프로그래밍 프로젝트(Programming Project)

목 적 : 본 단원에서는 테트리스 게임 및 미로 프로그램을 구현하면서 자료구조 강의에서 배운 다양한 자료구조들을 다루어 봄으로써, 자료구조에 대한 이해도를 높이기 위한 프로젝트를 수행한다. 이 프로젝트를 통하여 학생들로 하여금 주어진 문제를 해결하기 위한 적절한 자료구조를 선택하고, 이를 바탕으로 하는 효율적인 프로그램을 작성하는 능력을 갖도록 한다.

목차

실험 PRJ-1 테트리스 1주차/3주

1. 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.

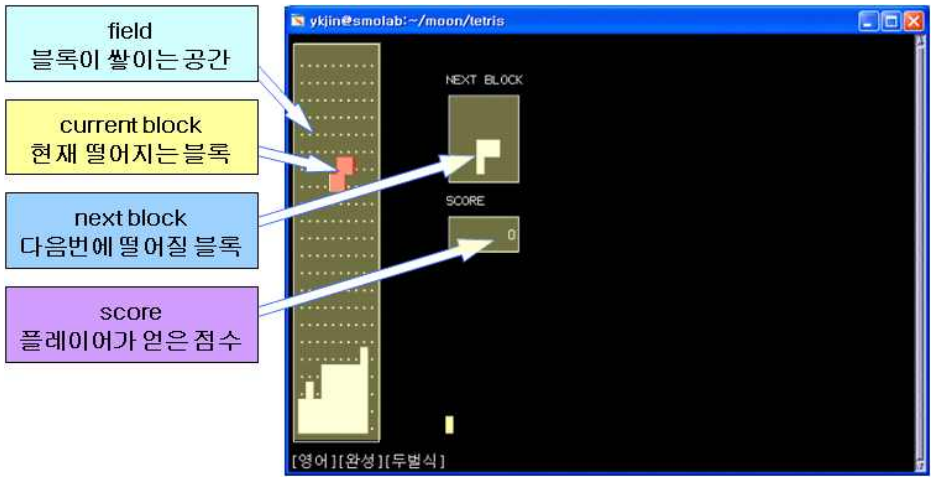
2. 테트리스란?

테트리스는 알렉스 파지노프가 1984년에 개발한 게임으로 그리스 숫자 접두어인 tetra와 파지노프가 좋아하던 테니스를 합쳐 만들어진 이름이다. 전 세계적으로 유명한 이 게임은 직사각형의 빈 공간에 7가지 모양의 블록을 쌓아 빈칸이 없게 되면, 사라지면서 점수를 얻는 퍼즐형 게임으로, 블록은 다음과 같은 특성을 갖는다.

- 한 번에 시계 반대 방향으로 90도씩 회전할 수 있다.
- 위 방향을 제외한 모든 방향으로 이동할 수 있다.
- 일정 시간마다 블록이 떨어진다.

이 테트리스 게임은 비디오 게임기와 컴퓨터 어떤 곳에서도 가동되고, 그 응용이 다양한데, 디자인 목적으로 빌딩, 조형물 등에 응용되어 사용되기도 한다. 그리고 실제 이 게임은 2005년부터 2010년 1월까지 휴대폰 게임 다운 횟수가 10억 회가 넘었을 만큼 여전히 대중들에게 친숙하고, 인기 있는 게임이다.

3. 테트리스 게임 화면의 구성 및 각 구성 요소



[그림 1] 테트리스 게임 화면의 구성요소

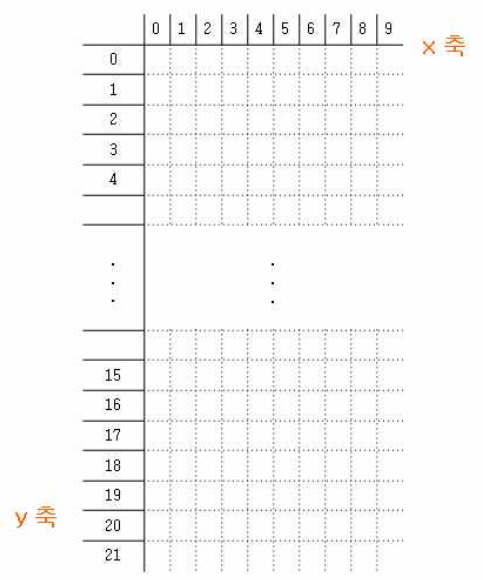
1) 필드(field) : 테트리스 프로그램에서 게임이 수행되고, 블록이 쌓이는 공간이다. 게임을 플레이(play)하는 사용자는 필드에서 테트리스 게임 진행 상태를 확인할 수 있다.

2) 현재 블록(current block) : 현재 떨어지는 블록으로, 사용자는 키보드 입력을 통해서 블록의 위치와 모양을 조절할 수 있다.

3) 다음 블록(next block) : 다음에 필드에서 떨어지게 될 블록의 모양을 확인할 수 있다. 만약 현재 블록이 필드에 쌓이게 되면, 이 블록은 필드의 상단에 위치하여 떨어지게 된다.

4) 점수(score) : 테트리스 게임을 플레이하는 사용자가 얻은 누적 점수를 보여주는 곳이다.

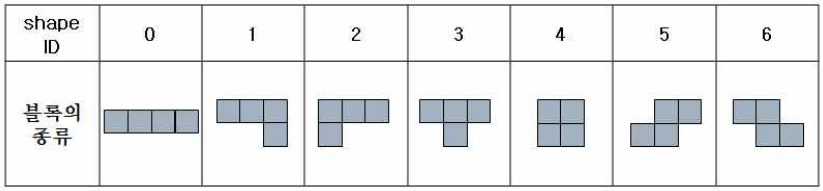
3-1 필드(field)



[그림 2] 테트리스 필드

- 테트리스 블록이 쌓이는 공간인 필드의 구조는 위와 같이 가로는 10, 세로는 22의 크기를 갖는다. 위의 구조는 GUI를 프로그래밍 환경을 지원하기 위한 ncurses 라이브러리를 따른다(ncurses 라이브러리에 자세한 사항은 부록1을 참조한다).
- 기존에 알고 있던 좌표 (x, y)와는 달리 테트리스 프로그램에서는 (y, x)로 좌표를 표시한다. 왼쪽 상단을 기준으로 하며, 그 좌표는 (0,0)이다.

3-2 블록(block)



[그림 3] 테트리스에서 사용되는 7가지 블록

위와 같이 테트리스 게임에서 고려되는 블록은 총 7가지이다. 이 블록은 다음과 같은 특징을 갖는다.

- 블록은 세로로 가장 길 때의 길이가 4, 가로로 가장 길 때의 길이가 4이므로, 4×4 배열을 사용해서 표현한다. 자세한 표현은 frame 코드를 참조한다.
- 블록은 총 4번의 회전수를 갖고, 한 번 회전할 때마다 시계 반대 방향으로 90도씩 회전한다.
  0 : 0도 회전
  1 : 시계 반대 방향으로 90도 회전
  2 : 시계 반대 방향으로 180도 회전
  3 : 시계 반대 방향으로 270도 회전

4. 테트리스 게임 진행 및 동작 방법

4-1 게임 진행

1. 현재 블록과 다음 블록이 화면상에 나타난다.
2. 현재 블록을 조작(키보드를 사용한 이동 및 회전)하여 원하는 위치에 놓는다.
3. 현재 블록이 필드에 쌓여, 빈 칸이 없는 줄이 생기면 그 줄은 지워지고, 다음 블록이 필드에 나타난다.
4. 지워진 줄의 수에 따라 점수가 계산되고, 증가한다.

