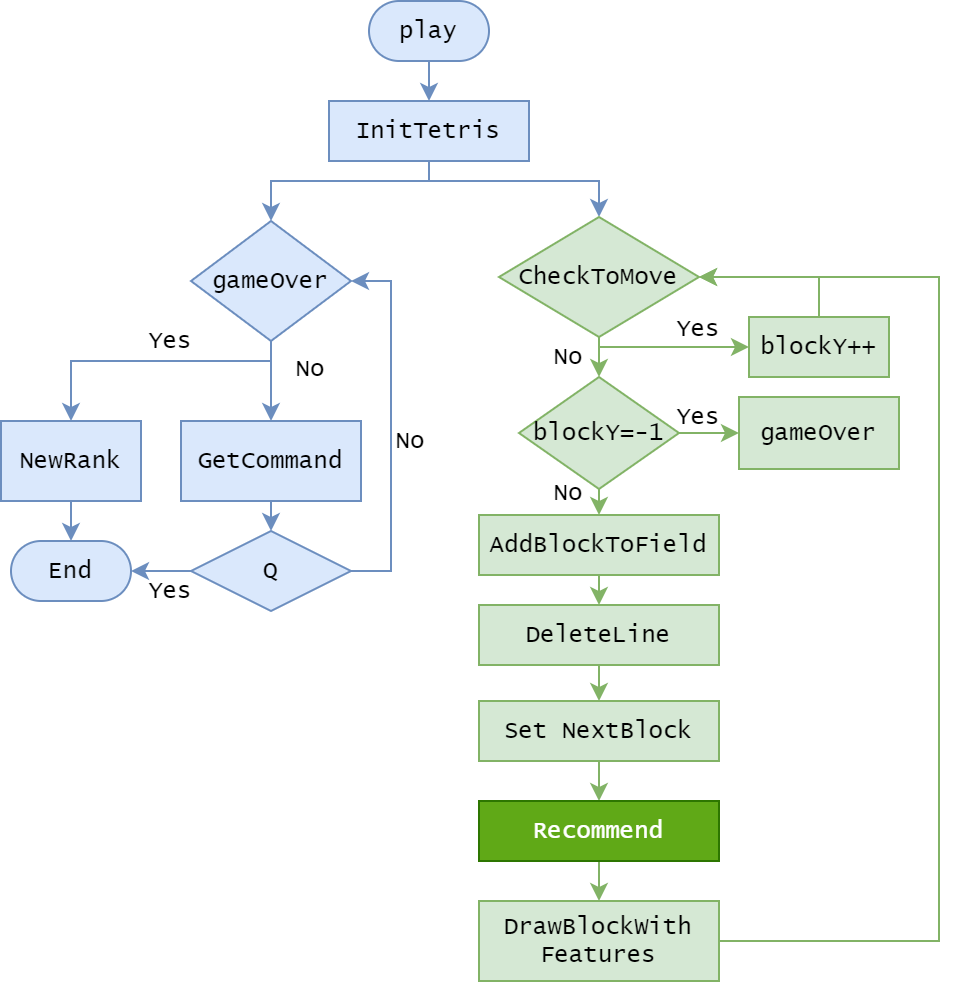
트리 자료구조

트리의 root node에서 현재 놓을 수 있는 블록의 경우를 계산해 child node로 추가한다. 그리고 각 child node에 대해 다음 블록이 놓일 수 있는 경우를 계산해 child node로 다시 추가한다. 이 과정을 탐색 깊이만큼 반복해 가장 높은 점수를 얻을 수 있는 경우를 추천한다.

트리의 탐색 깊이를 바꾸더라도 깊이에 따른 트리를 생성하기 위해 트리를 재귀적으로 생성한다. 트리의 한 level을 구성하는 함수를 함수 안에서 다시 호출하면, 해당 node에 대해 다음 level의 child node를 생성할 수 있다.

**2.3 분석**

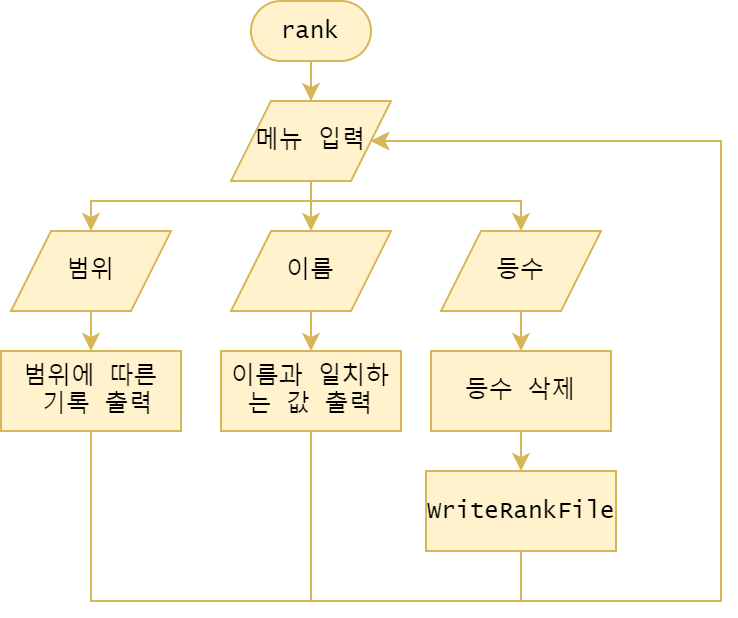
테트리스 게임은 다음의 순서로 진행된다. play 함수를 통해 전체적인 게임이 진행되며, 초록색 부분은 play 내부에서 매초 BlockDown 함수를 호출하며 실행된다.



