Tetris 코딩하기

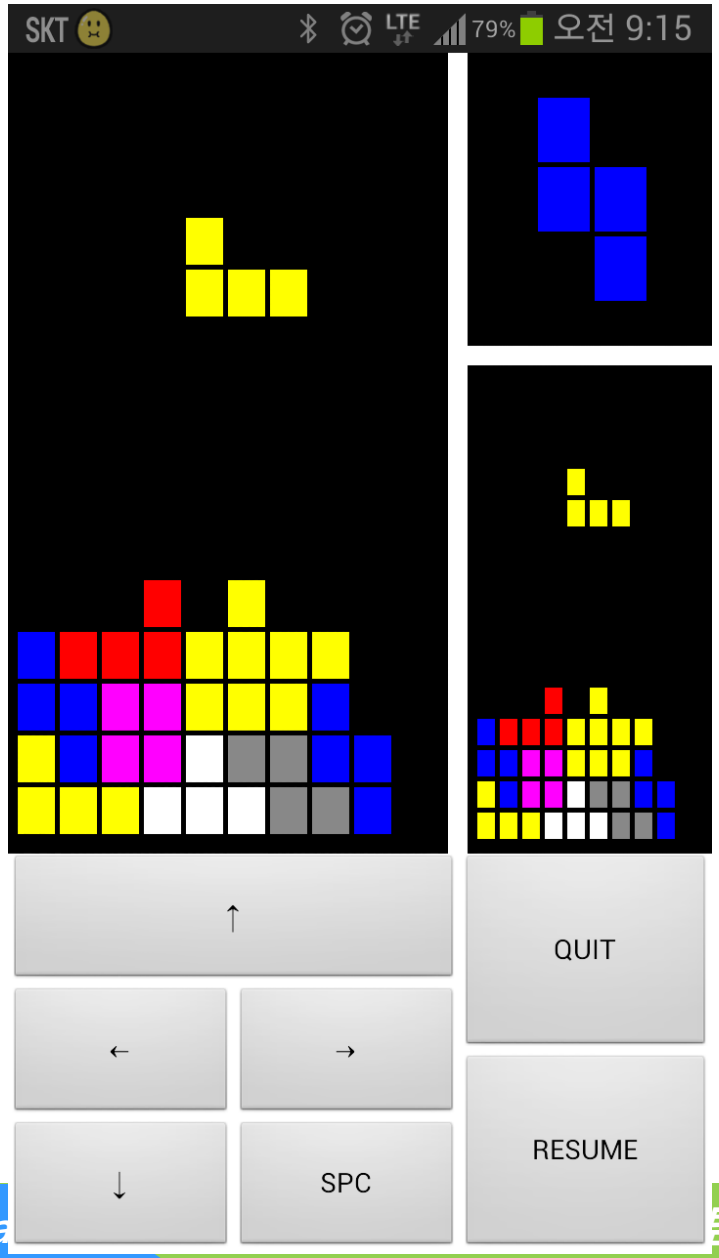
숭실대학교
김강희 교수
(khkim@ssu.ac.kr)