4-2 게임 동작

1. 키보드 ← : 블록을 좌로 이동한다. 더 이상 이동할 수 없을 때는 이동하지 않는다.
2. 키보드 → : 블록을 우로 이동한다. 더 이상 이동할 수 없을 때는 이동하지 않는다.
3. 키보드 ↑ : 블록을 회전시킨다. 한 번 누를 때마다, 시계 반대방향으로 90도씩 회전한다.
4. 키보드 ↓ : 블록을 아래로 이동한다. 더 이상 이동할 수 없을 때는 블록이 필드에 쌓이게 된다.
5. 키보드 'Q' or 'q' : 현재 플레이하고 있는 테트리스를 강제로 종료시키고, “Good-bye!!”라는 메시지를 화면에 출력한다.

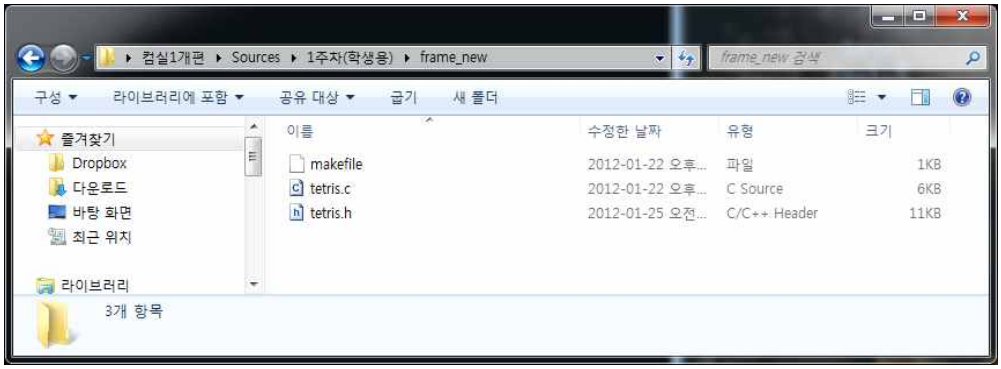
4-3 게임 종료

블록이 테트리스 게임을 수행하는 필드에 쌓여, 더 이상 쌓을 공간이 없을 때, 게임이 종료된다. 게임을 종료하기 전에 화면에 “GameOver!!”라는 메시지가 출력된다.

5. 테트리스 frame 프로그램 실행

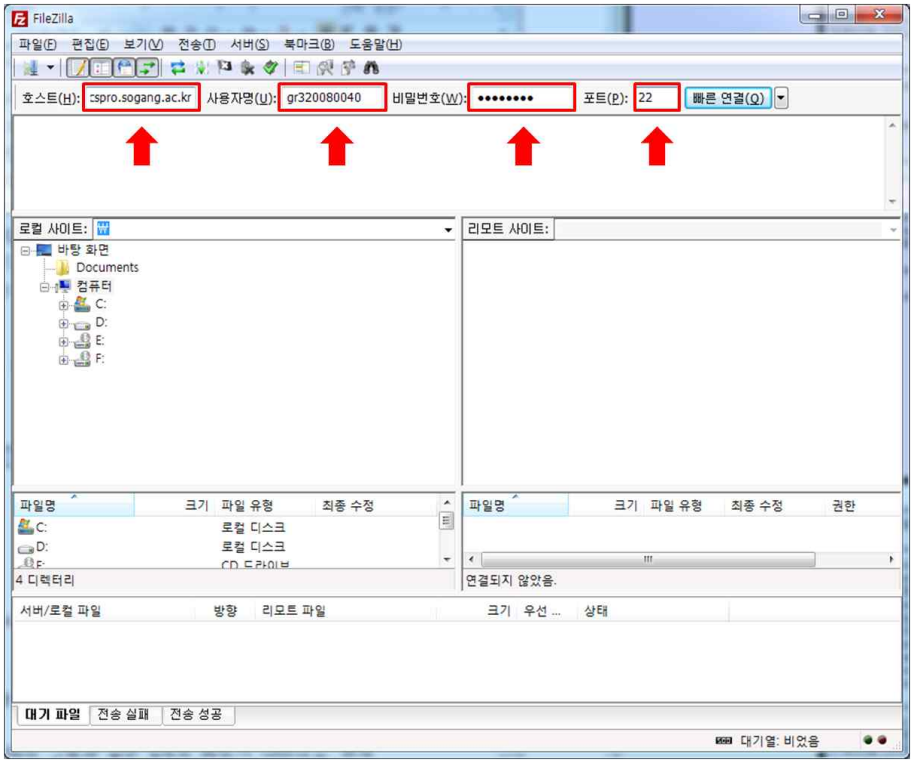
먼저 server에 접속하여 프로그램을 수행시키기 위해서 “putty 한글판”을 다운로드하여, 설치하고, frame.zip(or frame.tar)을 다운로드 받는다. 다운로드 받은 파일의 압축을 풀어서 다음 3가지 파일이 존재하는지 확인한다.

- tetris.c(테트리스 게임 source 파일)
- tetris.h(테트리스 게임 header 파일)
- makefile(테트리스 게임의 실행파일을 생성하기 위한 파일)



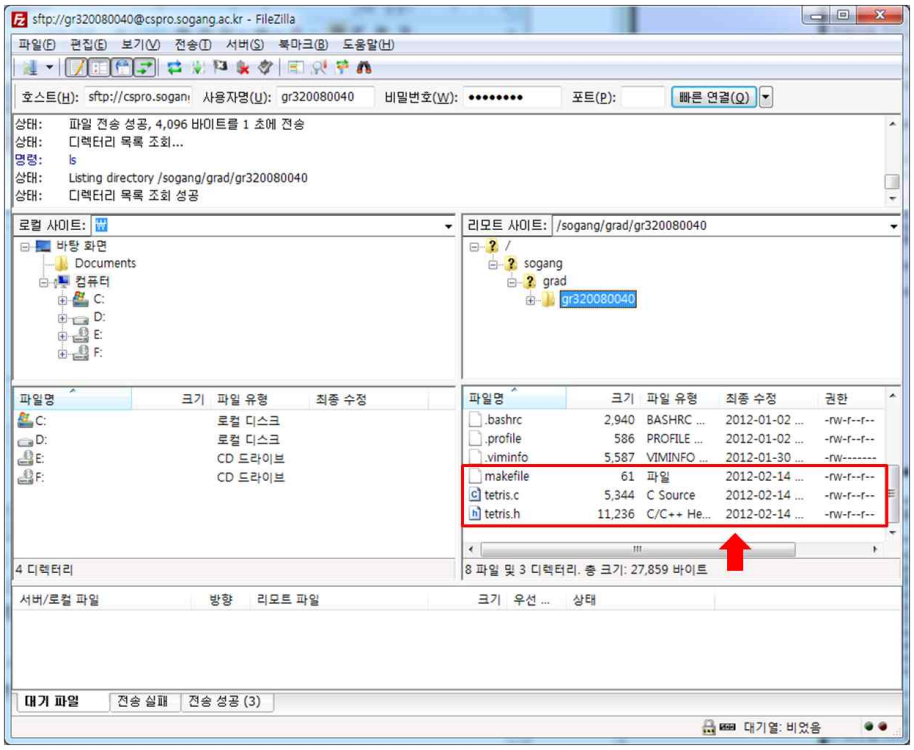
[그림 4] frame 프로그램의 3가지 파일

이 파일들을 자신의 계정(cs+학번)에 업로드(upload)하기 위해서 FileZilla 등 파일 전송을 지원하는 프로그램을 다운로드한다. 여기서는 FileZilla를 사용한다. 위 3가지 파일들을 업로드 하기 위해, FileZilla에 호스트(cspro.sogang.ac.kr), 사용자명(cs+학번), 비밀번호, 포트(22)를 입력하고, 빠른 연결을 클릭하여, cspro machine에 접속한다.