할당한 함수는 다음과 같은 기능을 한다.

|  |  |
| --- | --- |
| InitTetris | 게임 시작을 위해 변수를 초기화하고 게임 화면을 그린다. |
| play | 게임을 진행한다. gameOver 신호를 받으면 점수를 저장하고 게임을 종료한다. |
| BlockDown | 매초 호출되며 블록을 한 칸 씩 내린다. 블록이 쌓이면 함수를 호출해 필요한 조치를 취한다. |
| CheckToMove | 블록이 특정 위치에 놓일 수 있는 지 확인한다. |
| AddBlockToField | 블록을 필드에 쌓고, 블록과 바닥이 닿는 면적에 따른 점수를 반환한다. |
| DeleteLine | 한 줄이 모두 쌓인 줄이 있는 지 확인하고, 있다면 줄을 지우고 쌓인 블록을 한 칸 내린다. 지워진 줄에 따라 점수를 반환한다. |
| Recommend | 현재 필드와 블록에 따라 추천 위치를 계산한다. |
| DrawBlockWithFeatures | 블록과 그림자, 추천 위치를 그린다. |

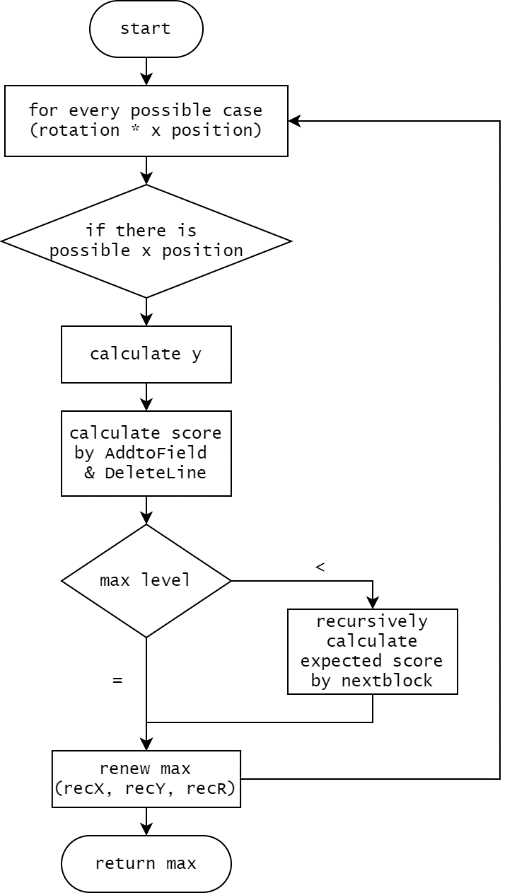
랭킹 시스템은 다음과 같은 구조로 이루어진다.



랭킹 시스템 구현을 위해 할당한 함수는 다음과 같다.

|  |  |
| --- | --- |
| rank | 랭킹 시스템을 총괄한다. 메뉴에 따라 해당하는 기능을 수행한다. |
| createRankList | rank.txt 파일을 읽어 연결 리스트로 저장한다. 프로그램이 시작되면 호출된다. |
| writeRankFile | 연결 리스트의 변경된 사항을 rank.txt에 다시 써 반영한다. |
| newRank | 게임이 종료되면 호출되어 새 기록을 작성한다. |

추천 시스템은 다음과 같은 알고리즘을 통해 실행된다. 이는 Recommend 함수를 통해 진행되며, 내부에서 재귀적으로 호출한다.



recommended play 기능은 play 함수와 유사하게 구현된다. 매초 추천 위치에 블록을 놓는 recBlockDown 함수를 호출한다. recBlockDown은 매초 블록이 한칸씩 내려오는 대신, 새 블록의 추천 위치를 계산해 해당 위치에 블록을 쌓는다. 마찬가지로 블록이 끝까지 쌓여 놓을 수 없다면 게임이 종료된다. 게임이 종료되면 추천 시스템의 시간/공간 효율성을 계산한 결과를 출력한다.

**2.4 제작**

위에서 설계한 내용을 C언어를 사용해 구현한다.

테트리스 게임

|  |  |
| --- | --- |
| int CheckToMove | char f[HEIGHT][WIDTH],int currentBlock,int blockRotate, int blockY, int blockX |
| 블록을 놓을 수 있으면 1, 아니면 0 |
| 입력 받은 위치에 블록을 놓을 수 있는 지 확인한다. 블록을 나타내는 4×4 행렬의 각 요소의 실제 필드상의 y 좌표가 HEIGHT보다 크거나 같거나, 실제 필드상의 x 좌표가 0보다 작거나 WIDTH보다 크거나, 블록을 놓으려고 하는 필드에 이미 블록이 쌓여져 있다면 놓을 수 없기 때문에 0을 반환한다. CheckToMove는 블록의 모든 요소를 검사하기 때문에 O(BLOCK\_HEIGHT\*BLOCK\_WIDTH)의 시간 복잡도를 가진다. |
| void DrawChange | char f[HEIGHT][WIDTH],int command,int currentBlock,int blockRotate, int blockY, int blockX |
|  |
| 입력한 커맨드를 인자로 받아 이전 블록의 정보를 찾는다. 이전 블록이 놓인 위치의 블록은 지우고 새 블록 정보에 따라 블록을 그린다. |
| void BlockDown | int sig |
|  |
| 매초 호출되며 블록이 움직일 수 있다면 한 칸 씩 내린다. 움직일 수 없다면, 블록을 필드에 그리고 채워진 줄을 비운다. 다음 블록의 값을 정하고 새 블록을 내리고, 변경된 사항을 그린다. blockY가 -1이라면 게임 종료를 선언한다. |
| int AddBlockToField | char f[HEIGHT][WIDTH],int currentBlock,int blockRotate, int blockY, int blockX |
| 바닥과 닿은 면적 \* 100의 점수 |
| 블록을 해당 위치에 쌓는다. (필드 값에 반영한다.) 블록의 모든 요소를 검사하며 1일 경우 블록을 그리기 때문에 O(BLOCK\_HEIGHT \* BLOCK\_WIDTH)의 시간 복잡도를 가진다. |
| int DeleteLine | char f[HEIGHT][WIDTH] |
| 지워진 줄 수의 제곱\*100의 점수 |
| 필드를 탐색하며 한 줄이 꽉 찬 구간이 있는 지 탐색한다. 한 줄 이 꽉 찬 구간이 있으면, 해당 구간을 지우고, 해당 구간으로 필드 값을 한 칸 씩 내린다. 필드를 모두 탐색하므로 O(HEIGHT \* WIDTH)의 시간 복잡도를 가진다. |
| void DrawShadow | int y, int x, int blockID,int blockRotate |
|  |
| 블록의 현재 위치에서 아래로 계속 내릴 경우의 Y 좌표를 계산한다. 그리고 해당 위치에 /로 블록을 그려 블록이 놓일 위치를 표시한다. |
| void DrawBlocksWithFeatures | int y, int x, int blockID, int blockRotate |
|  |
| DrawRecommend, DrawShadow, DrawBlock을 호출해 블록을 그린다. |

