CSED232 Object Oriented Programming (Spring 2023)

Assignment #5

- GUI Programming -

Due date: 2023년 6월 9일 금요일 23시 59분

담당 조교: 양희민 (heeminid@postech.ac.kr)

주의 사항

- 클래스는 .h 파일과 .cpp 파일에 나눠서 구현해야 합니다.
- 문제에 따로 명시되어 있지 않아도 클래스의 생성자 (constructor), 소멸자 (destructor)는 필수적으로 구현되어야 합니다.
- 문법 사용에 제약은 없습니다.
- 문제의 구현 요구 사항을 꼼꼼히 읽어 보시기 바랍니다.

감점 사항

- 제출 기한에서 하루가 경과할 때마다 20%씩 감점
  - 6월 10일에 제출: 20% 감점, 6월 11일에 제출: 40% 감점, ...
- 컴파일이 되지 않으면 0점 처리
- 제출 방법 위반 시 감점

제출 방법

- 같이 제공된 Qt Creator 설치 방법을 참고해 주십시오. 해당 설치 방법대로 설치된 Qt Creator 상에서 컴파일 및 채점이 진행될 것입니다.
- Qt Creator의 프로젝트 폴더를 그대로 압축하여 제출해 주십시오. 프로젝트 폴더 내부의 .pro 파일이 반드시 포함되어 있어야 합니다.
- Qt Creator의 프로젝트 폴더와 레포트를 같이 압축하여 "학번.zip"의 형태로 제출해 주십시오. 압축될 레포트의 이름 역시 "학번.pdf"의 형태로 설정해 주시기 바랍니다.

공통 채점 기준

1. 프로그램 기능
   - 프로그램이 요구 사항을 모두 만족하면서 올바르게 실행되는가?

2. 프로그램 설계 및 구현
   - 요구 사항을 만족하기 위한 변수 및 알고리즘 설계가 잘 되었는가?
   - 문제에서 제시된 세부 조건을 모두 만족하였는가?
   - 설계된 내용이 요구된 언어를 이용하여 적절히 구현되었는가?

3. 프로그램 가독성
   - 프로그램이 읽기 쉽고 이해하기 쉽게 작성되었는가?
   - 변수명이 무엇을 의미하는지 이해하기 쉬운가?
   - 프로그램의 소스 코드를 이해하기 쉽도록 주석을 잘 붙였는가?

4. 보고서 구성 및 내용, 양식
   - 보고서는 적절한 내용으로 이해하기 쉽고 보기 좋게 잘 작성되었는가?
   - 보고서의 양식을 잘 따랐는가?

다른 사람의 프로그램 혹은 인터넷에 있는 프로그램을 복사(copy)하거나 간단히 수정해서 제출하면 학점은 무조건 'F'가 됩니다. 이러한 부정행위가 발견되면 학과에서 정한 기준에 따라 추가적인 불이익이 있을 수 있습니다.

2048 Puzzle Game

1. 과제 개요

이번 과제에서는 C++ GUI 개발 프레임워크인 Qt framework를 이용하여 2048 퍼즐 게임을 제작해 본다. Qt Creator 개발 환경을 통해 2048 퍼즐게임을 제작하여 GUI 제작 경험을 쌓는 것을 목표로 한다.

2. 과제 설명

2.1 주의 사항

구현 요구 사항을 설명하기에 앞서, 이번 과제를 구현함에 있어서 반드시 지켜야 할 사항들을 명시한다.

- **GUI 구현 코드와 게임 내부 로직 구현 코드는 반드시 분리하여 구현한다.** 이는 게임 내부 로직 구현 코드와 GUI 구현 코드를 분리함으로써 각 객체들의 독립성을 높이고 구현의 편의성을 높이기 위함이다. 가장 상위 폴더에 .pro 파일과 main.cpp 함수가 위치하도록 하고 GUI 구현 코드는 ui 폴더에 위치하도록, 게임 내부 로직 구현 코드는 game 폴더에 위치하도록 한다.
- **Designer UI (.ui 파일)를 통한 GUI 구현을 금지한다.** GUI 구현에 필요한 세부 구현 사항들 (블록의 크기, 게임판의 크기 등)은 문제에서 주어지므로 이를 가지고 코드를 통해 GUI를 구현하도록 한다.

2.2 게임 진행

이번 섹션에서는 과제의 게임이 진행되는 과정을 서술한다. 세부적인 구현 요구 사항은 다음 섹션에서 서술한다. 구현 이전에 https://www.mathsisfun.com/games/2048.html 에서 게임을 해보며 게임의 전반적인 흐름을 파악하는 것을 추천한다.

2.2.1 게임 시작

프로그램을 실행하면 다음과 같이 랜덤한 게임판이 나타난다.

