



- m_pBoard (이동 완료된 블록 정보 저장)

0 , 0	0 , 1	0 , 2	0 , 3	0 , 4	0 , 5
1 , 0	1 , 1	1 , 2	1 , 3	1 , 4	1 , 5
2 , 0	2 , 1	2 , 2	2 , 3	2 , 4	2 , 5
3 , 0	3 , 1	3 , 2	3 , 3	3 , 4	3 , 5
4 , 0	4 , 1	4 , 2	4 , 3	4 , 4	4 , 5
5 , 0	5 , 1	5 , 2	5 , 3	5 , 4	5 , 5
6 , 0	6 , 1	6 , 2	6 , 3	6 , 4	6 , 5
7 , 0	7 , 1	7 , 2	7 , 3	7 , 4	7 , 5

- m_pTetromino (이동중인 블록 정보)

	(0 , 1)
	(1 , 1)
(2 , 0)	(2 , 1)

- 충돌체크

0 , 0	0 , 1	0 , 2	0 , 3	0 , 4	0 , 5
1 , 0	1 , 1	1 , 2	1 , 3	1 , 4	1 , 5
2 , 0	2 , 1	2 , 2	2 , 3	2 , 4	2 , 5
3 , 0	3 , 1	3 , 2	3 , 3	3 , 4	3 , 5
4 , 0	4 , 1	4 , 2	(0 , 1)	4 , 4	4 , 5
5 , 0	5 , 1	5 , 2	(1 , 1)	5 , 4	5 , 5
6 , 0	6 , 1	(2 , 0)	(2 , 1)	6 , 4	6 , 5
7 , 0	7 , 1	7 , 2	7 , 3	7 , 4	7 , 5

Board 에 쌓여있는 블럭좌표 (6, 3) (7, 2) (7, 3) (7, 4)

Tetrimino 의 시작좌표 : (4, 2)

Tetrimino 의 내부블럭 좌표 : (0, 1) (1, 1) (2, 0) (2, 1)

Tetrimino 내부블럭을 반복 실행

{

시작좌표 (4, 2) + 내부좌표 (0, 1) = 보드좌표 (4, 3)

시작좌표 (4, 2) + 내부좌표 (1, 1) = 보드좌표 (5, 3)

시작좌표 (4, 2) + 내부좌표 (2, 0) = 보드좌표 (6, 2)

시작좌표 (4, 2) + 내부좌표 (2, 1) = 보드좌표 (6, 3) 충돌!!

}

- 회전

0, 0	0, 1	0, 2	0, 0	0, 1	0, 2
1, 0	1, 1	1, 2	1, 0	1, 1	1, 2
2, 0	2, 1	2, 2	2, 0	2, 1	2, 2

0, 0	0, 1	0, 2	0, 0	0, 1	0, 2
1, 0	1, 1	1, 2	1, 0	1, 1	1, 2
2, 0	2, 1	2, 2	2, 0	2, 1	2, 2

- 기본 회전 알고리즘

```
for ( int y = 0; y < 3; y++ )
```

```
{
```

```
    for ( int x = 0; x < 3; x++ )
```

```
    {
```

```
        newBlock [ y ] [ x ] = block [ x ] [ 블록수 - y ]; // 블록수는 (3*3 블록의 3 에서 1 올빼값) = 2
```

```
    }
```

```
}
```

(1)

(0, 0) 블록이동 : newBlock [2] [0] = block [0] [2 - 2]; // (0, 0) 블록이 (2, 0) 으로 이동

(0, 1) 블록이동 : newBlock [1] [0] = block [0] [2 - 1]; // (0, 1) 블록이 (1, 0) 으로 이동

(0, 2) 블록이동 : newBlock [0] [0] = block [0] [2 - 0]; // (0, 2) 블록이 (0, 0) 으로 이동

(2)

(0, 0) 블록이동: newBlock [2] [0] = block [0] [2 - 2]; // (0, 0) 블록이 (2, 0) 으로 이동

(1, 0) 블록이동: newBlock [2] [1] = block [1] [2 - 2]; // (1, 0) 블록이 (2, 1) 으로 이동

(2, 0) 블록이동: newBlock [2] [2] = block [2] [2 - 2]; // (2, 0) 블록이 (2, 2) 으로 이동

0, 0	0, 1	0, 2
1, 0	1, 1	1, 2
2, 0	2, 1	2, 2

0, 0	0, 1	0, 2
1, 0	1, 1	1, 2
2, 0	2, 1	2, 2

- 수정한 회전 알고리즘

```

for ( int y = 0; y < 3; y++ )
{
    for ( int x = 0; x < 3; x++ )
    {
        newBlock [ y ] [ x ] = block [ x ] [ 가로블럭수 - y - 1 ];
    }
}

```

- (1) 가로블럭수 : 1
- (0, 0) 일때 : newBlock [0] [0] = block [0] [1 - 0 - 1]; // (0, 0) 블록이 (0, 0) 으로 이동
- (1, 0) 일때 : newBlock [0] [1] = block [1] [1 - 0 - 1]; // (1, 0) 블록이 (0, 1) 으로 이동
- (2, 0) 일때 : newBlock [0] [2] = block [2] [1 - 0 - 1]; // (2, 0) 블록이 (0, 2) 으로 이동
- (2) 가로블럭수 : 3
- (0, 0) 일때 : newBlock [2] [0] = block [0] [3 - 2 - 1]; // (0, 0) 블록이 (2, 0) 으로 이동
- (0, 1) 일때 : newBlock [1] [0] = block [0] [3 - 1 - 1]; // (0, 1) 블록이 (1, 0) 으로 이동
- (0, 2) 일때 : newBlock [0] [0] = block [0] [3 - 0 - 1]; // (0, 2) 블록이 (0, 2) 으로 이동

위와 같이 항상 블록의 세로가 위쪽으로 붙도록 함

블럭의 가로를 가운데정렬 함

(회전 전 가로 블럭 수 - 회전 후 가로 블럭 수) / 2

$$(1 - 3) / 2 = -2 / 2 = -1$$

Tetrimino 의 x좌표를 -1 만큼 이동 시키면 아래와 같이 중간으로 정렬 됨

0, 0	0, 1	0, 2
1, 0	1, 1	1, 2
2, 0	2, 1	2, 2

0, 0	0, 1	0, 2
1, 0	1, 1	1, 2
2, 0	2, 1	2, 2

(회전 전 가로 블럭 수 - 회전 후 가로 블럭 수) / 2

$$(3 - 1) / 2 = 2 / 2 = 1$$

Tetrimino 의 x좌표를 1 만큼 이동 시키면 아래와 같이 중간으로 정렬 됨

0, 0	0, 1	0, 2
1, 0	1, 1	1, 2
2, 0	2, 1	2, 2

0, 0	0, 1	0, 2
1, 0	1, 1	1, 2
2, 0	2, 1	2, 2