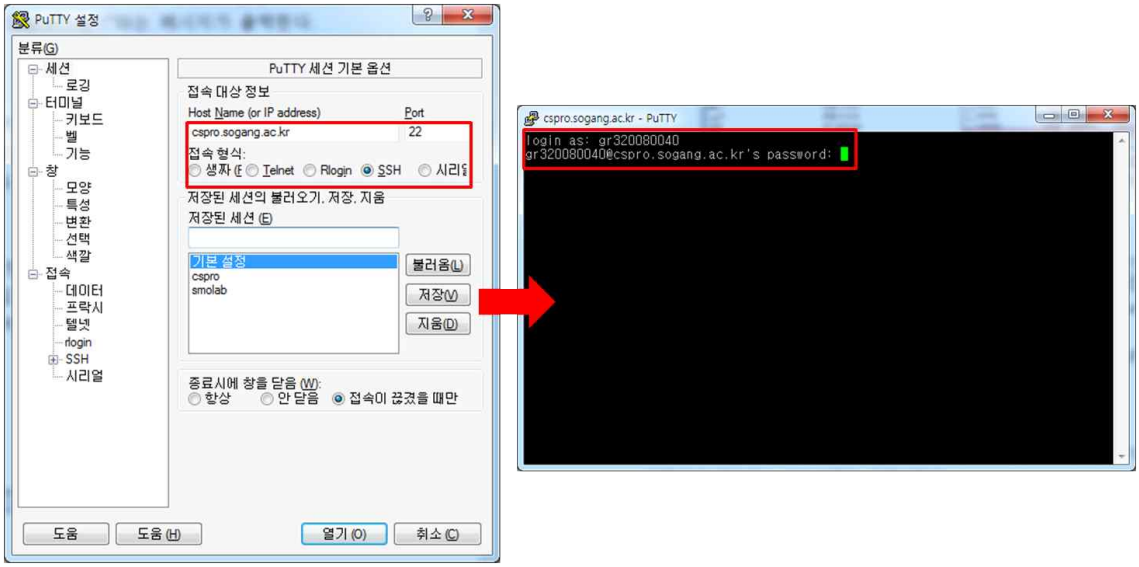
[그림 5] FileZilla 접속화면

Frame 프로그램을 구성하는 3가지 파일들을 업로드 하기 위해 파일을 마우스 등을 이용해 선택한 후, FileZilla의 오른쪽 아래 창에 드래그 앤 드롭(drag & drop)한다.



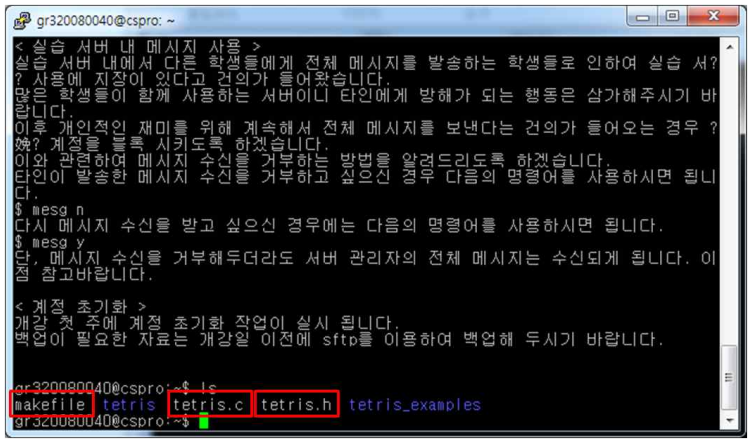
[그림 6] FileZilla를 이용해서 3개의 파일을 업로드(upload)한 상황

이제 putty를 이용해 cspro machine에 접속한다. 이 때, Host_Name(or IP address)은 “cspro.sogang.ac.kr”이고, 접속형식은 SSH, Port는 22이다.



[그림 7] Putty를 이용한 접속화면

만약 csrpo machine에 잘 접속이 되었다면, ‘ls’를 입력하여 위에서 확인한 3가지 파일인 tetris,c, tetris.h, makefile이 모두 잘 업로드 되어 있는지 확인한다. 만약, 새로운 폴더를 만들어서 cspro machine에 업로드하고 싶다면, FileZilla의 오른쪽 아래 창에서 새로운 폴더를 만들고자 하는 폴더(상위 폴더)를 선택하고, 마우스 오른쪽 버튼을 클릭해서 나오는 팝업 메뉴를 통해 새 폴더를 만들어, 그곳에 파일을 업로드 하면 된다.

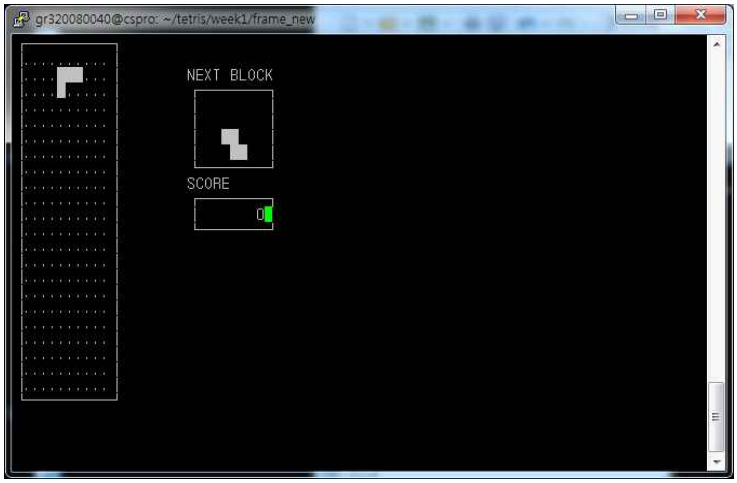


[그림 8] frame의 3가지 파일들 확인

이렇게 3가지 파일들이 모두 업로드 되어 있다면, “**make**”를 입력하여 테트리스 프로그램의 실행파일인 “a.out”을 생성한 후, “**./a.out**”을 입력하여 테트리스 프로그램을 실행시킨다. Frame 프로그램을 처음 실행시키면 아래의 그림과 같은 4개의 메뉴가 나타나는데, 현재 동작하는 메뉴는 1번 play와 4번 exit뿐이다.

[그림 9] frame의 수행 후 보여지는 메뉴

1번 play를 누르고, 게임을 수행하면, 테트리스 프로그램의 기본 기능이 구현되지 않았기 때문에, 다음과 같은 정지화면만이 화면상에 나타난다.