순서

- 이론:
  - 설계 단계들
  - 객체 모델링과 시나리오 열거
- 실습:
  - 단순 시나리오
  - 확장 시나리오
  - 과제

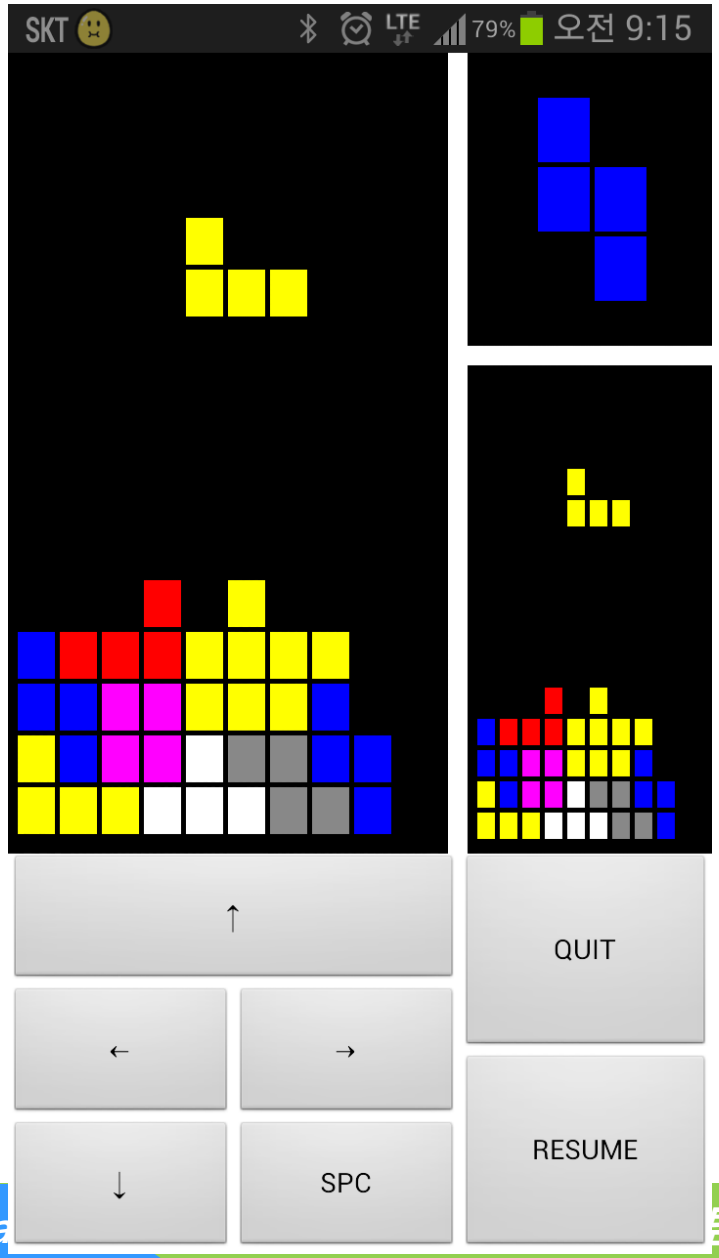
1단계: 설계 로드맵 구상

- 콘솔 환경
  - 1인용 흑백 테트리스
  - 1인용 흑백 테트리스 with a duplicated screen
- 안드로이드 환경
  - 1인용 컬러 테트리스 with a duplicated screen
  - 1인용 컬러 테트리스 with Echo server
    - 멀티 쓰레딩 + 소켓
  - 2인용 컬러 테트리스 with Tetris server
    - Tetris Server 프로그래밍 필요



2단계: Source tree 구상

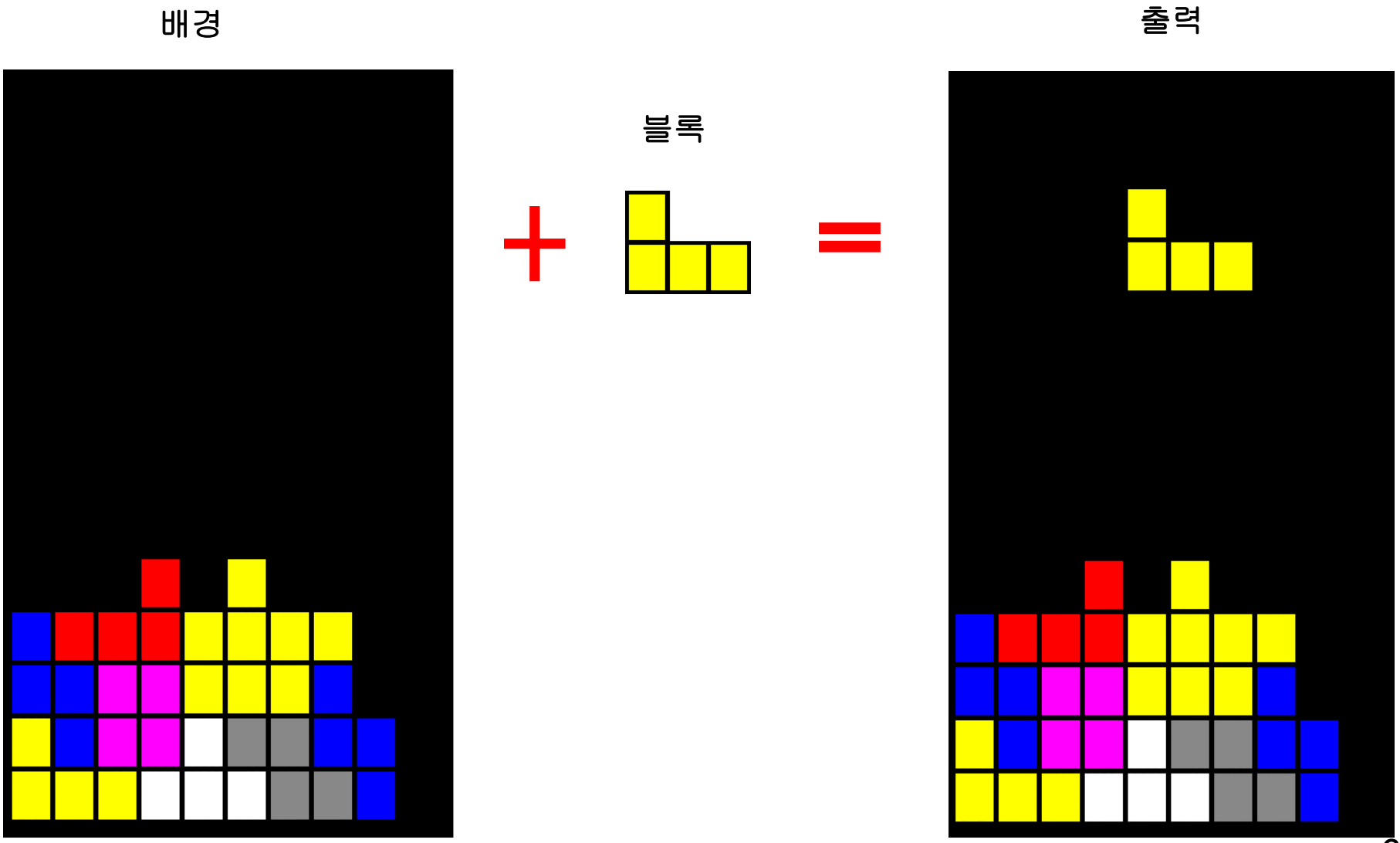
- 시스템 측면 (non-deterministic)
  - 키 입력, 화면 출력, 타이머 구동, 난수 발생 등
- 알고리즘 측면 (deterministic)
  - 블록 출현 후
    - 이동: 좌, 우, 아래, 추락
    - 회전(90도)
  - 블록 충돌
    - 좌/우 충돌
    - 아래/추락 충돌
    - 회전 충돌
  - 행 삭제
  - 배경 화면 갱신 및 신규 블록 출현



