C++프로그래밍및실습

테트리스 만들기

진척 보고서 #1

제출일자: 2024.11.17
제출자명: 주소연
제출자학번: 223310

1. 프로젝트 목표

1) 배경 및 필요성

테트리스는 1984년 출시 이후 전 세계적으로 사랑받아 온 클래식 퍼즐 게임으로, 간단하고 직관적인 규칙과 높은 접근성 덕분에 폭넓은 인기를 얻어 왔습니다. 하지만 테트리스는 겉보기와 달리 심도 있는 게임성을 제공하며, 플레이어는 블록을 빠르게 회전시키고 적절한 위치에 맞추어 쌓아야 하는 도전 과제에 몰입하게 됩니다.

프로그래밍 학습의 관점에서, 콘솔 기반의 테트리스 개발은 자료구조와 알고리즘, 사용자 인터페이스 같은 다양한 프로그래밍 개념을 실제로 적용해볼 수 있는 이상적인 프로젝트입니다. 게임의 블록 생성과 회전, 충돌 감지, 라인 제거 같은 기능은 다양한 자료구조와 알고리즘을 적용해볼 수 있는 기회를 제공합니다. 또한, 게임의 흐름과 상태를 관리하기 위해 인터페이스와 게임 루프 설계가 필요하며, 이 과정에서 유용한 프로그래밍 패턴과 설계 기법을 학습할 수 있습니다.

2) 프로젝트 목표

이 프로젝트는 콘솔 기반의 테트리스 게임을 C++로 구현하여, 블록의 이동과 회전, 라인 제거 등 게임의 핵심 메커니즘을 학습하고, 이를 통해 자료구조, 알고리즘, 객체지향 설계의 이해를 높이는 것을 목표로 합니다.

3) 차별점

단순하고 직관적인 구조를 통해 테트리스의 핵심 기능을 구현하고, 코드의 가독성과 효율성을 높이는 데 초점을 둡니다. 이를 통해 유지보수와 확장성을 고려한 깨끗한 코드 작성을 목표로 합니다.

2. 기능 계획

1) 기능 1: 블록이 상단에서 하단으로 내려오는 기능

(1) 세부 기능 1 : 일정 시간 간격으로 블록이 한 칸씩 내려오도록 설정

– 설명: 블록의 초기 위치를 설정해서 구현한다.

2) 기능 2: 총 7가지 모양의 도형

- 설명: 새로운 블록이 생성될 때 7가지 중 하나를 무작위로 선택하여 생성한다.

3) 기능 3: 특정 키를 눌렀을 때 도형 회전

- 설명: 특정 키를 눌렀을 때 도형이 회전한다. (4가지 방향으로 회전)

4) 기능 4: 블록이 바닥 또는 다른 블록과 닿으면 다음 도형으로 넘어감

- 설명: 블록이 바닥에 닿았는지 또는 다른 블록과 닿았는지 검사하고 블록이 닿은 경우 현재 위치에 고정하고 새로운 블록 생성한다. 새로운 블록을 게임 상단에서 생성하고 다시 하강 시작한다.

5) 기능 5: 도형을 맞추어 일자가 되면 제거하고 다른 블록을 아래로 이동

- 설명: 블록이 고정될 때마다 가로 라인이 꽉 찼는지 검사하고 가득 찬 라인이 있으면 해당 라인을 제거한다. 제거된 라인 위에 있는 블록들을 한 줄씩 아래로 이동한다.

3. 진척사항

1) 기능 구현

(1) 블록이 상단에서 하단으로 내려오는 기능

- 입출력: 입력: #define MAPWIDTH 15, #define MAPHEIGHT 30

출력: 블록이 위에서 아래로 한 칸씩 이동하는 모습이 콘솔에 출력됩니다.

- 설명: 초기 맵 생성 후, 블록이 랜덤으로 생성됩니다. MoveDown()이 블록의 Y좌표를 증가시켜 아래로 이동시키고 DrawBlock()이 변경된 위치에 블록을 그립니다. 따라서 블록의 위치가 한 칸씩 내려가고 화면에 출력됩니다.