랭킹 시스템

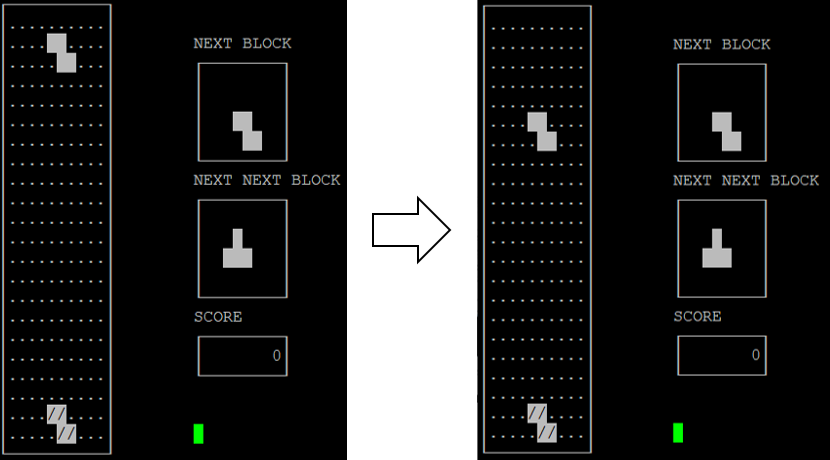
|  |  |
| --- | --- |
| void createRankList |  |
|  |
| rank.txt 파일을 열고, 파일의 내용을 연결 리스트에 저장한다. rank.txt 파일이 없다면 생성한다. 시간 복잡도는 O(NUMBER\_OF \_SCORES)이다. |
| void rank |  |
|  |
| 랭킹 시스템의 탐색/삭제 가능을 담당한다. 입력 받은 명령에 따라 범위 탐색, 이름으로 탐색, 기록 삭제의 기능을 수행한다. 1을 입력 받으면, 범위를 입력 받고 해당 등수의 이름과 기록을 출력한다. 이때, 시간 복잡도는 O(NUMBER\_OF \_SCORES)이다. 2를 입력 받으면, 이름을 입력 받고, 이름과 일치하는 기록의 정보를 출력한다. 마찬가지로 O(NUMBER\_OF \_SCORES)의 시간 복잡도를 가진다. 3을 입력 받으면 등수를 입력 받고 해당 등수의 기록을 삭제한다. O(NUMBER\_ OF\_ SCORES)의 시간 복잡도를 가진다. 기록이 삭제되어 rank.txt 파일의 변경이 필요할 경우 writeRankFile을 호출한다. |
| void writeRankFile |  |
|  |
| rank.txt 파일 열고 연결 리스트에 저장된 이름과 기록을 다시 작성한다. |
| void newRank | int score |
|  |
| 게임이 종료되면 호출된다. 새 기록과 이름을 받아 연결 리스트에 저장한다. 이때, 랭킹 순 출력을 원활하게 하기 위해 점수 내림차순으로 저장한다. 새 기록이 들어갈 위치를 탐색한 뒤 해당 위치에 삽입한다. 시간 복잡도는 O(NUMBER\_OF \_SCORES)이다. |