[그림 10] play 메뉴 수행화면

6. Frame 프로그램의 전역 변수와 제공된 함수들

6-1. 전역 변수

☞ **char field[HEIGHT][WIDTH]**

필드 정보를 저장하고 있는 배열이다. 이 필드 정보는 0, 1로 나타내는데, 각각은 다음을 의미한다.

▶ 0 : 필드가 채워지지 않음을 나타내고, 화면에 '.'로 표시된다.

▶ 1 : 필드가 채워져 있다는 것을 나타내고, 공백 ' '을 attron(), attroff()를 사용해서 화면에 표시한다.

HEIGHT는 앞의 필드 설명과 같이 22로, WIDTH는 10으로 tetris.h에 define되어 있다.

☞ **int nextBlock[BLOCK_NUM];**

현재 블록과 다음 블록의 ID를 저장하고 있는 변수로, 배열의 첫 번째 원소(nextBlock[0])는 현재 블록의 ID를 저장하고 있고, 다음에 이어지는 원소들은 다음 블록들의 ID들을 차례로 저장하고 있다. 초기에는 BLOCK_NUM이 2로 define되어 있으므로, 현재 블록과 다음 블록 1개의 ID가 저장된다. 더 많은 다음 블록을 미리 생성하고자 한다면, BLOCK_NUM을 수정하고, 이에 따른 프로그램의 변경사항을 수정하면 된다. 이 때, 다음 블록의 ID는 rand() 함수를 사용해 0~6사이의 수를 임의로 생성된다(rand()%7).

☞ **int blockRotate**

현재 블록이 얼마나 회전 했는지에 대한 정보를 저장하고 있는 변수이다. 총 0~3까지 4가지 회전수를 갖는다. 가장 먼저, 0은 0도 회전, 1은 시계 반대 방향으로 90도 회전, 2는 시계 반대 방향으로 180도 회전, 3은 시계 반대 방향으로 270도 회전을 나타낸다.

☞ **int blockY**

현재 블록의 필드상에서의 y 좌표 값을 저장하는 변수이다. 현재 블록이 새로운 블록으로(미리 보기를 통해 확인할 수 있는 다음 블록) 갱신될 때, 이 값은 -1로 초기화된다.

☞ **int blockX**

현재 블록의 필드상에서의 x 좌표 값을 저장하는 변수이다. 현재 블록이 새로운 블록으로(미리 보기를 통해 확인할 수 있는 다음 블록) 갱신될 때, 이 값은 필드의 가운데에 위치시키기 위해서 (WIDTH/2)-2로 초기화된다.

☞ **int score**

사용자가 플레이하여 얻은 점수를 저장하는 변수이다. 이 변수는 테트리스 게임 시작할 때, 0으로 초기화 된다.

☞ **int gameOver**

테트리스 게임이 종료 되었는지의 여부를 나타내기 위한 변수로, 게임을 시작할 때 0으로 초기화되고, 게임이 종료되는 순간 1로 그 값을 변경하게 된다.

☞ **int timed_out**

현재 블록을 일정 시간 마다 한 칸씩 떨어뜨리기 위해 사용되는 변수로 0으로 초기화한다.

6-2 제공된 함수들

테트리스 프로젝트를 위해 frame 프로그램에서 기본적으로 제공된 함수들은 다음과 같다.

☞ **void InitTetris()**

- Input : 없음
- Output : 없음
- 테트리스 게임을 수행하기 위해 기본적인 변수와 자료구조를 초기화하는 함수이다. 가장 먼저, 테트리스 필드를 모두, 채워지지 않은 상태를 나타내는 '0'으로 초기화하고, 현재 블록과 다음 블록의 ID를 임의로 생성하여 결정한다(`rand()%7`). 그리고 현재 블록의 회전수는 0, 블록의 위치는 가장 위의 가운데에 위치하므로, x 좌표는 `(WIDTH/2)-2`, y 좌표는 -1로 초기화한다. 점수를 나타내는 `score` 변수, 게임이 종료되었는지 나타내는 `gameover` 변수는 0으로 초기화하고, `BlockDown()` 함수에서 사용되는 `timed_out`도 0으로 초기화 한다. 마지막으로 이렇게 결정된 정보를 바탕으로 테트리스 초기화면을 그려준다. 초기화면을 그리는 과정에서는 `Draw Outline()`, `DrawField()`, `DrawBlock()`, `DrawNextBlock()`, `PrintScore()`를 차례로 호출하게 된다.

☞ **void DrawOutline()**

- Input : 없음
- Output : 없음
- 블록이 떨어지는 테트리스 필드의 테두리, 다음 블록이 미리 그려질 상자의 테두리, 점수를 보여줄 상자의 테두리를 그리는 함수이다.

☞ **int GetCommand()**

- Input : 없음
- Output : 사용자가 입력한 명령(`command` 변수)
- 사용자의 입력을 기다리다, 사용자가 입력을 하게 되면, 입력을 구분하여 입력받은 명령을 return하게 된다. 구분되는 입력은 총 5가지이다.
  - ▶ KEY_UP
  - ▶ KEY_DOWN
  - ▶ KEY_LEFT
  - ▶ KEY_RIGHT
  - ▶ 'q' or 'Q'

☞ **int ProcessCommand()**

- Input : int command – GetCommand() 함수로부터 입력받은 명령
- Output : 초기에 1이지만, 강제 종료시만 QUIT(or 'q', tetris.h 참조)로 변경됨.
- GetCommand() 함수에서 입력받은 command를 input으로 받아서 적절한 동작을 취하는 함수이다. 이 함수에서도 GetCommand() 함수에서와 마찬가지로 명령을 구분하여 수행하게 된다. 각 명령에 따라 수행하는 동작은 다음과 같다.
  ▶ QUIT : 'q' 또는 'Q'를 입력해서 게임이 강제 종료된 경우 return 값을 tetris.h에 정의되어 있는 QUIT, 즉 'q'로 설정한다.
  ▶ KEY_UP : CheckToMove() 함수를 이용해서 시계 반대방향으로 90도 회전해서 화면에 그릴 수 있는지 체크한다.
  ▶ KEY_DOWN : CheckToMove() 함수를 이용해서 블록을 아래로 이동할 수 있는지 체크한다.
  ▶ KEY_LEFT : CheckToMove() 함수를 이용해서 블록을 왼쪽으로 이동할 수 있는지 체크한다.
  ▶ KEY_RIGHT : CheckToMove() 함수를 이용해서 블록을 오른쪽으로 이동할 수 있는지 체크한다.

☞ void DrawField()
- Input : 없음
- Output : 없음
- 테트리스 필드를 그리는 함수이다. 필드의 정보는 0, 1로 구분된다.
  ▶ 0 : 필드가 채워지지 않음을 나타내고, 화면에 '.'로 표시된다.
  ▶ 1 : 필드가 채워져 있다는 것을 나타내고, 공백 ' '을 attron(), attroff()를 사용해서 화면에 표시한다.

☞ void PrintScore()
- Input : int score – 현재까지 사용자가 플레이하고 얻은 점수
- Output : 없음
- 점수를 표시하는 상자 안의 정해진 위치로 이동해서 score 값을 화면에 출력하는 함수이다.

☞ void DrawNextBlock()
- Input : int *nextBlock – 블록의 ID를 저장하고 있는 배열
- Output : 없음
- Input parameter인 nextBlock의 정보를 바탕으로 테트리스 게임 화면에 다음 블록을 확인할 수 있는 상자 안에 4×4인 다음 블록을 그리는 함수이다.