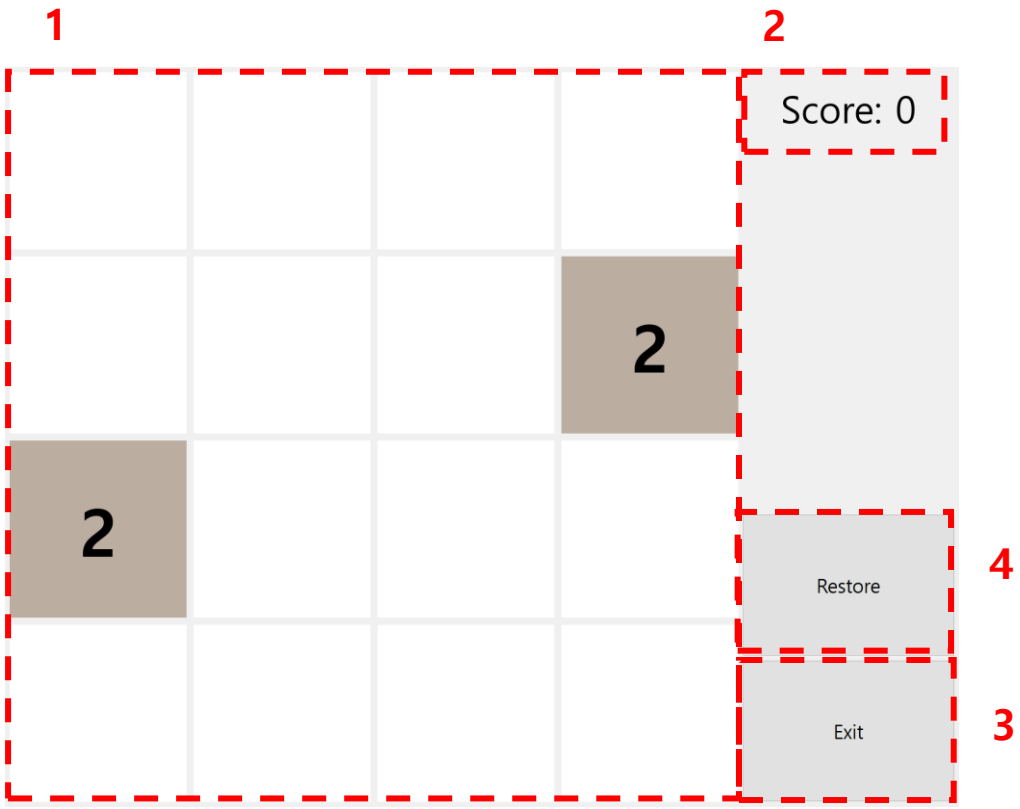


그림 1. 프로그램 시작 시 나타나는 화면

4 x 4 격자 형태의 게임판 내부 랜덤한 2군데에 숫자 2 블록이 등장하는 것으로 게임이 시작된다. 전체 레이아웃은 다음 네 개의 요소들로 구성된다.

1. 게임판**: 4 x 4 격자 형태의 공간**으로, 퍼즐 게임의 블록들이 합쳐지고 이동하는 공간이다.
2. 점수: 현재 게임판의 점수가 표시되는 공간이다. 점수를 계산하는 공식은 추후에 설명한다.
3. 게임 종료 버튼: 게임을 중간에 종료할 수 있는 버튼이다. 해당 버튼을 누를 경우 정말로 게임을 종료할 것인지 묻는 확인창이 출력된다. **"Yes" 버튼을 누를 경우 프로그램이 종료되며 "No" 버튼을 누르거나 확인창을 닫을 경우 아무 일도 일어나지 않는다.**
4. 게임판 되돌리기 버튼: **해당 버튼을 누를 경우 게임판이 이전의 상태로 되돌아간다.** 한 게임 당 되돌리기 기회는 3번 주어진다.

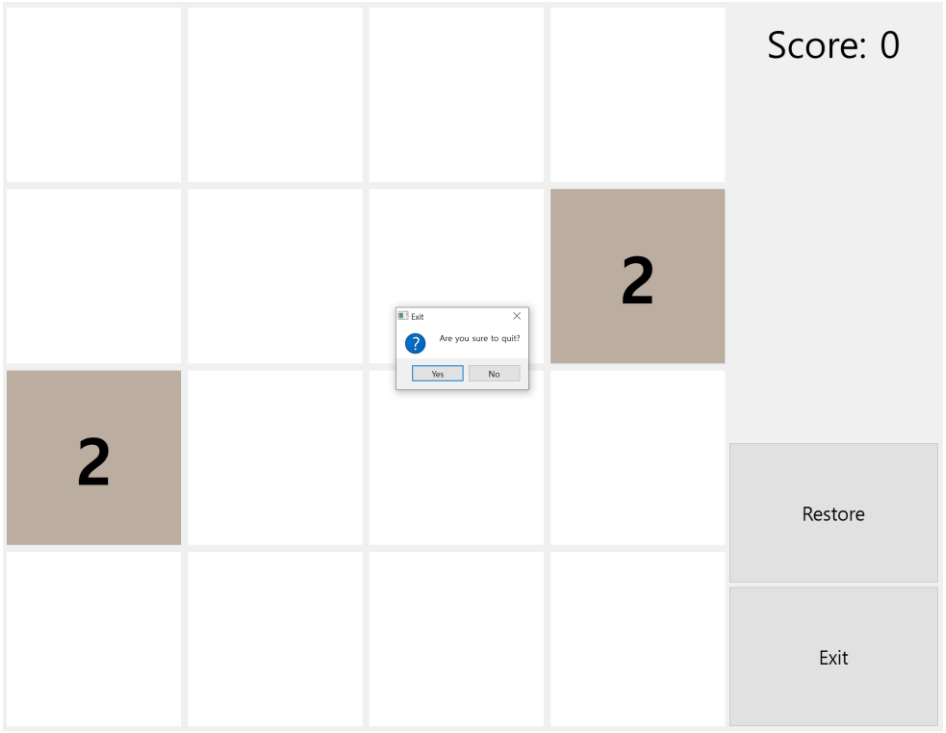


그림 2. 게임 종료 버튼을 누를 경우 나타나는 확인창

그림 2는 게임 종료 버튼을 눌렀을 때 나오는 확인창이다. Yes를 누를 경우 프로그램이 그대로 종료되며, No를 누를 경우 아무 일도 일어나지 않고 게임으로 되돌아간다.