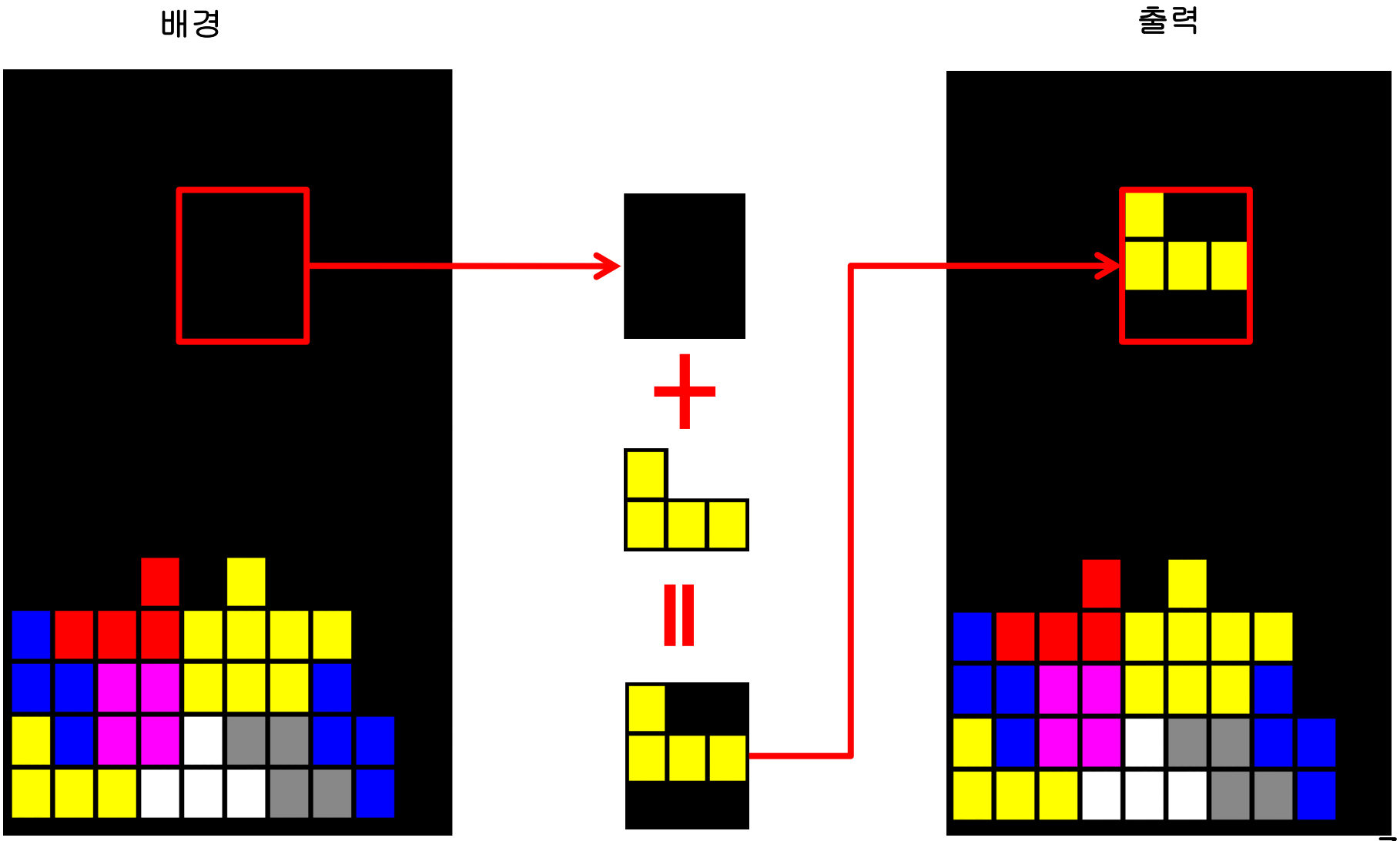
3단계: 객체 모델링 및 시나리오 열거

- 시나리오 **연산화** : common scenario 고려 (결과물: 객체 연산의 정의)
  - 주요 객체들 구상 : 배경, 벽, 쌓인 블록들, 내려오는 블록, …
  - 객체 추상화 : 서로 다른 성격의 객체들도 가능한 한 동일한 클래스로 취급할 수 있도록 클래스를 정의 (예: 배경, 벽, 7가지 블록들 → 행렬)
    - 이러한 '공격적인' 추상화가 시나리오 코딩에 미치는 영향을 추후에 신중히 검토해야 함!!
  - 객체 단순화 : 추상화된 클래스가 너무 복잡한 속성들로 정의되지 않도록 최대한 속성을 단순화 (예: 컬러 블록 → 흑백 블록)
- 시나리오들 **집합화** : 모든 시나리오 고려 (결과물: 순서도)
  - 가능한 시나리오들을 순서도 형태로 열거한다.
  - 각 시나리오를 선택하여 가급적이면 동일한 타입의 객체의 연산으로 표현한다.