☞ void DrawBlock()
- Input : int y – 블록의 y좌표
  int x – 블록의 x좌표
  int blockID – 블록의 ID
  int blockRotate – 블록의 회전수
  char tile – 블록을 채울 문자 모양

- Output : 없음
- 블록의 좌표, 블록의 모양, 블록의 회전수에 대한 정보들을 이용해서 정해진 위치에 블록을 그리는 함수이다. 이 때, 블록은 입력 받은 모양(tile)으로 채워진다.

☞ **void DrawBox()**

- Input : int y – 상자의 왼쪽 상단의 y 좌표
  int x – 상자의 왼쪽 상단의 x 좌표
  int height – 상자의 높이
  int width – 상자의 너비
- Output : 없음
- 상자의 왼쪽 상단의 위치, 상자의 높이와 너비를 입력받아서 입력받은 위치에, 입력받은 크기를 갖는 상자를 그리는 함수이다.

☞ **void play()**

- Input : 없음
- Output : 없음
- 테트리스 게임을 플레이하기 위한 기본 정보를 초기화하고, 테트리스 게임을 플레이를 하기 위한 전반적인 과정을 제어하는 함수이다. 아래 더 자세히 flow chart와 함께 설명된다.

☞ **char menu()**

- Input : 없음
- Output : 입력받은 메뉴 번호(character)
- main() 함수에서 동작하는 함수로 화면에 1~4번까지 메뉴를 출력하고, 사용자가 메뉴를 선택하길 기다린다. 사용자로부터 입력을 받으면, 그 값을 return한다. menu()에서 출력하는 메뉴의 구성은 다음과 같다.

```
1. play ← 테트리스 프로젝트 1주차 구현
2. rank ← 테트리스 프로젝트 2주차 구현
3. recommended paly ← 테트리스 프로젝트 3주차 (숙제) 구현
4. exit
```

7. 테트리스 게임 프로그램의 기본 흐름

테트리스 게임은 2가지 독립된 처리 과정으로 구성된다. 이는 다음 두 가지 함수에서 각각 이루어진다.

1. play() 함수

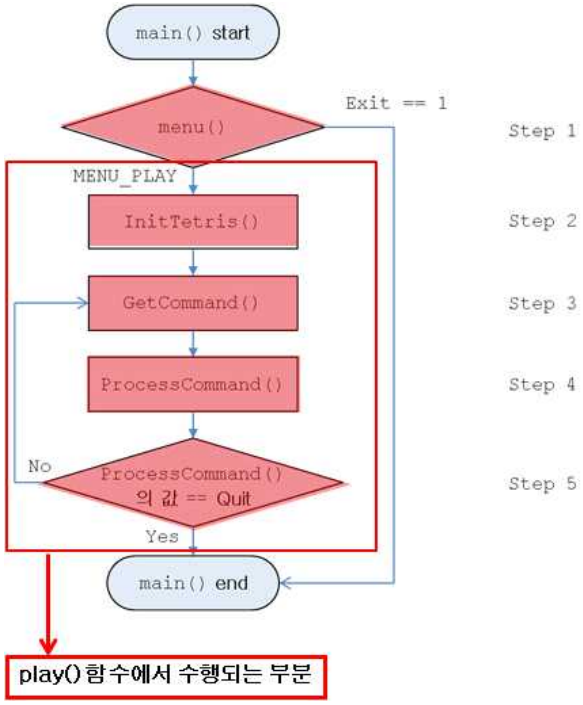
테트리스 게임을 위한 전역 변수들을 초기화하고, 대기 상태에서 사용자로부터 키보드 입력을 받아 그에 따른 동작을 수행하는 함수이다. 여기서 수행하는 동작들은 키보드 좌, 우, 상, 하 동작에 대응하는 블록의 이동 및 회전에 대한 처리와 'Q' 또는 'q'를 입력했을 때, 강제로 테트리스 게임 플레이를 종료시키고, "Good-bye!!"라는 메시지를 화면에 출력하는 처리이다. 다시 말해서, 함수 play()는 입력 대기상태에서 키보드의 입력이 발생했을 때, 그것을 구분하여 처리하는 동작을 수행하게 된다.

2. BlockDown() 함수

일정 시간 마다 호출(call)되는 함수로, 자동으로 수행된다. 이 함수는 일정 시간 마다 블록을 아래로 한 칸씩 떨어뜨리는 기능과 더 이상 블록을 내릴 수 없을 때, 블록을 필드에 쌓고, 빈 칸 없는 줄이 있는지 확인해서 삭제하며, 이에 따른 점수를 계산하는 기능을 수행한다. 그리고 필드에 더 이상 블록을 놓을 곳이 없을 때, 게임을 종료하는 기능도 수행한다.

7-1. play() 함수에 대한 동작 과정

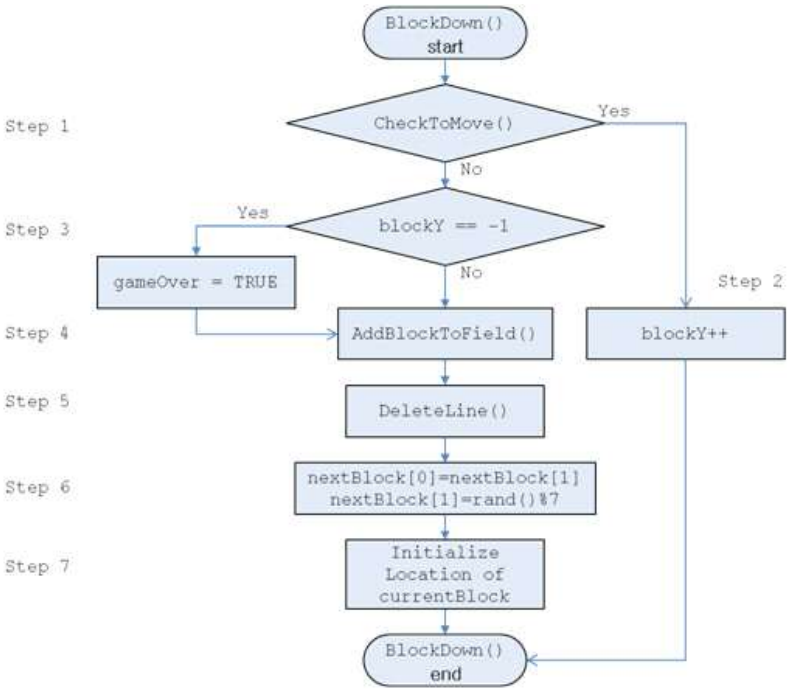
이 play()함수는 테트리스 게임을 플레이하기 위한 기본 정보를 초기화하고, 테트리스 게임을 플레이하는 사용자의 입력을 대기한다. 사용자로부터 키보드 입력이 있을 때, 입력에 따른 적절한 동작을 수행하게 된다. 키보드 입력이 좌(←), 우(→), 상(↑), 하(↓)면, 블록을 원하는 위치로 이동시키고, 'Q' 혹은 'q'를 입력하면, 플레이 중인 테트리스 게임을 강제로 종료하면서 “Good-bye!!”라는 메시지를 화면에 출력한다. 그리고 입력이 지정된 동작이 아니면, 입력 대기 상태로 돌아가서 다시 입력을 기다린다. 이 일련의 동작이 이루어지는 과정은 아래의 flow chart를 통해서 확인할 수 있다.