- 적용된 배운 내용: 4주차 조건문, 반복문, 7주차 함수

- 코드 스크린샷

```
void MoveDown()
{
    blockPosition.Y++;
}

void DrawBlock(char map[MAPHEIGHT][MAPWIDTH])
{
    for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            if (currentBlock[i][j] == 1 &&
                blockPosition.Y + i < MAPHEIGHT &&
                blockPosition.X + j < MAPWIDTH)
            {
                map[blockPosition.Y + i][blockPosition.X + j] = '1';
            }
        }
    }
}
```

(2) 총 7가지 모양의 도형

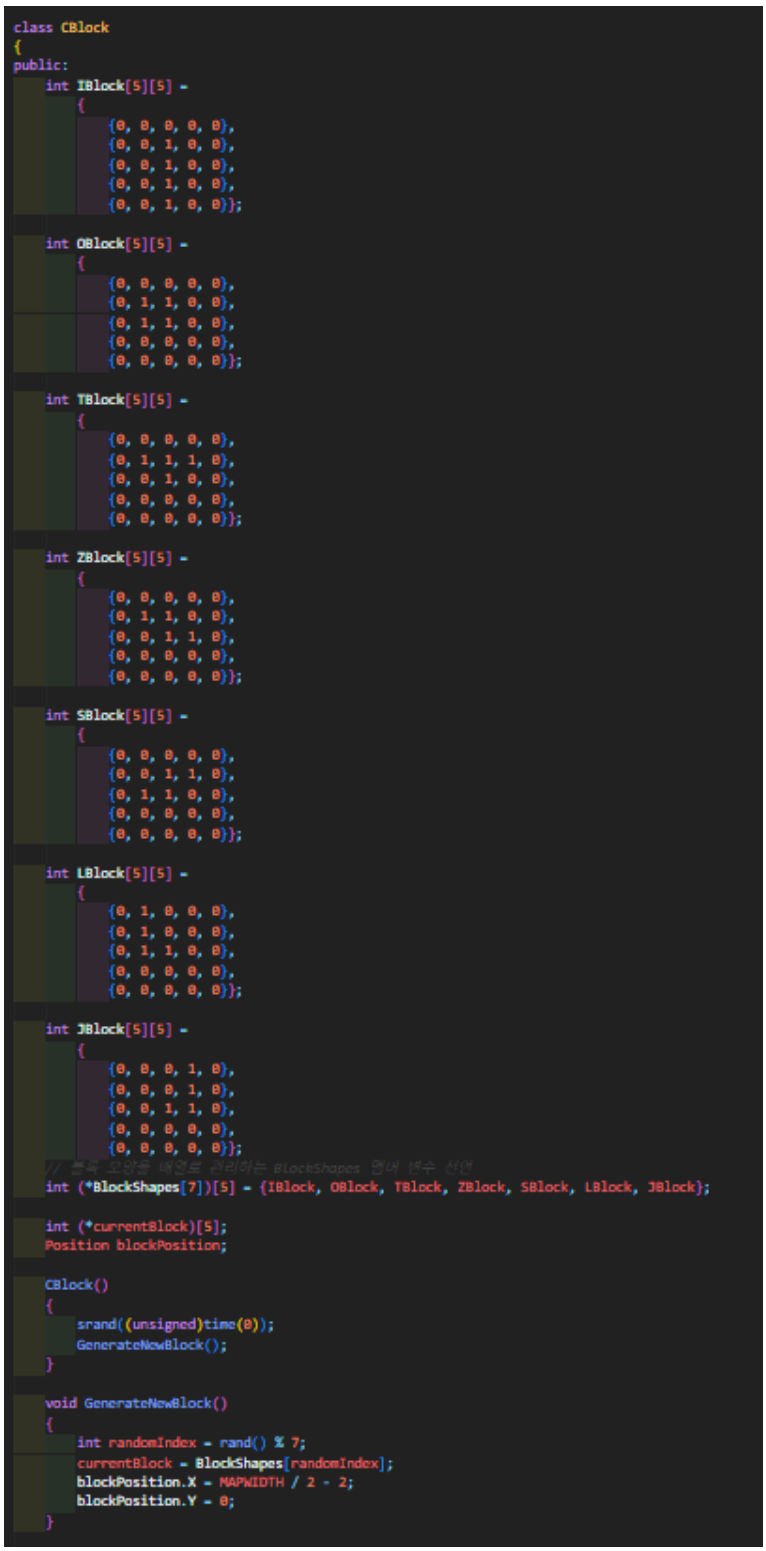
- 입출력: 입력: #define MAPWIDTH 15

출력: 7가지 도형 중 하나가 랜덤하게 생성되어 초기위치에 설정됩니다.