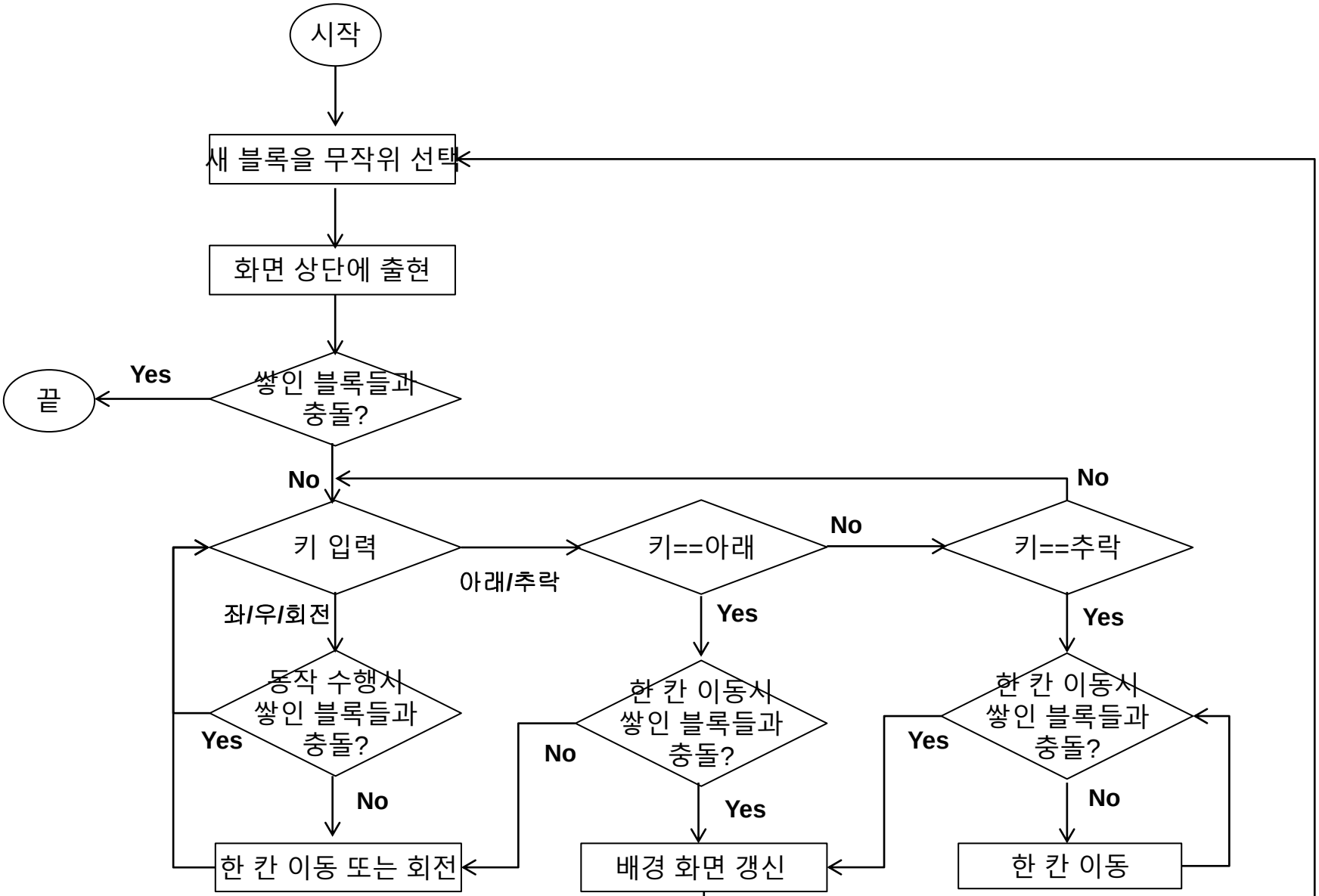
3단계: common scenario 연산화



3단계: common scenario 연산화



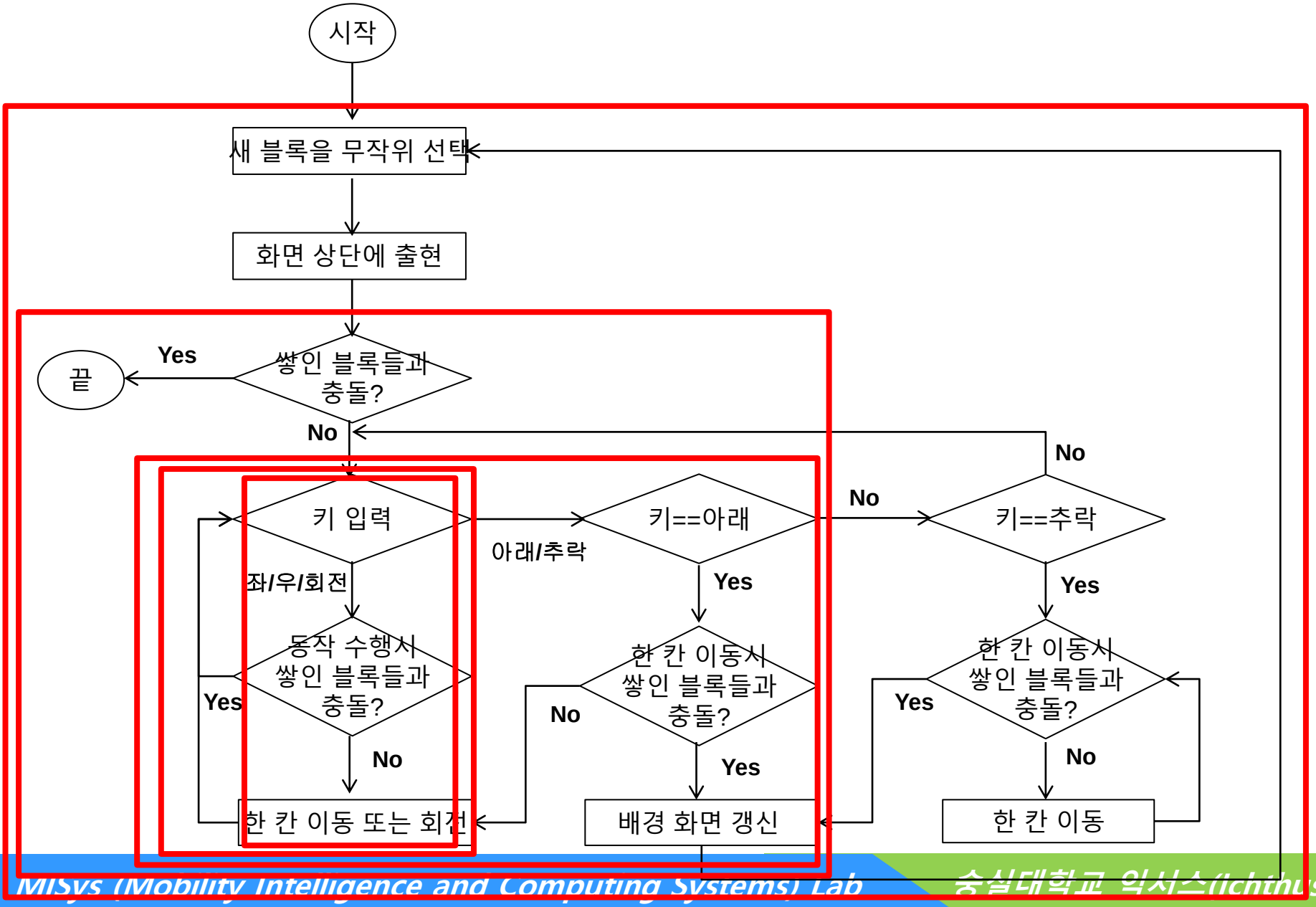
4단계: 모든 시나리오들을 순서도로 표현



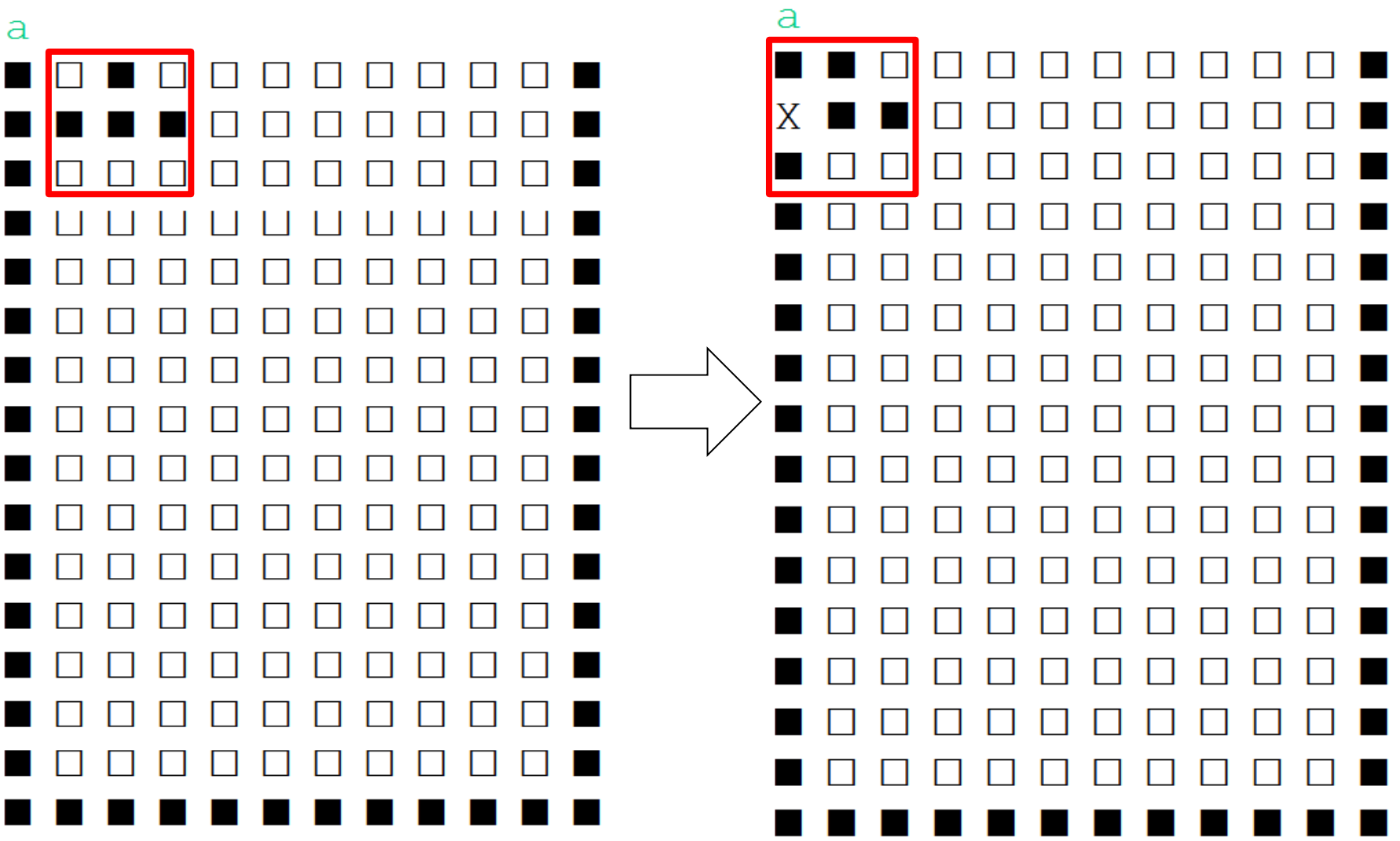
4단계: 단순 시나리오 코딩

```
int top = 0;
int left = 4;
Matrix *iScreen = new Matrix(arryScreen);
Matrix *currBlk = new Matrix(arrayBlk);
Matrix *tempBlk = iScreen->clip(top, left, top+currBlk.dy, left+currBlk.dx);
tempBlk = tempBlk->add(currBlk);
Matrix *oScreen = new Matrix(iScreen);
oScreen->paste(tempBlk, top, left);
```