[그림 11] play() 함수의 flow chart

7-2 BlockDown() 함수에 대한 동작 과정

이 BlockDown() 함수는 일정 시간마다 수행되는 함수로, 현재 필드에 나타난 블록을 일정시간 마다 아래로 떨어뜨리는 함수이다. 이 때, 수행되는 동작은 다음에 제시되는 flowchart와 같다.



[그림 12] BlockDown() 함수의 flow chart

위에서 보는 바와 같이 BlockDown() 함수는 블록이 한 칸 내려갈 수 있는지를 CheckToMove() 함수를 통해서 확인해, 만약 블록을 한 칸 내릴 수 있다면, 해당 블록을 한 칸 내리고, 바뀐 블록의 위치를 다시 그린 후 함수를 종료한다. 그리고 블록을 한 칸 내릴 수 없는 경우는 두 가지 경우로 나누어 동작하게 된다. 첫 번째, 만약 블록의 y좌표가 -1이라면, 블록이 필드 상에 나타나자마자 내릴 수 없는 상태가 되기 때문에, 더 이상 테트리스 게임을 플레이 할 수 없는 상태이다. 따라서 gameover 변수를 1로 만들어서 테트리스 게임 플레이를 종료하게 만든다. 블록의 y좌표가 -1이 아니라면, 해당 블록은 필드 위에 쌓이게 되는데, 블록이 필드에 쌓이는 동작은 AddBlockToField() 함수를 통해서 수행된다. 블록이 필드에 쌓인 후에는 필드 상에 빈 칸이 없는 줄이 있는지 확인하여 삭제하고, 제거된 줄의 수를 세어, 점수를 업데이트 한다. 이 모든 동작이 수행되고 나면, 다음 블록이 현재 블록이 되고, 각 블록에 대한 변수들이 초기화 되며, 다음 블록의 ID가 새롭게 임의로 생성된 후, 위의 과정에서 변경된 모든 정보를 화면에 그려 주고 함수를 종료한다.

8. 실험 방법(테트리스 프로젝트 1주차에서 구현할 함수들과 구현 결과)

8-1 구현할 함수들

테트리스 프로젝트 1주차에서 구현할 함수는 테트리스 게임을 play할 때, 사용자의 키보드 입력에 따라 블록의 동작을 제어하는 4가지 함수들과 변경된 정보를 화면에 그려주는 1가지 함수를 구현하게 된다.

☞ **int CheckToMove()**

- input : char field[HEIGHT][WIDTH] - 테트리스 필드의 정보
  int currentBlock - 현재 블록의 ID
  int blockRotate - 현재 블록의 회전수 혹은 회전하게 될 회전수

int blockY – 현재 블록의 y 좌표 혹은 이동하게 될 y 좌표
int blockX – 현재 블록의 x 좌표 혹은 이동하게 될 x 좌표

- output : input의 정보를 갖는 블록이 원하는 좌표로 이동할 수 있다면 1
input의 정보를 갖는 블록이 원하는 좌표로 이동할 수 없다면 0

- 이 함수는 현재 필드의 상태, 현재 블록의 ID와 회전수, 이동할 좌표를 input으로 하여 해당 블록이 정해진 위치로 이동할 수 있는지 없는지를 체크하여 주는 함수이다.

- 구현

블록은 4×4 행렬이므로 이중 loop를 사용하고, 가장 안의 loop에서 체크해야 할 사항은 다음과 같다.

1) 블록을 놓으려고 하는 필드에 이미 블록이 쌓여져 있는지 여부
2) 블록을 나타내는 4×4 행렬의 각 요소의 실제 필드상의 y 좌표가 HEIGHT 보다 크거나 같은지 여부
3) 블록을 나타내는 4×4 행렬의 각 요소의 실제 필드상의 x 좌표가 0보다 작은지 여부
4) 블록을 나타내는 4×4 행렬의 각 요소의 실제 필드상의 x 좌표가 WIDTH보다 크거나 같은지 여부

만약 이 4가지 경우 중 하나로 충족하는 경우는 블록을 input 좌표로 이동할 수 없으므로 0을 return하고, 모든 경우를 만족하지 않는다면 블록을 input 좌표로 이동할 수 있으므로 1을 return한다.

☞ **void DrawChange()**

- input : char field[HEIGHT][WIDTH] – 테트리스 필드의 정보
int command – 현재 수행하게 될 command
int currentBlock – 현재 블록의 ID
int blockRotate – 현재 블록의 회전수
int blockY – 현재 블록의 y 좌표
int blockX – 현재 블록의 x 좌표
- output : 없음

- 변경된 정보를 반영하여 새로운 위치나 다른 회전수를 갖는 블록을 다시 그리는 함수이다.

- 구현

현재 수행하게 될 command를 입력받아서 command에 따라 블록의 회전수, 좌표를 바꾸고, 변경된 위치나 모양의 블록을 다시 그린다. 블록을 다시 그릴 때는 DrawBlock() 함수를 이용한다. 단, 새로운 블록을 그리기 전에, command를 수행하기 전에 화면에 그려졌던 블록을 지우는 과정도 반드시 수행해야 한다. 이 과정에서는 블록의 모양이 4×4 행렬이므로 이중 loop를 사용하여, 이전 블록 부분을 모두 '.'으로 바꾼다.

☞ **int BlockDown()**

- input : int sig
- output : 없음

- 현재의 블록을 한 칸씩 아래로 떨어뜨리는 함수이다.

- 구현

  CheckToMove() 함수를 이용해서 블록을 한 칸 내릴 수 있는지 없는지 체크하여 블록을 한 칸 내릴 수 있다면, 블록을 한 칸 내리고 필드 정보 및 블록 정보를 갱신하여 다시 그린다. 만약 블록을 내릴 수 없고 블록의 y좌표가 -1이면, gameover를 1로 갱신하고, 갱신된 정보를 다시 그린다. 이와 비슷하게 블록을 한 칸 내릴 수 없고 블록의 y좌표가 -1이 아니면, AddBlockToField() 함수를 사용하여 블록을 필드에 추가하고, 점수를 계산하여 누적하며, 필드 정보를 갱신하여 다시 그린다. BlockDown() 함수는 앞의 flow chart에서 자세히 설명되어 있다.

☞ **int AddBlockToField()**

- input : char field[HEIGHT][WIDTH] - 테트리스 필드의 정보
  int currentBlock - 현재 블록의 ID
  int blockRotate - 현재 블록의 회전수
  int blockY - 현재 블록의 y 좌표
  int blockX - 현재 블록의 x 좌표
- output : 현재 결과값은 0이다. 하지만, 1주차 숙제 구현시, 필드와 블록이 맞닿은 면적을 고려한 점수를 return한다.

- 주어진 필드상의 위치에 현재 블록을 쌓는 함수이다.

- 구현

  CheckToMove() 함수에서 이동할 수 있는지 없는지를 체크하기 때문에, 이 함수에서는 단순히 4×4 크기의 블록을 필드에 추가한다. 따라서 이중 loop(각 좌표에 대해 index i, j를 사용)를 사용하여 구현할 수 있다. 이 loop를 수행 중 2번째 loop에서 현재 블록이 채워져 있다면, 즉, currentBlock[i][j] == 1이라면, 현재 테트리스 필드의 정보, field[blockY+i][blockX+j]를 1로 바꾸어 주면 된다.