- 설명: 7가지 테트리스 블록의 모양을 2차원 배열로 정의하고 이들을 포인터 배열로 관리하며 새 블록이 필요할 때마다 랜덤하게 하나를 선택하여 맵 상단 중앙에 생성하는 기능을 구현합니다.

- 적용된 배운 내용: 5주차 배열, 9주차 클래스, 11주차 포인터

- 코드 스크린샷



```
class CBlock
{
public:
    int IBlock[5][5] =
        {
            {0, 0, 0, 0, 0},
            {0, 0, 1, 0, 0},
            {0, 0, 1, 0, 0},
            {0, 0, 1, 0, 0},
            {0, 0, 1, 0, 0}};

    int OBlock[5][5] =
        {
            {0, 0, 0, 0, 0},
            {0, 1, 1, 0, 0},
            {0, 1, 1, 0, 0},
            {0, 0, 0, 0, 0},
            {0, 0, 0, 0, 0}};

    int TBlock[5][5] =
        {
            {0, 0, 0, 0, 0},
            {0, 1, 1, 1, 0},
            {0, 0, 1, 0, 0},
            {0, 0, 0, 0, 0},
            {0, 0, 0, 0, 0}};

    int ZBlock[5][5] =
        {
            {0, 0, 0, 0, 0},
            {0, 1, 1, 0, 0},
            {0, 0, 1, 1, 0},
            {0, 0, 0, 0, 0},
            {0, 0, 0, 0, 0}};

    int SBlock[5][5] =
        {
            {0, 0, 0, 0, 0},
            {0, 0, 1, 1, 0},
            {0, 1, 1, 0, 0},
            {0, 0, 0, 0, 0},
            {0, 0, 0, 0, 0}};

    int LBlock[5][5] =
        {
            {0, 1, 0, 0, 0},
            {0, 1, 0, 0, 0},
            {0, 1, 1, 0, 0},
            {0, 0, 0, 0, 0},
            {0, 0, 0, 0, 0}};

    int JBlock[5][5] =
        {
            {0, 0, 0, 1, 0},
            {0, 0, 0, 1, 0},
            {0, 0, 1, 1, 0},
            {0, 0, 0, 0, 0},
            {0, 0, 0, 0, 0}};
    // 블록 모양을 배열로 관리하는 BlockShapes 멤버 변수 선언
    int (*BlockShapes[7])[5] = {IBlock, OBlock, TBlock, ZBlock, SBlock, LBlock, JBlock};

    int (*currentBlock)[5];
    Position blockPosition;

    CBlock()
    {
        srand((unsigned)time(0));
        GenerateNewBlock();
    }

    void GenerateNewBlock()
    {
        int randomIndex = rand() % 7;
        currentBlock = BlockShapes[randomIndex];
        blockPosition.X = MAPWIDTH / 2 - 2;
        blockPosition.Y = 0;
    }
```

프로그램 실행방법

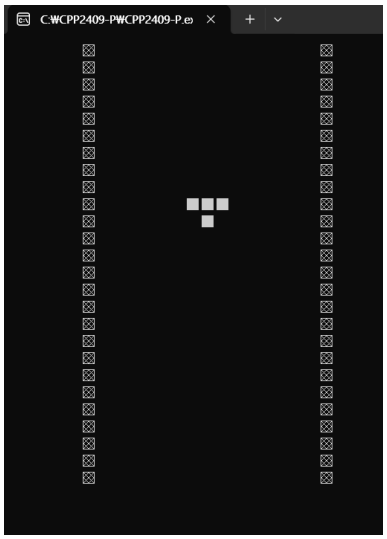
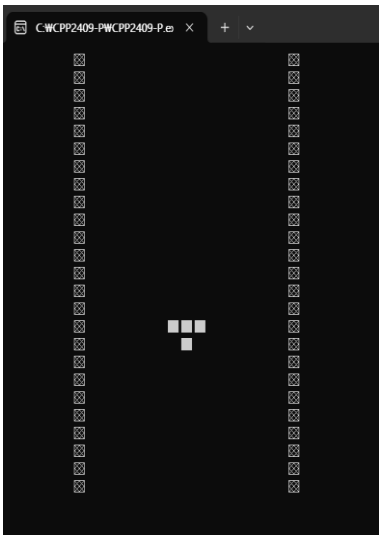
프로그램 실행 시 '1' 입력: 게임 시작, '2' 입력: 프로그램 종료를 선택하여 실행합니다.