2.2.2 게임 조작 및 블록 이동

게임판에서의 조작은 키보드 방향키로 이루어진다. 사용자가 누른 방향키에 대한 방향으로 블록들이 이동하며 (예를 들어, 왼쪽 방향키를 눌렀을 때 왼쪽으로의 이동이 가능하다면 블록들이 왼쪽으로 이동), **방향키를 눌렀을 때 해당 방향으로의 이동이 불가능하면 그 입력을 무시한다. 방향키 이외의 입력은 무시하도록 구현한다.**

방향키를 눌렀을 때 블록들이 이동하는 규칙을 자세히 알아보자.

1. 입력된 방향에 대해 빈 공간이 존재하면 해당 블록은 그 방향으로 최대한 이동한다.
2. 블록이 이동하면서 다른 값을 갖는 블록과 맞닿을 경우 두 블록은 합쳐지지 않는다.
3. 블록이 이동하면서 같은 값을 갖는 블록과 맞닿을 경우 두 블록은 합쳐진다. 또한, 블록들은 합쳐질 때 한 번에 최대 두 개까지만 그 병합에 관여할 수 있다.

우선, 입력된 방향에 대해 빈 공간이 존재하면 블록들은 그 방향으로 최대한 이동한다. 그림 3을 참고하도록 한다.

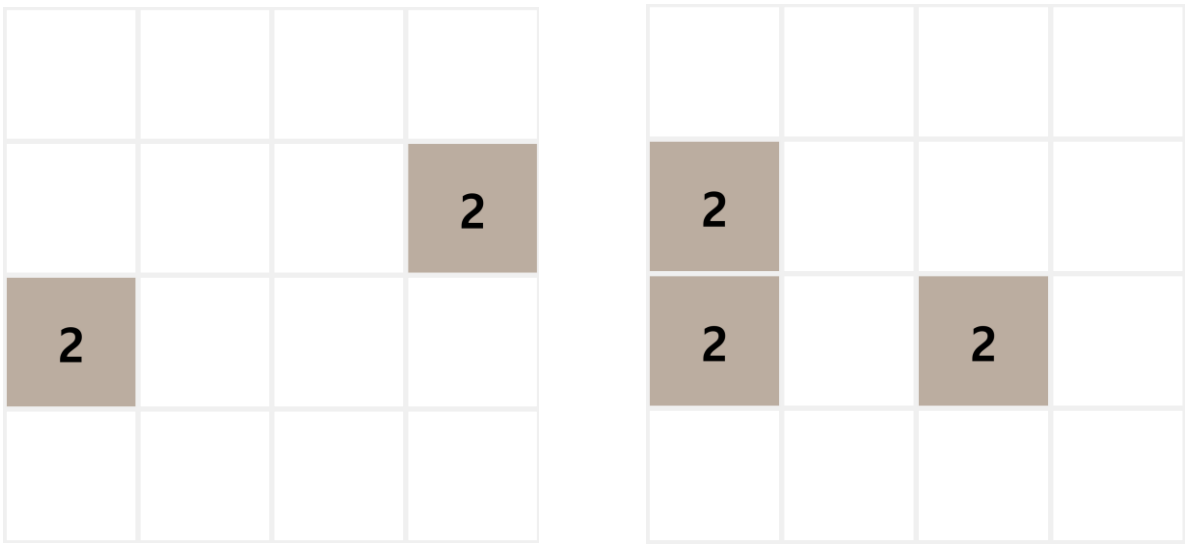


그림 3. (좌) 이동하기 전 게임판, (우) 왼쪽 방향키를 눌러서 블록들이 이동한 게임판. 3행 3열의 블록은 이동 후 새로 생성된 블록이다.

블록이 이동하면서 다른 값을 갖는 블록과 맞닿을 경우 두 블록은 합쳐지지 않고 충돌한다. 그림 4를 참고하도록 한다.

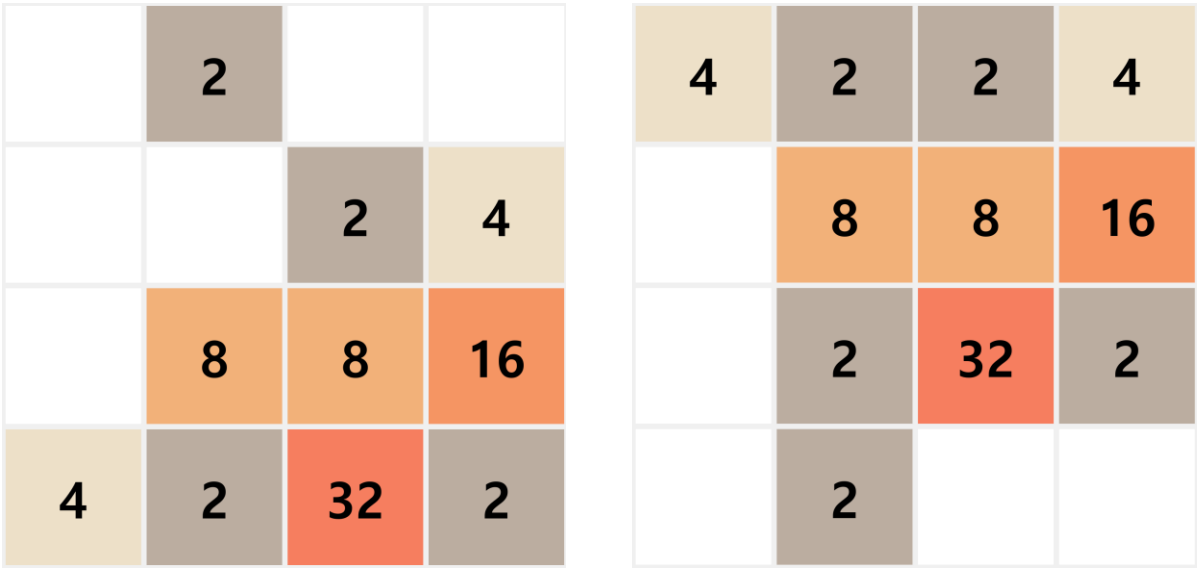


그림 4. (좌) 이동하기 전 게임판, (우) 위쪽 방향키를 눌러서 블록들이 이동한 게임판. (우)의 4행 2열 블록은 이동 후 새로 생성된 블록이다.