추천 시스템

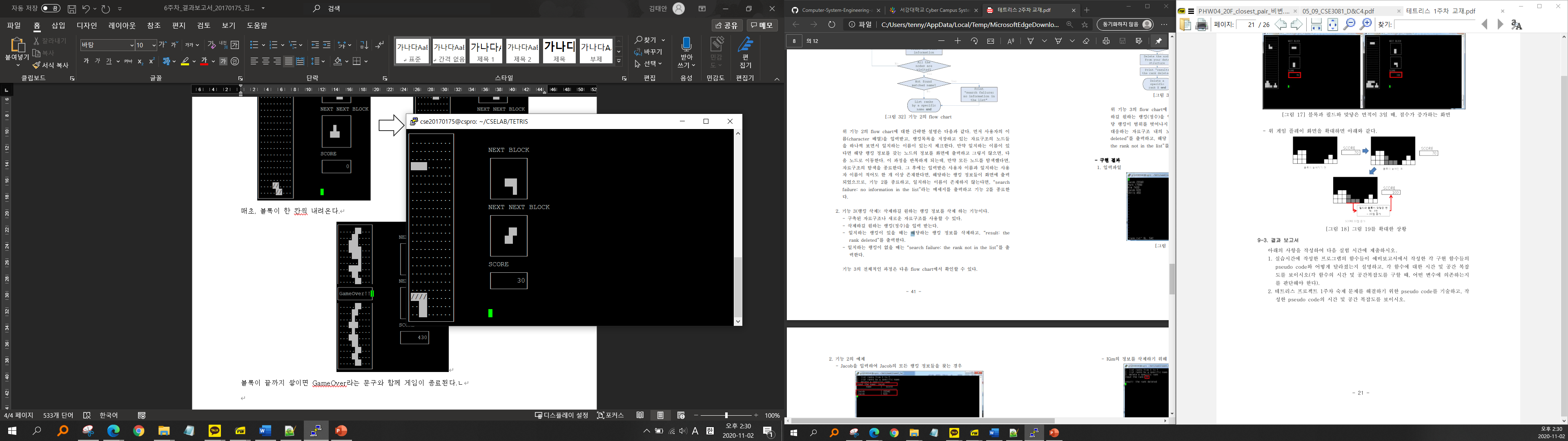
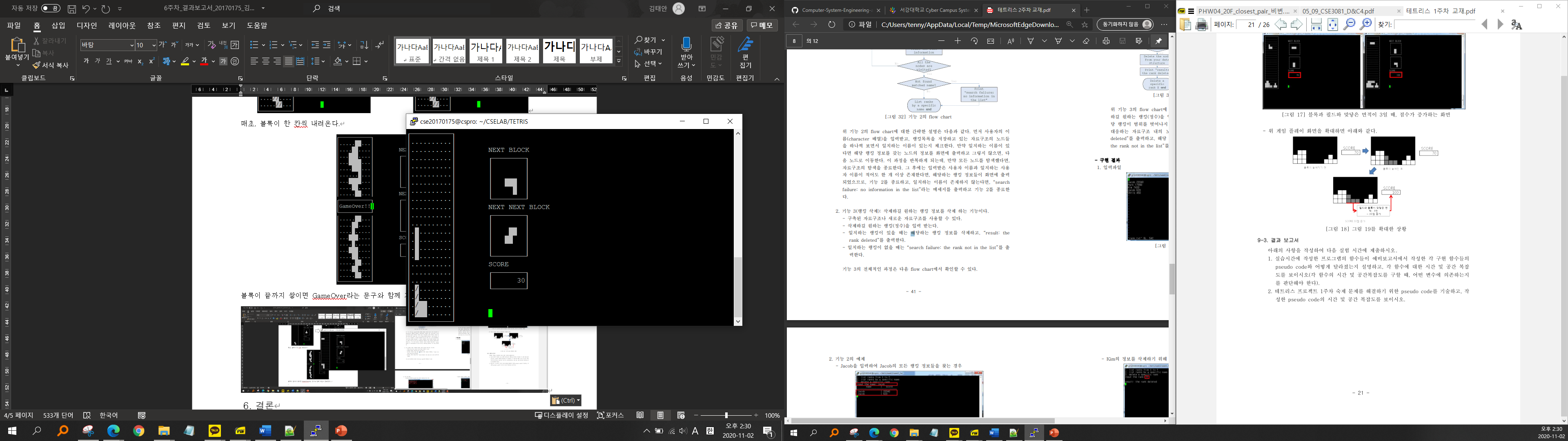
|  |  |
| --- | --- |
| void DrawRecommend | int y, int x, int blockID,int blockRotate |
|  |
| 해당 위치에 추천 블록 R을 그린다. |
| int recommend | RecNode \*root |
| 얻을 것을 기대되는 최댓값 max |
| 현재 노드에 대해 블록의 방향과 위치의 경우를 모두 고려한 Child Node를 생성한다. Child Node는 현재 블록이 놓였을 경우의 필드 값과 점수를 저장하고 있다. 현재 깊이가 최대 탐색 깊이가 아니라면 Child Node에 대해 재귀적으로 recommend를 호출하여 탐색한다. 이때, 공간 복잡도는 O((NUMBER\_OF\_ROTATION \* WIDTH)2)이다. |
| void recommendedPlay() |  |
|  |
| recommend에서 계산한 추천 위치를 바탕으로 자동으로 블록을 쌓는 기능이다. play 함수와 마찬가지로 매 시그널마다 recBlockDown을 호출한다. Q를 입력받거나 블록이 끝까지 쌓여 게임이 종료되면 recommend 함수의 분석 결과를 출력한다. recommend 함수는 time(t)와 space(t)를 기준으로 평가한다. time(t)는 score/게임 실행 시간으로 recommendPlay를 시작할 때부터 종료될 때까지 시간을 기록해 계산한다. space(t)는 recommend에서 사용한 RecNode에 따라 결정된다. space(t) = score/used memory for RecNode로 recommend 함수에서 RecNode를 생성할 때마다 RecNode의 크기만큼 값을 더해 계산한다. 이때, 사용되는 메모리를 바이트 단위로 계산할 경우 너무 큰 수이기 때문에 값을 1024로 나눈 킬로바이트 단위를 사용한다. |
| void recBlockDown | int sig |
|  |
| 매 초마다 호출되어 블록을 쌓는 함수이다. 블록을 쌓을 수 없는 경우 gameover를 선언한다. BlockDown의 알고리즘과 유사하나 커맨드를 입력 받고 해당 위치로 블록을 이동하는 단계를 생략한다. 대신, 계산된 추천 위치에 바로 블록을 추가하고, 한 줄이 쌓이면 해당 줄을 지운다. 그 후, nextBlock을 갱신하고, 추천 위치를 계산한다. |

**2.5 시험**

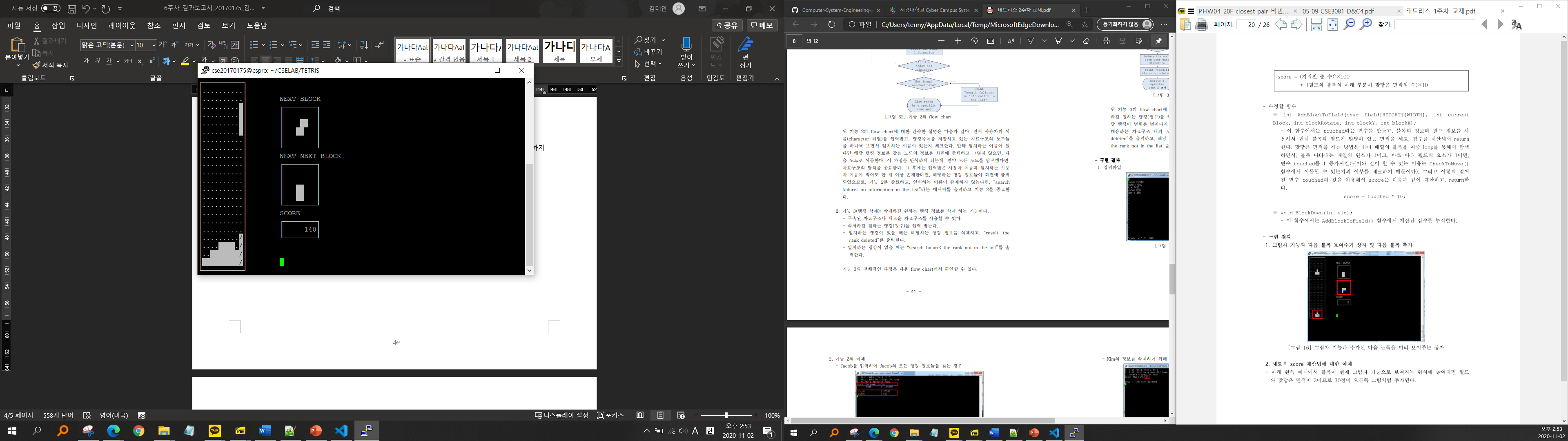
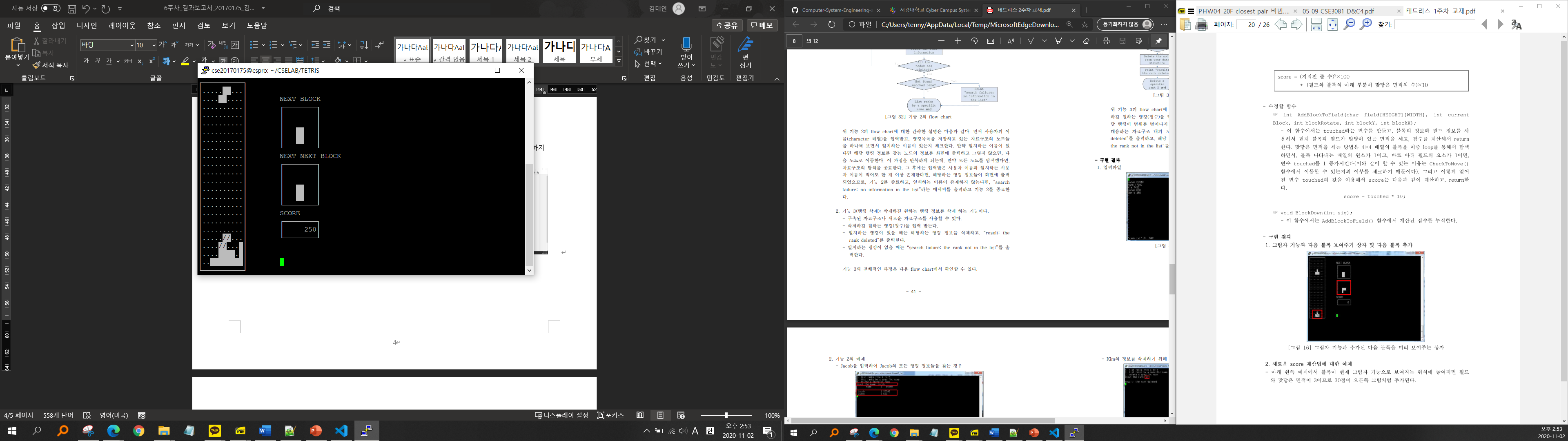
테트리스 게임