☞ **int DeleteLine()**

- input : char field[HEIGHT][WIDTH] - 테트리스 필드의 정보
- output : 지워진 줄(or 라인) 수에 따른 점수

- 완전히 채워진 라인을 찾아 지우고, 지운 라인수에 대한 점수를 계산하여 return하는 함수이다.

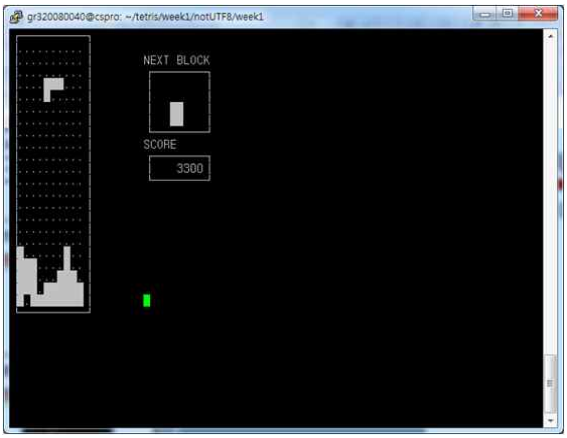
- 구현

테트리스 필드를 이중 loop를 통해서 탐색하면서 완전히 채워진 라인이 있는지 체크한다. 이를 체크하는 방법은 각 라인을 살펴보다가, 해당 라인에 하나의 빈 칸이라도 발생하면, 지금까지 동작을 멈추고, 다음 라인을 체크한다. 만약 빈 칸 없는 라인이 있다면, 지워진 라인 수를 증가시키고, 이 라인을 테트리스 필드에서 삭제하며, 지워진 라인 위의 필드 정보를 모두 한 줄씩 내려 갱신한다. 마지막으로 지워진 라인수를 사용해 점수를 계산하고, return한다.

8-2 구현결과

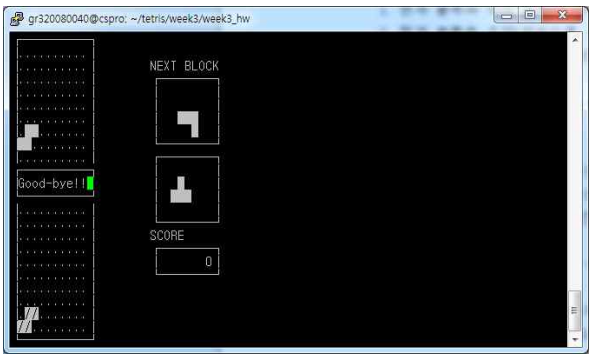
테트리스 프로젝트 1주차 실습이 완료되면, 다음과 같이 메뉴 1번의 테트리스 플레이가 정상적으로 수행된다.

- 게임화면



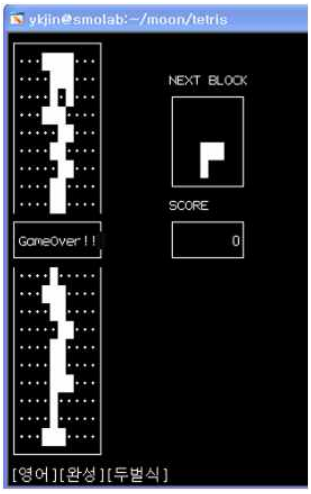
[그림 13] 테트리스 게임 화면

- ‘Q’ or ‘q’를 입력하여 강제 종료



[그림 14] ‘Q’ or ‘q’를 누르고 강제 종료시, “Good-bye!!”가 출력된 화면

- 더 이상 블록을 놓을 곳이 없게 되어 게임 종료



[그림 15] 게임이 종료되었을 때, “GameOver!!”가 출력된 화면

8-3 테트리스 프로젝트 1주차 실험 평가

테트리스 프로젝트 1주차 실험은 다음의 평가 항목들을 통해서 평가된다.

- 5가지 함수가 모두 구현이 되었는가?
- 블록이 필드에 정확하게 쌓이는가?
- 블록이 잘 회전하는가?
- 블록이 필드를 벗어나 이동하지 않는가?
- 빈 칸 없는 줄이 삭제되고 점수 계산이 올바르게 되는가?
- 블록이 정상적으로 일정시간마다 떨어지는가?
- 블록을 더 이상 놓을 곳이 없을 때, 게임이 종료되는가?
- 바뀐 정보들이 화면에 올바르게 표시되는가?
- 다음 블록이 7가지 블록 중 임의로 생성되는가?

9. 테트리스 프로젝트 1주차 숙제 및 보고서 작성

9-1. 예비보고서

1. 테트리스 frame 프로그램 파일을 미리 읽어보고(부록1 ncurses 라이브러리 포함), 테트리스 게임의 flow chart를 자세히 작성하시오. 그리고 테트리스 게임을 구성하는 각 함수의 기능에 대해서 설명하시오.
2. 실습시간에 구현할 5가지 함수들에 대한 간단한 pseudo code를 제시하시오.

9-2. 숙제

1. 일정 시간 마다 떨어지고 있는 현재 블록의 위치에서 그대로 아래로 내린 상황을 가정했을 때, 블록이 필드에 놓여질(or 쌓일) 위치를 표시하는 그림자 기능을 구현한다.

   - 수정할 함수

   ☞ `void DrawBlock(int y, int x, int blockID, int blockRotate, char tile);`

- DrawBlock() 함수는 문자(char tile)를 parameter로 받아서 블록을 그릴 때, tile로 블록을 채워 그리는 기능을 한다. 현재 블록은 ' ', 그림자 기능은 '/'을 tile로 사용한다.

☞ void DrawShadow(int y, int x, int blockID, int blockRotate);

- DrawShadow() 함수는 현재 블록을 가장 아래로 내렸을 때, 더 이상 내려갈 수 없는 위치를 그림자의 위치로 결정하고, 그 위치에 '/'으로 채워진 현재의 블록을 그린다.

- 구현할 함수

☞ void DrawBlockWithFeatures(int y, int x, int blockID, int blockRotate);

- 이 함수는 위 두 가지 DrawBlock(), DrawShadow() 함수를 호출하는 함수로, 기존의 DrawBlock() 함수의 위치에 삽입하여 움직임이 갱신될 때마다 현재 블록과 함께 그림자를 그린다.

2. 다음 블록을 표시하기 위한 상자를 한 개 더 추가하고, 2개의 다음 블록을 미리 볼 수 있도록 한다.

☞ 1주차 구현 후 next 블록을 하나 더 추가하기 위해서 tetris.h의 nextBlock의 배열의 크기를 2에서 3으로 1개 증가하여, 현재 블록과 다음 2개의 블록을 저장할 수 있도록 수정한다.

- 수정할 함수

☞ void InitTetris();

- 이 함수에서는 현재 블록과 다음 블록 한 개의 정보만을 초기화 했으나, 블록의 크기가 증가되었기 때문에, 두 번째 다음 블록의 ID를 임의로 생성한다 (rand()%7).

☞ void DrawNextBlock(int *nextBlock);

- 이 함수는 위에 그린 블록의 초기 위치를 변경하여 두 번째 다음 블록을 그리는 과정을 추가한다.