블록이 이동하면서 같은 값을 갖는 블록과 맞닿을 경우 두 블록은 합쳐진다. 그림 5를 보면 1행 1열의 블록과 2행 1열의 블록이 아래로 이동하면서 합쳐진 것을 확인할 수 있다. 또한, 블록들이 합쳐질 때 하나의 병합에 대해 최대 두 개까지만 관여할 수 있다. 그림 5를 보면 2-2-4 블록이 1열에 놓여 있는 것을 볼 수 있다. 여기서 2-2-4가 합쳐져서 8이 나오는 것이 아니라 4-4가 되는 것을 알 수 있다.

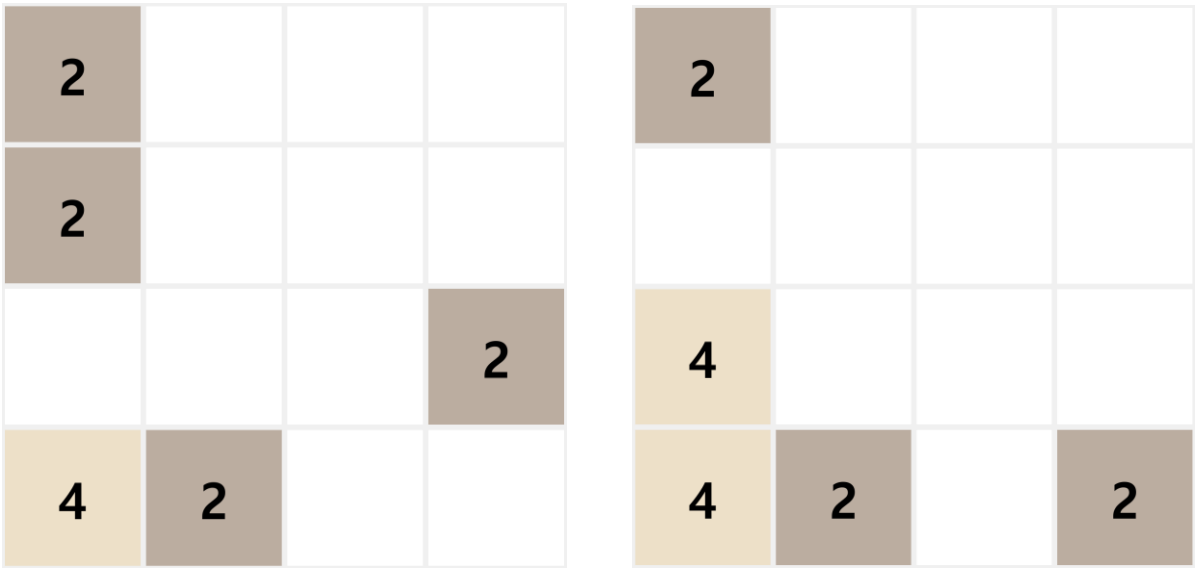


그림 5. (좌) 이동하기 전 게임판, (우) 아래쪽 방향키를 눌러서 블록들이 이동한 게임판. 1행 1열의 블록은 이동 후 새로 생성된 블록이다.

또한, 2-2-2가 가로로 연달아 있을 때 왼쪽으로 이동할 경우 합쳐져서 4-2가 되며, 오른쪽으로 이동할 경우 합쳐져서 2-4가 된다. 2-2-2가 세로로 연달아 있을 때도 마찬가지로, 위쪽으로 이동할 경우 위에서부터 4-2가 되며 아래쪽으로 이동할 경우 위에서부터 2-4가 된다.

2.2.3 점수 계산

블록들이 합쳐질 때 게임판의 점수가 올라간다. **합쳐져서 생성된 블록의 값만큼 점수가 올라가도록 구현한다.** 예를 들어, 8 블록과 8 블록을 합쳐서 16 블록을 만들었다면 점수는 16이 올라간다. 한 번의 이동으로 여러 개의 병합이 발생한다면 모든 병합에 대해 점수가 올라가도록 해야 한다.

2.2.4 새로운 블록 생성

게임판에서 블록들을 이동한 뒤에는 랜덤한 빈 공간에 새로운 블록이 하나 생성된다. 생성되는 새로운 블록은 정해진 확률에 따라 2 또는 4의 값을 가진다. 이번 과제를 구현할 때는 **20% 확률로 4의 값을 갖는 블록이 생성되도록, 80% 확률로 2의 값을 갖는 블록이 생성되도록 구현한다.**

2.2.5 되돌리기 버튼 (Restore)

되돌리기 버튼을 누르면 게임판이 이전의 상태로 되돌아간다. 버튼을 누르면 정말 되돌릴 것인지 묻는 메시지와 함께 현재 남은 되돌리기 기회를 알려주는 창이 출력된다 (그림 6).