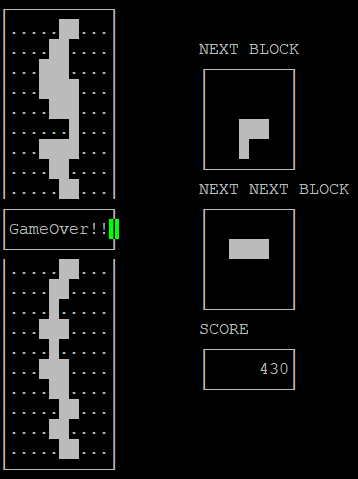
매초, 블록이 한 칸씩 내려온다.

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

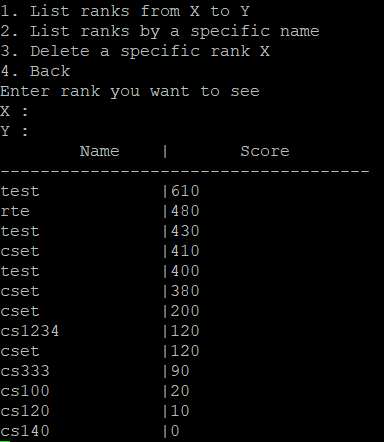
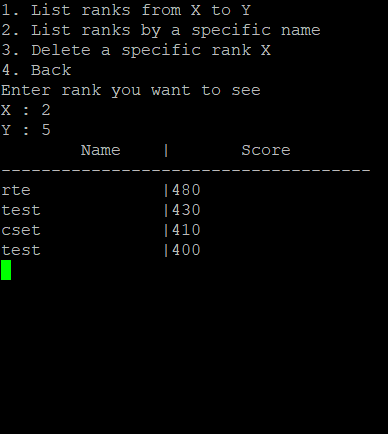
 

한 줄이 꽉 차면 줄이 삭제되며, 삭제된 줄 수2 X 100 + 블록의 밑이 닿은 면 수 X 10의 점수를 얻는다.

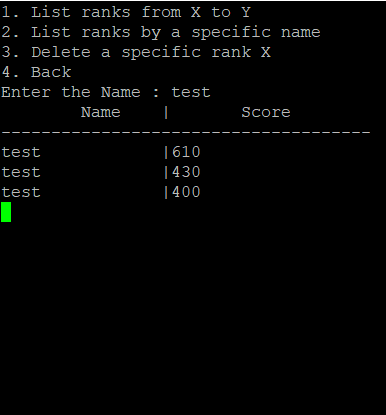


블록이 닿으면 블록이 쌓이며, 블록이 끝까지 쌓이면 게임이 종료된다.

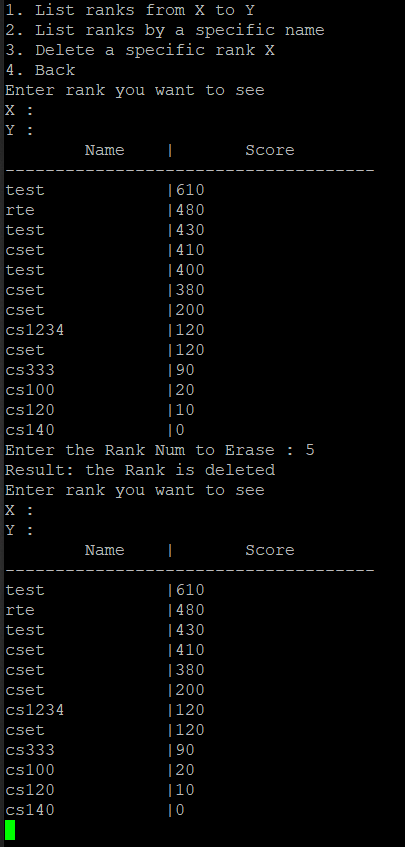
랭킹 시스템

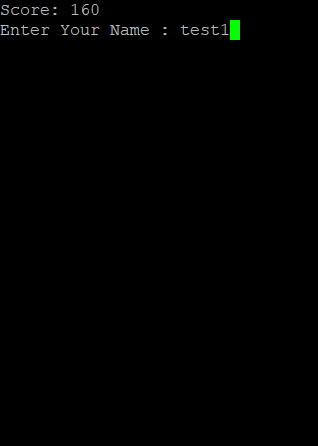
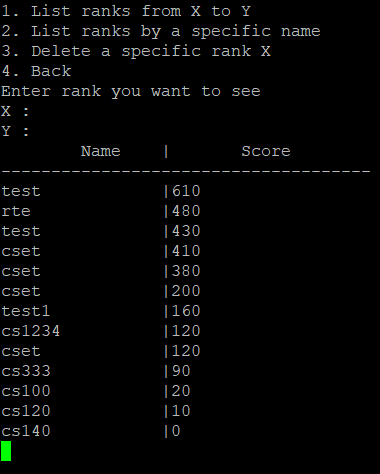
X, Y에 아무것도 입력하지 않으면 모든 rank를 볼 수 있다. 숫자를 넣어 보고 싶은 범위를 지정할 수 있다.