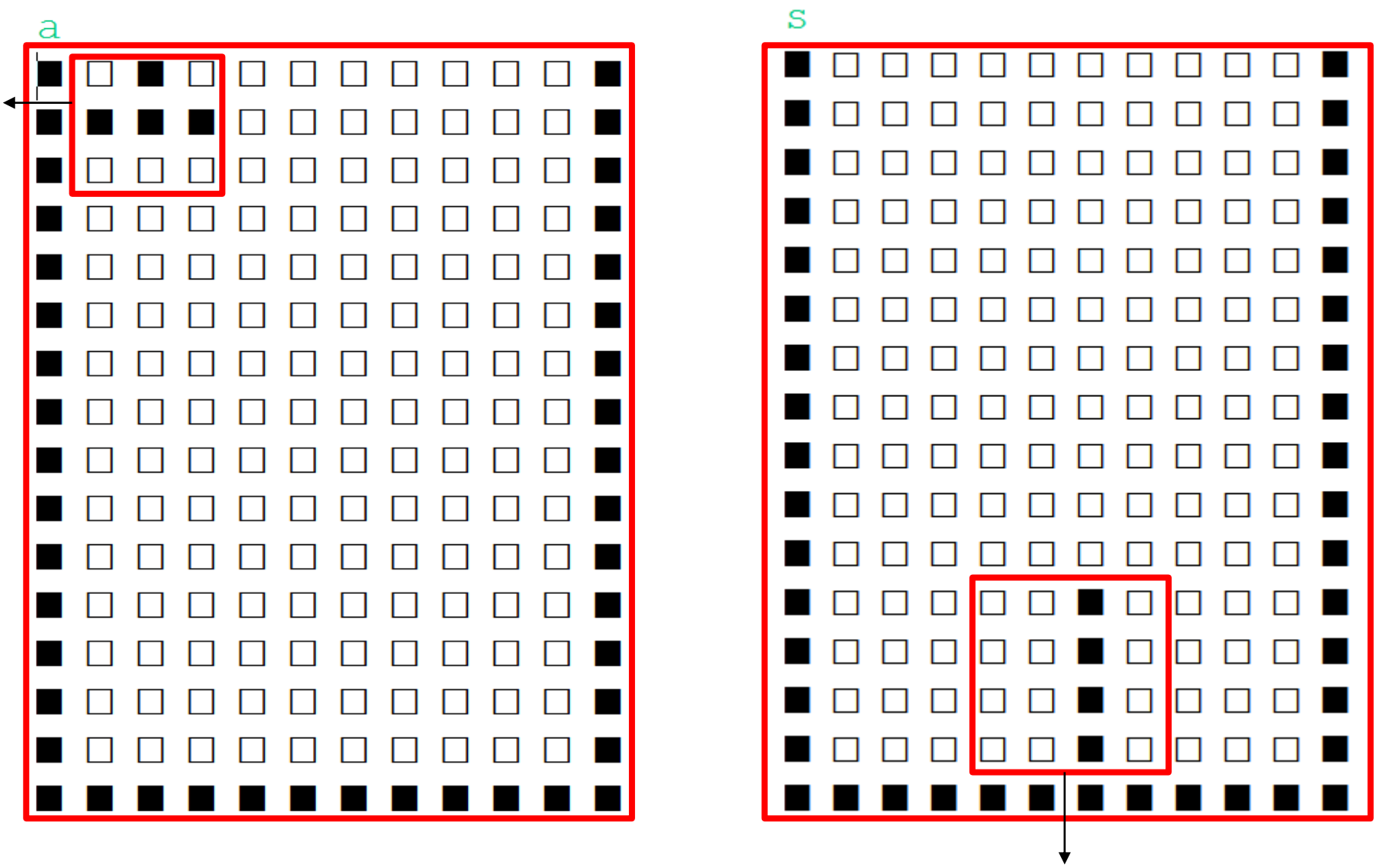
4단계: 시나리오 코딩 (안쪽 → 바깥쪽)