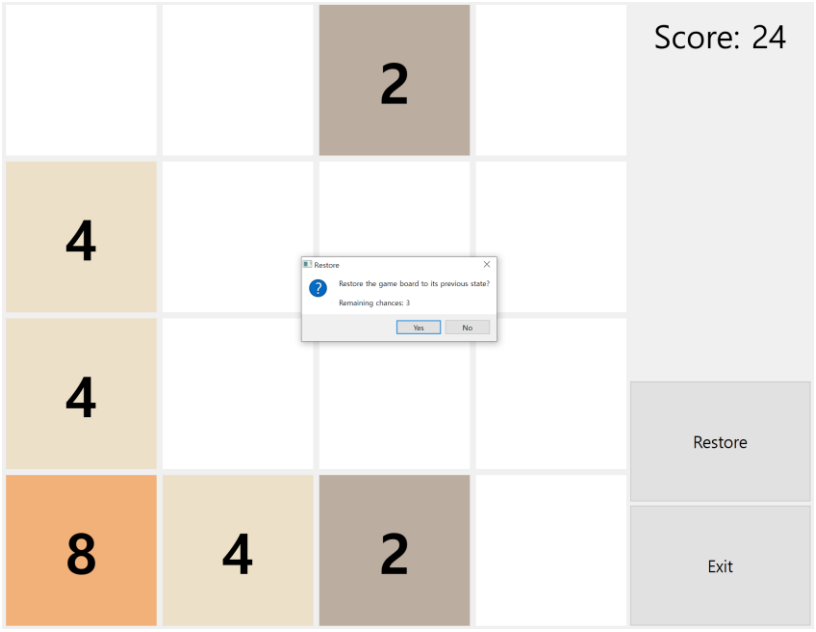


그림 6. 게임 진행 중 Restore 버튼을 눌렀을 때 출력되는 메시지 창

이 상태에서 Yes를 누를 경우 이전 상태로 되돌아가며 성공적으로 되돌려졌다는 메시지를 출력한다 (그림 7). No를 누를 경우 아무 일도 벌어지지 않는다.

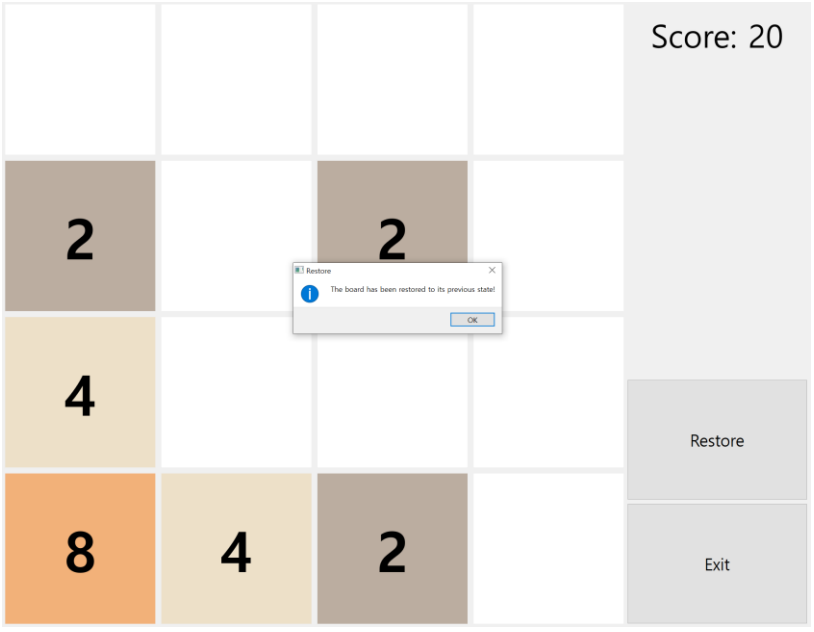


그림 7. 그림 6에서 Yes 버튼을 눌러 게임판을 되돌린 모습

되돌리기 버튼을 구현하기 위해서는 이전 게임판 상태를 버퍼에 저장해야 한다. 버퍼에 저장되는 게임판은 1개로 제한한다. 다시 말하자면, 되돌리기 버튼을 사용하여 버퍼에 저장된 게임판 상태로 되돌렸을 경우 더 이상 버퍼에는 아무 것도 남아있지 않을 것이다. 그렇기 때문에 되돌리기 버튼을 두 번 연속으로 사용할 수는 없다. 그리고 게임이 시작한 순간에도 버퍼에는 아무 것도 저장되어 있지 않을 것이다. 이렇게 되돌리기 버튼을 두 번 연속으로 사용하거나 게임이 시작된 순간에 되돌리기 버튼을 눌러서 버퍼에 아무 것도 없을 경우에는 그림 8과 같은 에러 창을 출력한다. 에러 창에서 OK 버튼을 누르면 다시 게임이 재개된다.