이름을 입력해 이름으로 결과를 탐색할 수 있다.

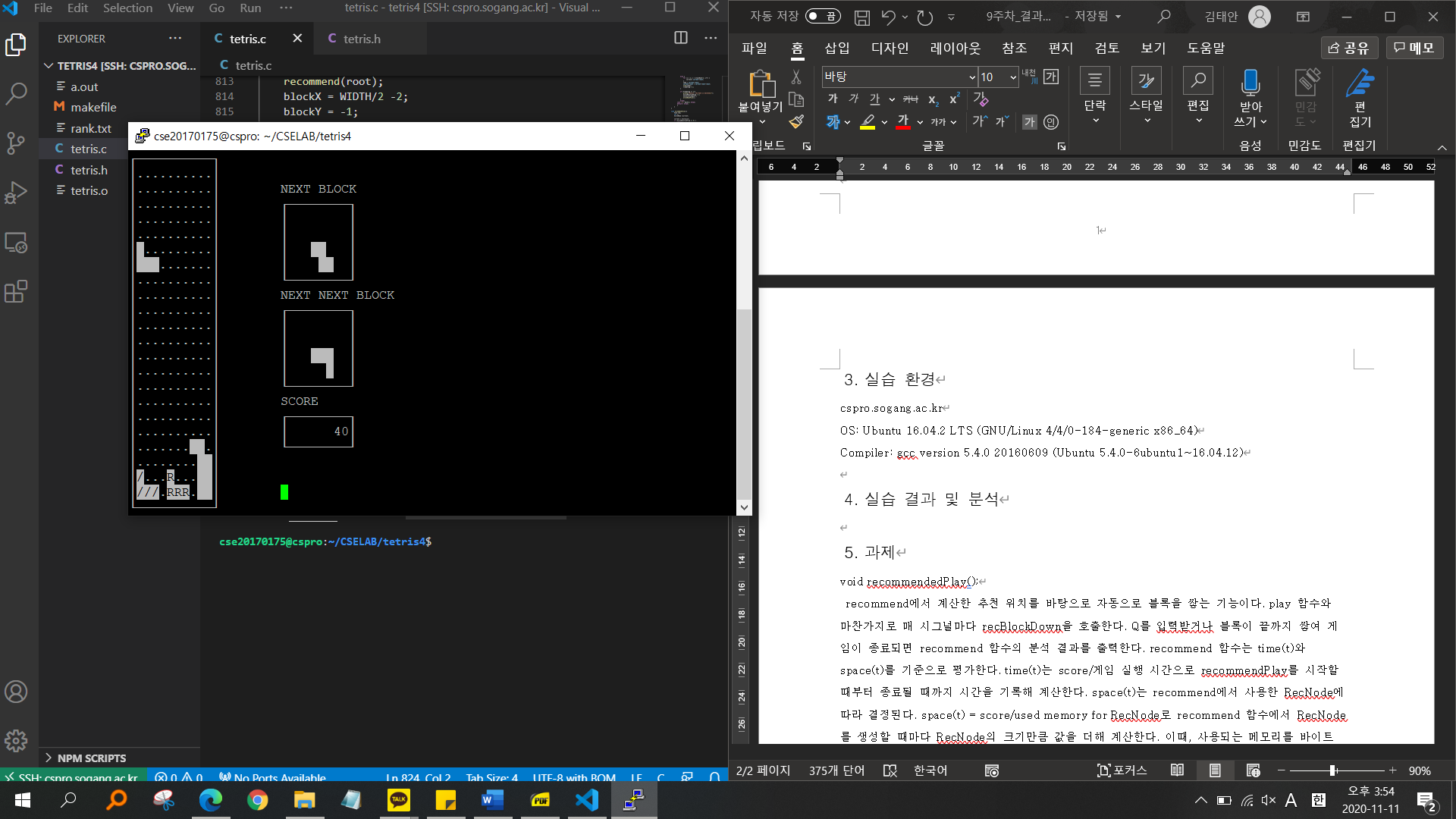
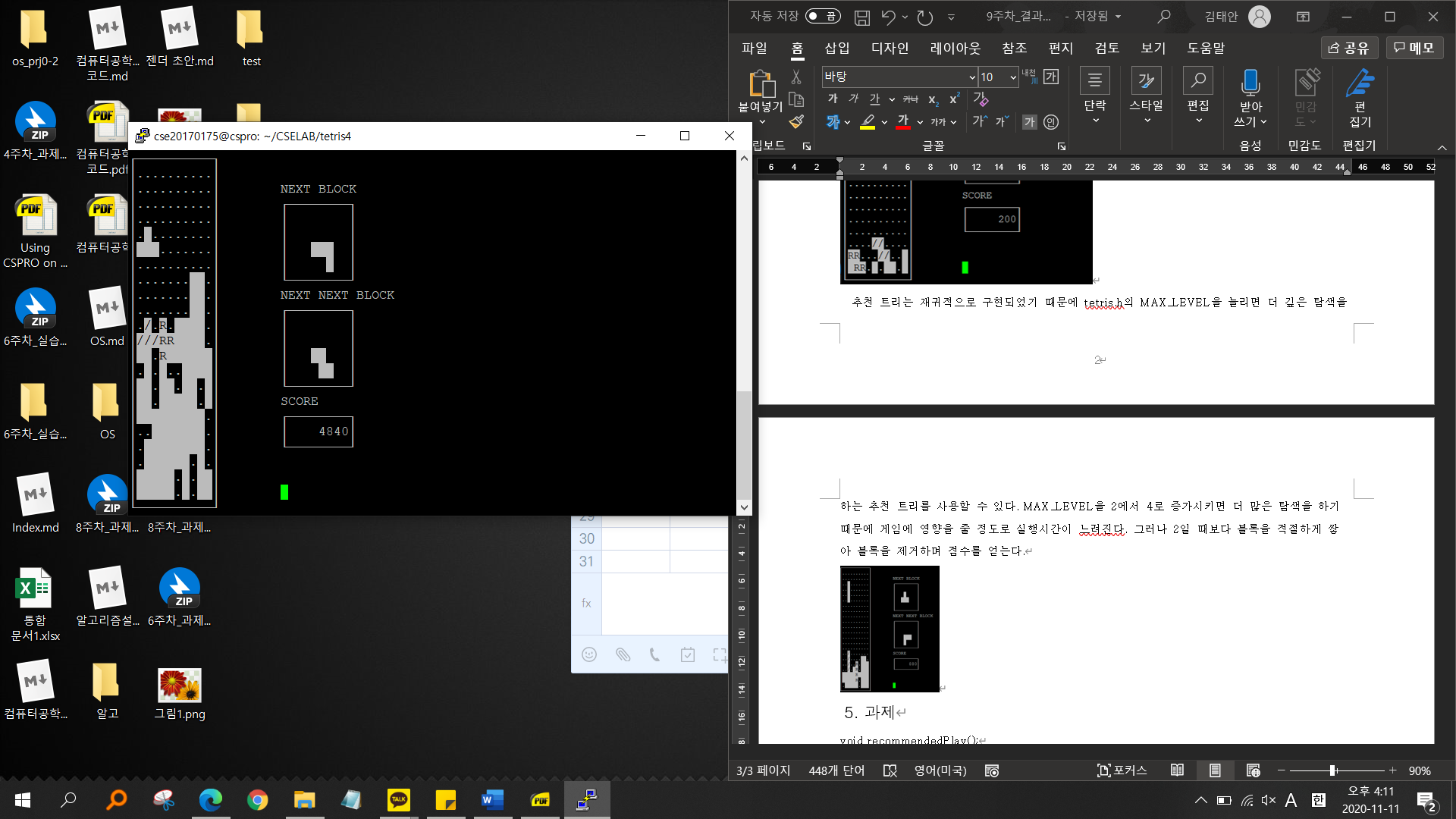