2) 테스트 결과

(1) 블록이 상단에서 하단으로 내려오는 기능

- 설명: 생성된 블록이 0.5초(Sleep(500)) 간격으로 한 칸씩 자동으로 하단으로 이동합니다.

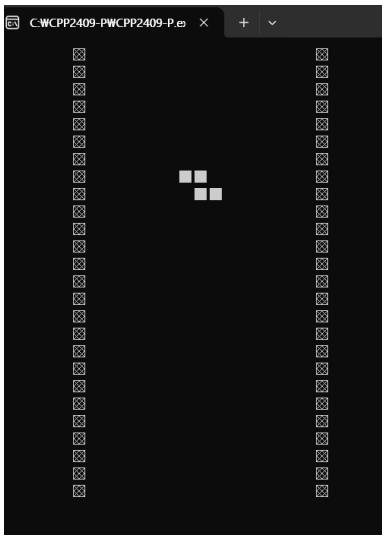
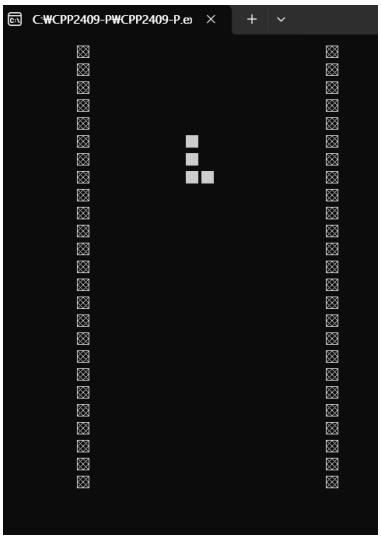
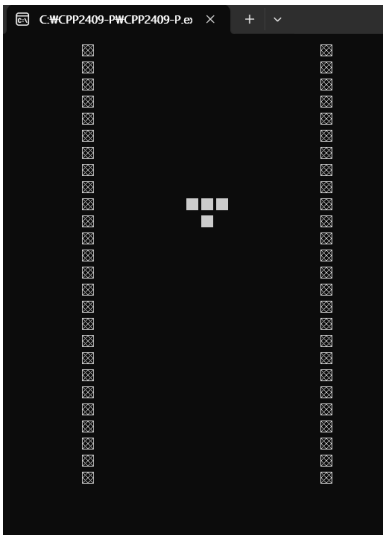
- 테스트 결과 스크린샷

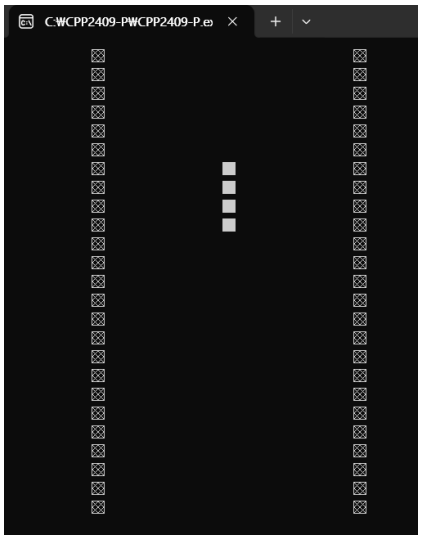



(2) 총 7가지 모양의 도형

- 설명: 모든 블록 모양이 정상적으로 구현되었는지 테스트합니다.

- 테스트 결과 스크린샷



4. 계획 대비 변경 사항

없음

5. 프로젝트 일정

(진행한 작업과 진행 중인 작업 등을 표기)

업무		11/3	11/10	11/17	12/1
제안서 작성		완료			
기능1	세부기능1		---------->		
기능2			---------->		
기능3				---------->	