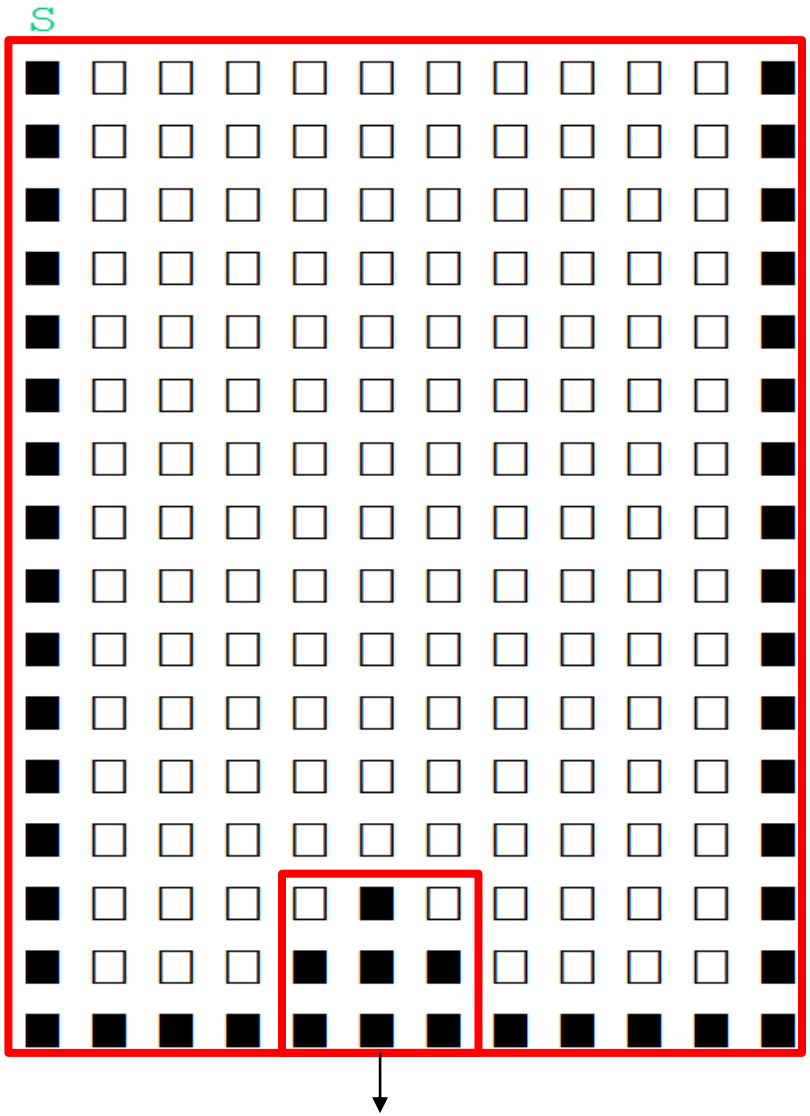
벽에 충돌하면?

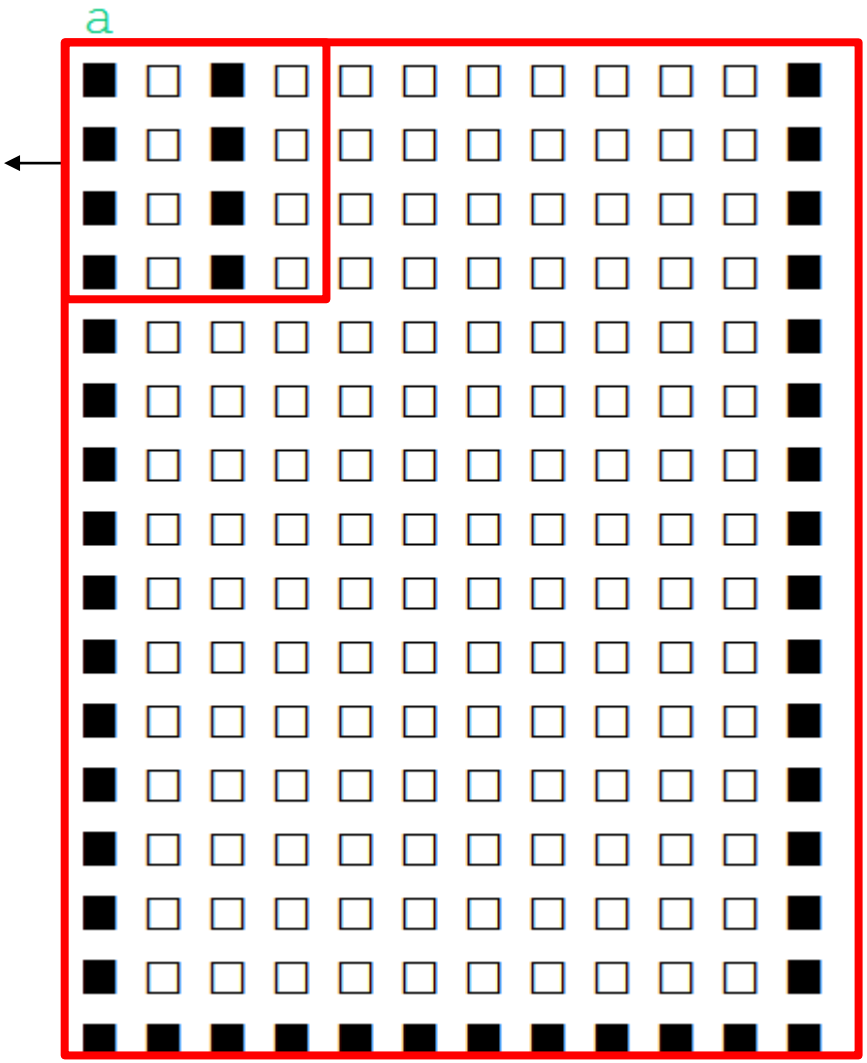


블록 경계에서 충돌이 발생하는 경우?