Delete 기능을 사용해 삭제하고 싶은 등수를 삭제할 수 있다.

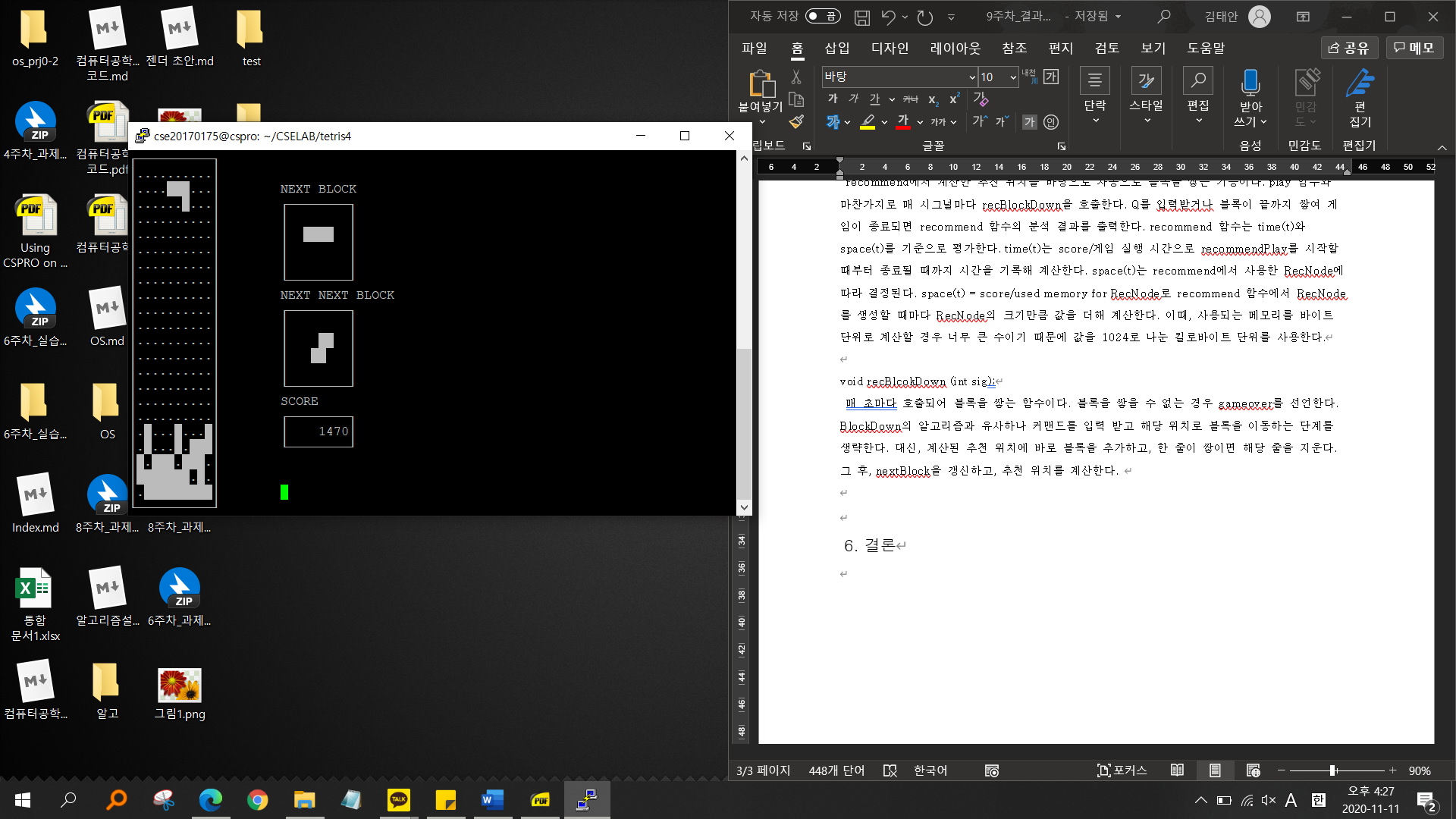
 

게임이 끝나면 왼쪽과 같은 화면이 나타나고, 이름을 입력해 점수를 등록할 수 있다.