☞ void BlockDown(int sig);

- 이 함수에서는 블록을 더 이상 아래로 내릴 수 없을 때, 첫 번째 다음 블록은 현재 블록의 ID를, 두 번째 다음 블록은 첫 번째 다음 블록 ID를 갖게 되고, 마지막 두 번째 블록의 ID는 새롭게 생성되는 과정을 추가한다.

3. 블록이 더 이상 내려갈 수 없을 때, 필드에 블록이 놓여진다. 이 때, 점수가 누적되는데, 현재는 빈 칸이 없는 줄을 삭제 하고 점수를 증가시키는 방식을 갖고 있다. 이 방법에 추가하여, 블록이 놓여진 위치에서 필드와 블록의 아래 부분이 맞닿은 면적의 수에 비례하여 점수를 증가시키는 기능을 추가 구현한다. 따라서 score는 다음과 같이 계산된다.

score = (지워진 줄 수)^2×100
+ (필드와 블록의 아래 부분이 맞닿은 면적의 수)×10

- 수정할 함수

☞ `int AddBlockToField(char field[HEIGHT][WIDTH], int currentBlock, int blockRotate, int blockY, int blockX);`

- 이 함수에서는 touched라는 변수를 만들고, 블록의 정보와 필드 정보를 사용해서 현재 블록과 필드가 맞닿아 있는 면적을 세고, 점수를 계산해서 return 한다. 맞닿은 면적을 세는 방법은 4×4 배열의 블록을 이중 loop를 통해서 탐색하면서, 블록 나타내는 배열의 원소가 1이고, 바로 아래 필드의 요소가 1이면, 변수 touched를 1 증가시킨다(이와 같이 할 수 있는 이유는 CheckToMove() 함수에서 이동할 수 있는지의 여부를 체크하기 때문이다). 그리고 이렇게 얻어진 변수 touched의 값을 이용해서 score는 다음과 같이 계산하고, return한다.

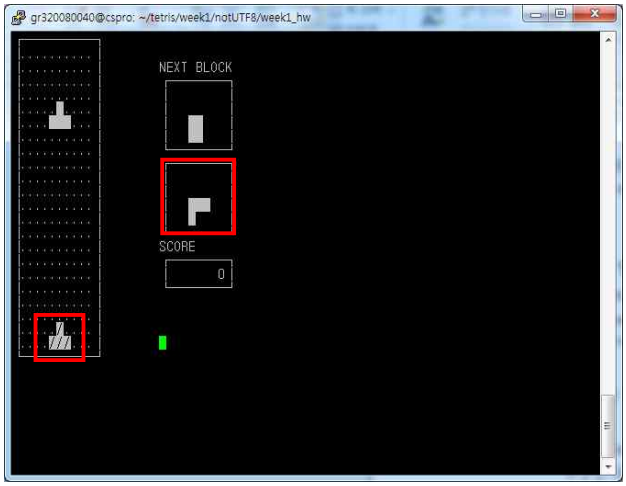
```
score = touched * 10;
```

☞ `void BlockDown(int sig);`

- 이 함수에서는 AddBlockToField() 함수에서 계산된 점수를 누적한다.

- 구현 결과

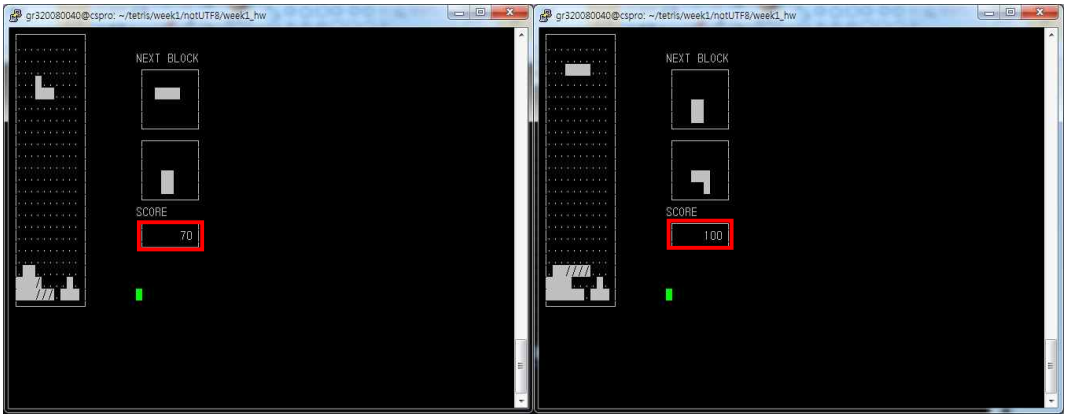
1. 그림자 기능과 다음 블록 보여주기 상자 및 다음 블록 추가



[그림 16] 그림자 기능과 추가된 다음 블록을 미리 보여주는 상자

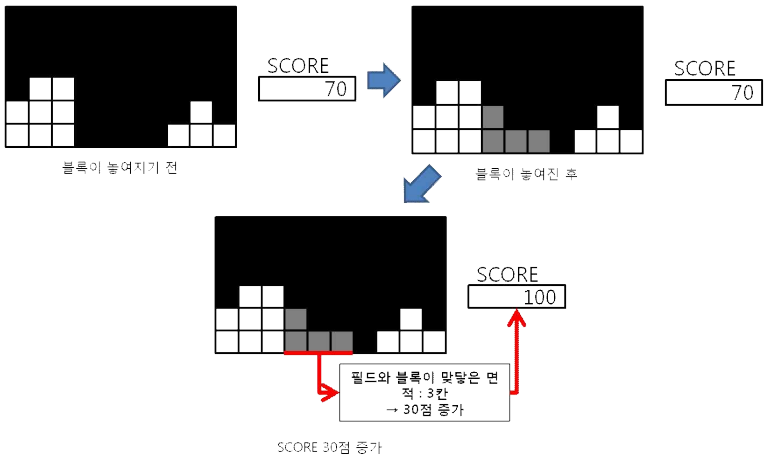
2. 새로운 score 계산법에 대한 예제

- 아래 왼쪽 예제에서 블록이 현재 그림자 기능으로 보여지는 위치에 놓여지면 필드와 맞닿은 면적이 3이므로 30점이 오른쪽 그림처럼 추가된다.



[그림 17] 블록과 필드와 맞닿은 면적이 3일 때, 점수가 증가하는 화면

- 위 게임 플레이 화면을 확대하면 아래와 같다.



[그림 18] 그림 19를 확대한 상황

9-3. 결과 보고서

아래의 사항을 작성하여 다음 실험 시간에 제출하시오.

1. 실습시간에 작성한 프로그램의 함수들이 예비보고서에서 작성한 각 구현 함수들의 pseudo code와 어떻게 달라졌는지 설명하고, 각 함수에 대한 시간 및 공간 복잡도를 보이시오(각 함수의 시간 및 공간복잡도를 구할 때, 어떤 변수에 의존하는지를 판단해야 한다).
2. 테트리스 프로젝트 1주차 숙제 문제를 해결하기 위한 pseudo code를 기술하고, 작성한 pseudo code의 시간 및 공간 복잡도를 보이시오.

실험 PRJ-1 1주차 기본 테트리스 프로그램 예비보고서

전공: 학년: 학번: 이름

실험 PRJ-1 1주차 기본 테트리스 프로그램 결과보고서

전공: 학년: 학번: 이름