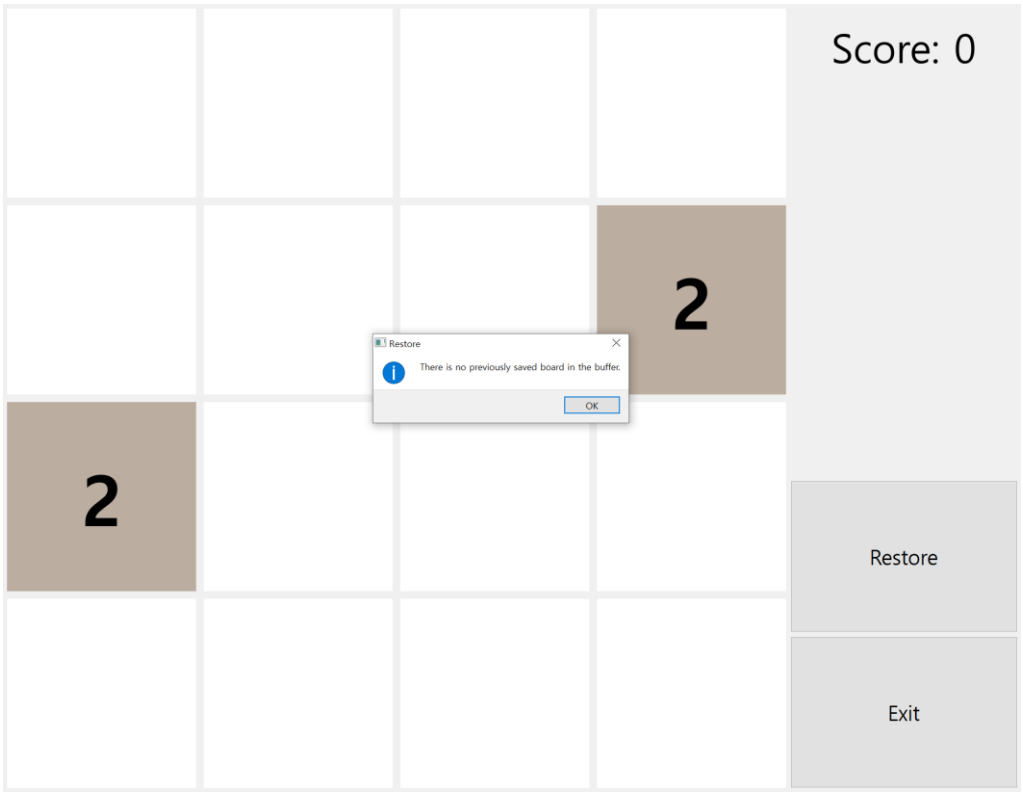


그림 8. 버퍼에 아무 것도 저장되어 있지 않은 상태에서 Restore 버튼을 눌렀을 때

또한, 더 이상 되돌리기 기회가 남아있지 않은 상태에서 Restore 버튼을 누르면 그림 9와 같은 에러 메시지 창을 출력한다.

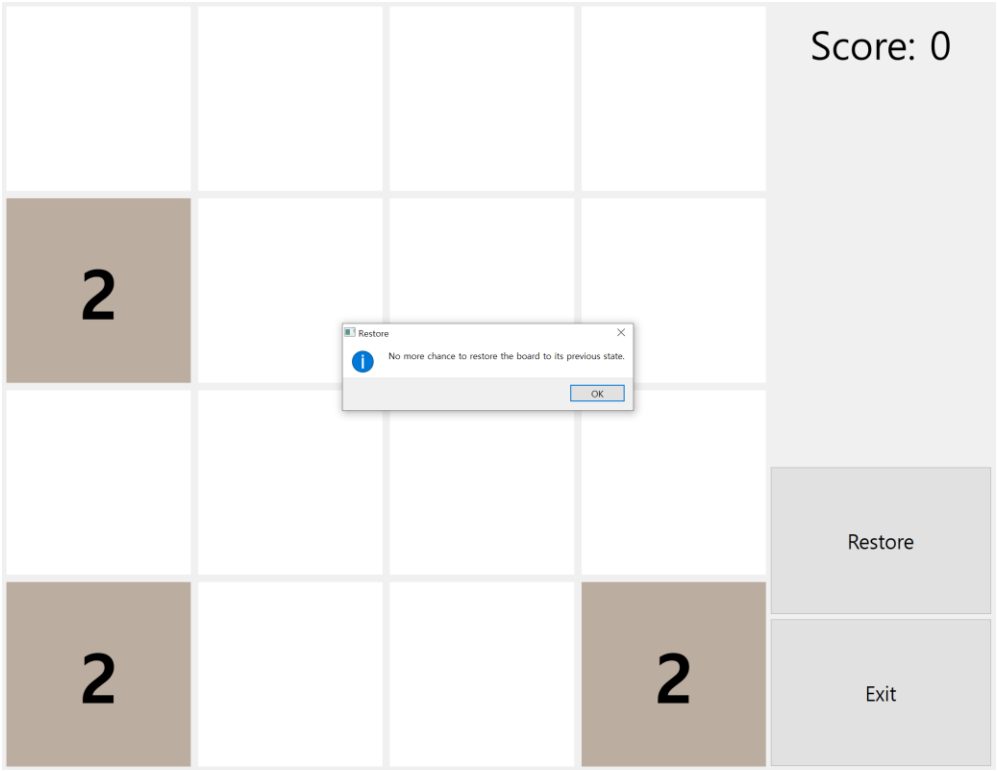


그림 9. 되돌리기 기회가 남지 않은 상태에서 Restore 버튼을 눌렀을 때 출력되는 메시지 창

2.2.6 게임 승리

게임을 진행하여 2048 블록을 만들면 유저가 승리한다. 조교는 게임을 하면서 2048을 만들지 못했기에 2048이 만들어져서 게임이 승리하는 예시를 보여주지는 못한다. 그 대신 코드를 살짝 변경하여 32가 생성되었을 때 게임이 종료되도록 코드를 바꿔놓았다. 보여주는 예시에서는 32 블록을 만들어서 게임이 끝나지만 실제 구현에서는 2048 블록이 만들어졌을 때 승리 조건이 달성되도록 구현한다. 그림 10을 보면 임의로 설정된 승리 조건인 32 블록이 만들어져 있다. 그림 10과 같이 **승리 조건에 해당하는 블록이 생성되면 1초 뒤에 "Congratulations!" 메시지와 함께 클리어 시점의 점수를 출력하는 메시지 박스를 출력한다.**