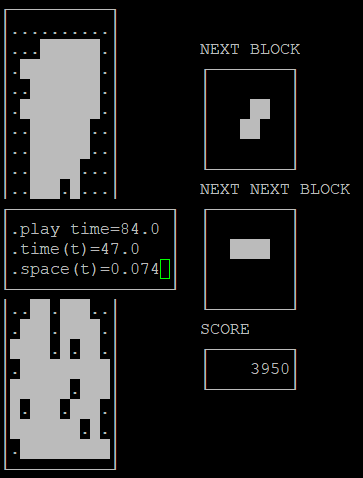
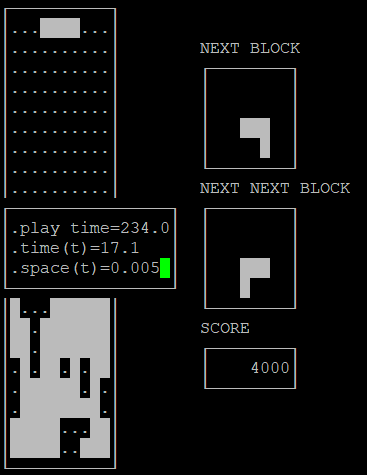
추천 시스템

추천하는 위치에 R 블록이 놓이며 추천한다. 트리의 탐색 깊이를 늘리면 더 많은 탐색을 하며 더 높은 점수를 기대할 수 있는 위치를 추천한다. 그러나 게임에 영향을 줄 정도로 실행 시간이 느려진다.



recommend play 모드를 실행하면 블록이 자동으로 추천 위치에 쌓인다.

실행을 종료하면 시간과 공간 사용량에 따른 점수를 보여준다. time(t)와 space(t)를 기준으로 평가한다. time(t)는 score/게임 실행 시간으로 recommendPlay를 시작할 때부터 종료될 때까지 시간을 기록해 계산한다. space(t)는 recommend에서 사용한 RecNode에 따라 결정된다. space(t) = score/used memory for RecNode로 recommend 함수에서 RecNode를 생성할 때마다 RecNode의 크기만큼 값을 더해 계산한다. 이때, 사용되는 메모리를 바이트 단위로 계산할 경우 너무 큰 수이기 때문에 값을 1024로 나눈 킬로바이트 단위를 사용한다.

**2.6 평가**

구현한 테트리스 프로그램은 이상 없이 테트리스 게임을 수행한다. 특히, 블록이 내려올 위치를 미리 알려주는 그림자 기능은 게임을 매우 수월하게 플레이할 수 있도록 돕는다. 랭킹 시스템의 경우, 사용 방법과 입출력이 조잡하다는 단점이 있다. 랭킹을 순서대로 보여주며 페이지를 넘기는 방식으로 구현했으면 하는 아쉬움이 있다. 전체적으로 일반 데스크톱에서 버벅임이 느껴지지 않을 정도의 프로그램이지만, 추천 시스템의 경우 확실히 프로그램이 느려지는 것이 느껴질 정도로 시간 복잡도가 크다. 더 나은 시간 복잡도를 가진 추천 시스템을 구현할 수 있는 방법을 고안해야 한다.

**2.7 환경**

학생들이 리눅스 서버를 접속하여 프로젝트를 진행하므로 해당 서버에 접속할 수 있는 데스크탑과 ssh 접속 프로그램을 제공한다. 접속하는 리눅스 서버에 각 학생들에게 하위 계정을 발급하여 할당 받는 용량에 한하여 자유롭게 이를 이용하여 프로젝트를 진행할 수 있는 환경을 제공한다.

**2.8 미학**

테트리스 프로그램은 매주 기능을 확장해가도 게임 프로그램의 모양은 변화하지 않는다. 이는 매주 추가한 각 기능은 메뉴의 1, 2, 3으로 분리되어 독립적으로 작동하기 때문이다. 또한, 주어진 프로그램 위에 새로운 기능을 하는 함수를 추가하는 방식으로 확장하여 기존 메뉴를 확장하여도 프로그램의 모양에 변화가 없다. 이때, DrawBlockFeatures와 같은 함수는 추가된 블록 출력을 총괄하며 일관성 있는 프로그램 작성을 돕는다.

**2.9 보건 및 안정**

테트리스 프로그램은 에러 상황이 발생하지 않도록 설계되었다. 1주차의 경우, 키의 입력을 받는 GetCommand 함수는 게임에서 사용하는 키의 경우만 처리하며, 이외의 경우는 모두 NOTHING으로 처리해 사용하지 않는 키의 입력에 대한 행동이 없도록 하였다. 모든 블록의 움직임은 CheckToMove를 통해 놓일 수 있는 지 검사된다. CheckToMove 함수는 블록을 놓을 수 없는 모든 경우를 검토하며 잘못된 위치에 블록이 놓이는 상황을 방지하였다.

랭킹 시스템의 경우, 랭킹의 검색과 삭제에 있어 논리적으로 불가능한 입력이 존재한다. 검색의 시작은 검색의 끝보다 클 수 없으며, 시작과 끝은 모두 0보다 크고 기록의 개수보다 작거나 같아야 한다. 삭제의 경우도 마찬가지로 기록의 개수 범위의 수를 입력 받아야한다. 범위에 벗어난 입력에 대해 에러 메시지를 출력하여 예외 처리를 하였고, 이를 통해 존재하지 않는 값에 접근하지 못하도록 하였다. 또한, rank.txt가 존재하지 않는 경우, rank.txt 파일을 생성해 파일을 읽고 쓸 때 오류가 발생하지 않도록 하였다.

에러가 발생하지 않도록 많은 노력을 하였으나 몇몇 특수한 상황에서 에러가 발생하는 것을 목격하였고, 발견된 오류는 다음과 같다.

1. 테트리스 추천 플레이 기능을 오랜 시간 사용할 경우 프로그램이 죽는 경우가 발생한 적이 있다.

2. VS Code에서 ssh를 통해 서버에 접속한 뒤, 테트리스 프로그램을 실행 중에 터미널 창의 크기를 조정하면 에러가 발생한다.

**3. 기 타**

**3.1 환경 구성**

운영체제: 리눅스 우분투

컴파일러: gcc

디버깅 툴: gdb

텍스트 에디터: vs code

주요 라이브러리: ncurses

**3.2 참고 사항**

참고사항 없음

**3.3 팀 구성**

김태안 (20170175, 100%)

**3.4 수행기간**

2020년 10월 13일부터 2020년 11월 11일까지