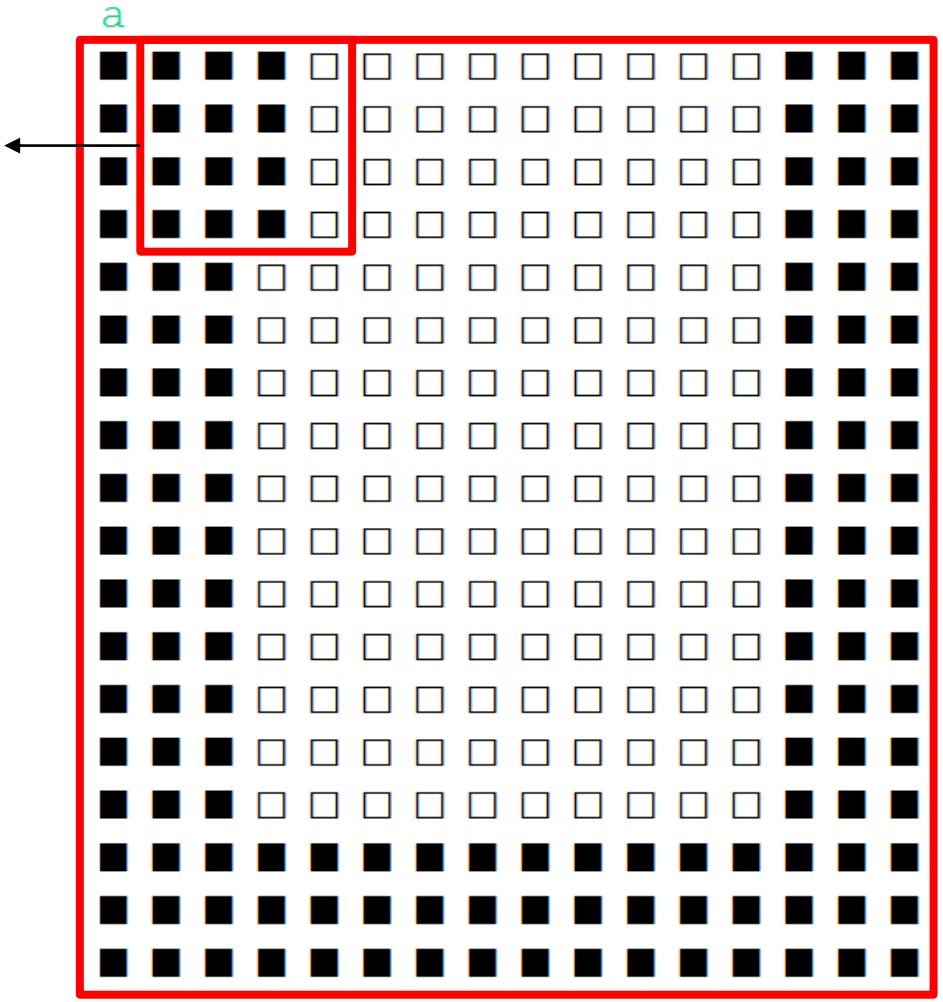


블록 내부에서 충돌이 발생하는 경우?

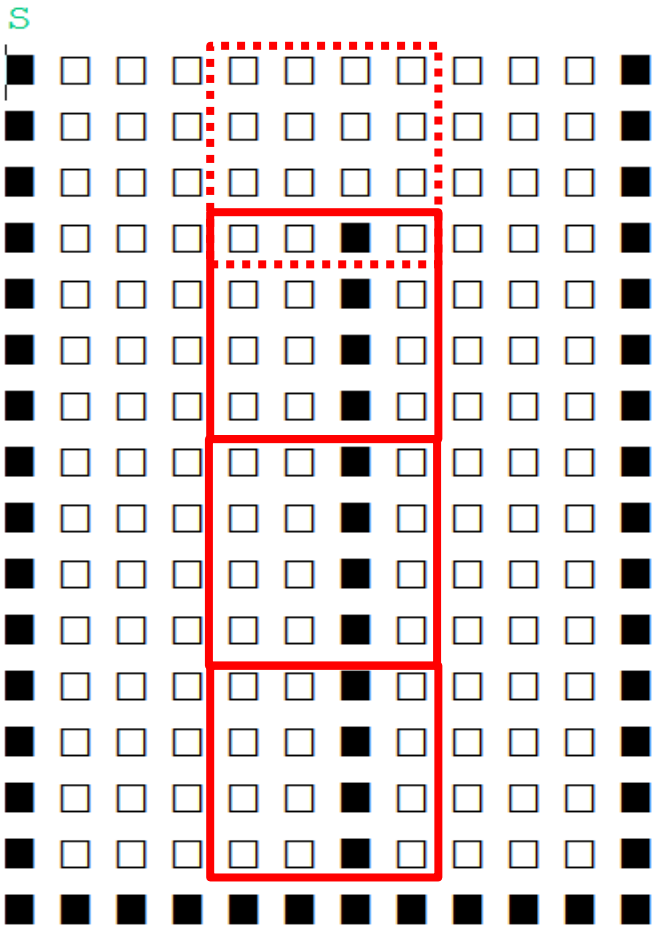




충돌에 대한 해결책?



게임 종료 조건



Game Over!

남은 숙제들

- 7가지 블록의 무작위 선택
  - 7가지 블록들을 Matrix 객체들로 미리 생성함
  - 이 객체들을 필요할 때마다 난수를 발생시켜 선택함
- Rotate키 처리:
  - Rotate의 결과를 표현하는 Matrix 객체들을 미리 생성함

```
idxBlockDegree = (idxBlockDegree + 1) % 4;
Matrix currBlk = setOfBlockObjects[idxBlockType][idxBlockDegree];
```

- Space키 처리 (블록 추락)
  - 바닥에 충돌할 때까지 아래로 한 칸씩 이동함을 루프 형태로 반복함
  - 한 칸씩 이동한 결과 생성되는 oScreen을 화면에 출력하지 말되, 마지막으로 블록이 바닥에 인접한 장면의 oScreen은 출력해야 함
- Full line 삭제
  - Down키 입력시 충돌 있고, full line들이 발견되면 해당 line들을 배경 화면에서 삭제함 : Matrix class의 sum, clip, paste 메쏘드 이용하고, screen 배열의 검색 범위를 줄이는 것에 유의해야 함