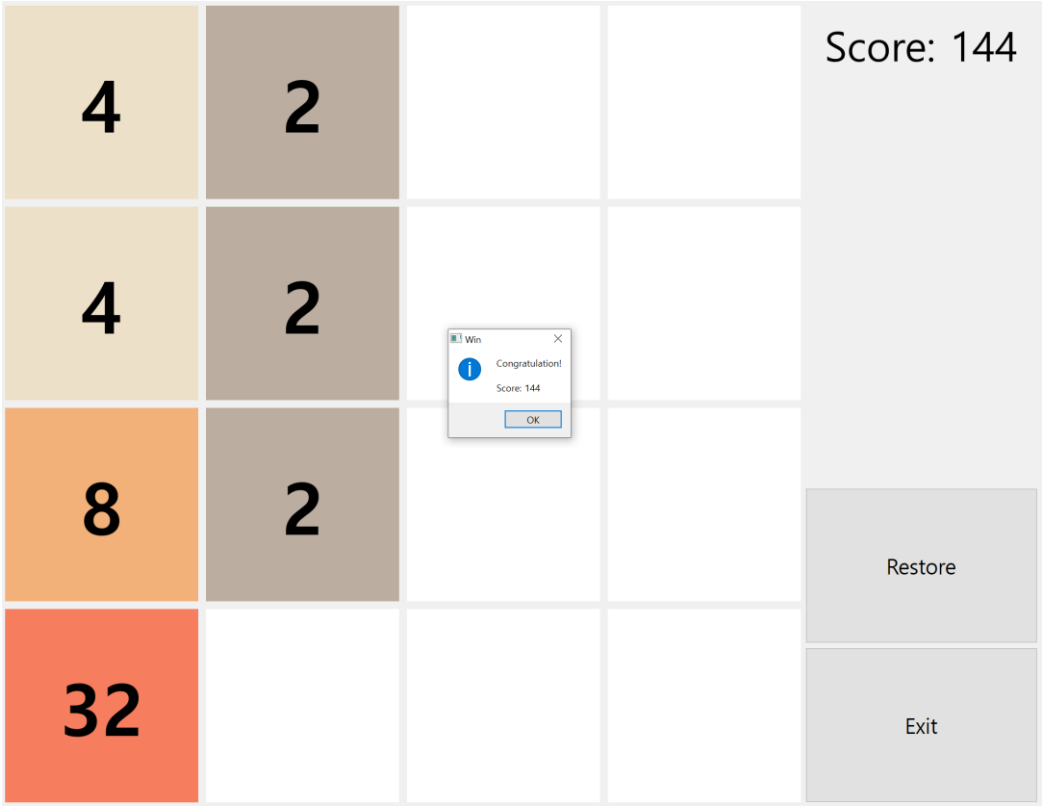


그림 10. 승리 조건이 달성되었을 때 출력되는 메시지 창

2.2.7 게임 패배

그 어떤 방향으로도 블록들이 이동할 수 없는 상태가 되면 게임에서 패배한다.

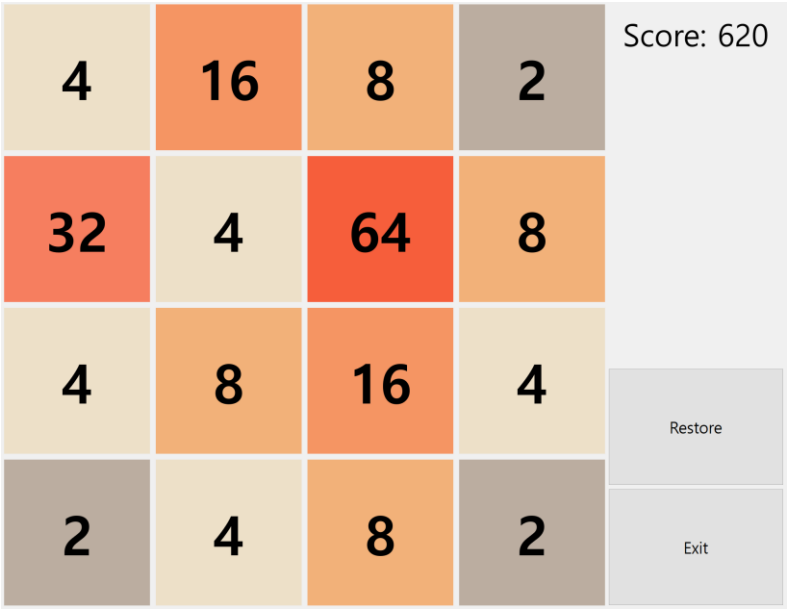


그림 11. 블록들이 어떤 방향으로도 이동할 수 없는 상태. 이 상태에 도달한 것만으로는 아무 일도 벌어지지 않는다.

그림 11을 보면 더 이상 어떤 방향으로도 이동할 수 없는 상태가 된 것을 알 수 있다. 이 상태에 도달했을 때는 게임 패배 메시지가 출력되지 않지만, 이 상태에서 아무 방향키를 누르면 그림 12와 같이 "You lose..." 메시지와 함께 패배 시점의 점수를 출력하는 메시지 박스를 출력한다.

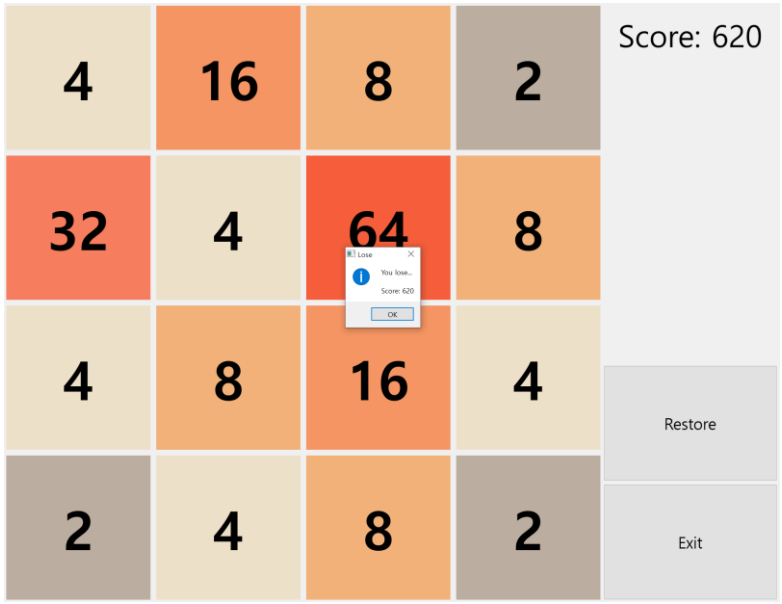


그림 12. 그림 11의 상태에서 방향키를 눌렀을 때 출력되는 메시지 창

2.3 GUI 구현 세부 사항

2.3.1 프로그램 레이아웃

- 프로그램 실행 창
  - 프로그램을 실행했을 때 나오는 전체 창의 크기: 가로 1300, 세로 1000
- 게임 판
  - QGridLayout으로 구현
  - 프로그램 창 왼쪽의 가로 1000, 세로 1000 공간에 위치
  - 게임판의 크기: 가로 1000, 세로 1000
- 점수 표시 레이아웃
  - QLabel로 구현
  - 프로그램 창 오른쪽 상단에 위치
  - 글자 크기 제한 없음
- 게임 종료 버튼 및 되돌리기 버튼
  - QPushButton으로 구현
  - 프로그램 창 오른쪽 하단에 위치
  - 버튼 크기: 가로 300, 세로 200
  - 폰트 크기: 20pt
  - 버튼에 존재하는 문구: 종료 버튼 (Exit), 되돌리기 버튼 (Restore)

2.3.2 블록

- 블록의 크기: 가로 250, 세로 250
- QLabel 클래스를 상속받아 구현
- setStyleSheet를 사용하여 블록 스타일 설정 (Widget의 스타일을 CSS로 설정해 주는 메서드)
  - setStyleSheet("QLabel { background: rgb(rrr, ggg, bbb); border-style: solid; font: 70pt; font: bold; }");

- 비어 있는 블록: rgb(255, 255, 255)
- 2 블록: rgb(187, 173, 160)
- 4 블록: rgb(237, 224, 200)
- 8 블록: rgb(242, 177, 121)
- 16 블록: rgb(245, 149, 99)
- 32 블록: rgb(246, 126, 95)
- 64 블록: rgb(246, 94, 59)
- 128 블록: rgb(237, 207, 114)
- 256 블록: rgb(237, 204, 97)
- 512 블록: rgb(237, 200, 80)
- 1024 블록: rgb(237, 197, 63)
- 2048 블록: rgb(237, 194, 46)

2.3.3 메시지 창

- 메시지 창을 띄울 때는 QMessageBox를 활용할 것
- 문제에 메시지 내용이 정확하게 명시되어 있는 것을 제외하고는 메시지 창에 표시되는 문구는 자유롭게 설정 가능함. 해당 상황에 맞는 뜻을 전할 수만 있으면 OK

2.4 필수 구현 파일 및 클래스

이번 섹션에서는 이번 과제에서 필수적으로 구현되어야 하는 파일 및 클래스를 설명한다. 이번 섹션에서 구현을 요구하는 클래스들은 반드시 구현되어야 하며, 이외의 추가적인 클래스나 파일은 자유롭게 구현이 가능하다. 보고서를 작성할 때 본인이 추가로 구현한 클래스나 파일이 있으면 해당 클래스 및 파일을 추가한 이유를 서술해야 한다. **GUI 구현 코드와 게임 내부 로직 구현 코드를 분리하지 않는 것은 감점 사항이다.**

2.4.1 class Block

게임 내부 로직 구현에 사용되는 Block 클래스가 구현되어 있어야 한다. Block 클래스는 각 블록이 가지고 있는 값을 관리한다.

2.4.2 class Board

게임 내부 로직 구현에 사용되는 Board 클래스가 구현되어 있어야 한다. Board 클래스는 Block들로 이루어져 있는 게임판을 관리하며, 게임판이 이동하는 로직 또한 이 클래스에 구현되어 있어야 한다.

2.4.3 class Game

게임을 전반적으로 관리하는 Game 클래스가 구현되어 있어야 한다. 게임에 사용되는 Board 클래스를 멤버 변수로 가지고 있어야 하며, 게임의 종료 여부를 판단하는 메서드를 구현한다.

2.4.4 class BlockUi

Block을 GUI 상에 나타내기 위한 BlockUi 클래스가 구현되어 있어야 한다. BlockUi는 QLabel 클래스를 상속받도록 하여 구현한다.

2.4.5 class GameUi

전체 레이아웃을 관리하는 GameUi 클래스가 구현되어 있어야 한다. QWidget을 상속받도록 하여 구현한다.

2.5 게임 진행 상황 저장

채점의 편의성을 위해 게임이 진행되는 상황을 "progress.txt" 파일에 저장하도록 한다. txt 파일은 main.cpp와 같은 곳에 저장하도록 하며, 저장 규칙은 다음과 같다.

1. 게임 시작 시 블록이 생성되는 위치 x1행 y1열, x2행 y2열: **INITIAL x1 y1 x2 y2**
   A. 1행 1열과 3행 2열에 생성되었을 경우 ➔ **INITIAL 1 1 3 2**

2. 위쪽/아래쪽/오른쪽/왼쪽을 누를 경우: **UP/DOWN/RIGHT/LEFT**

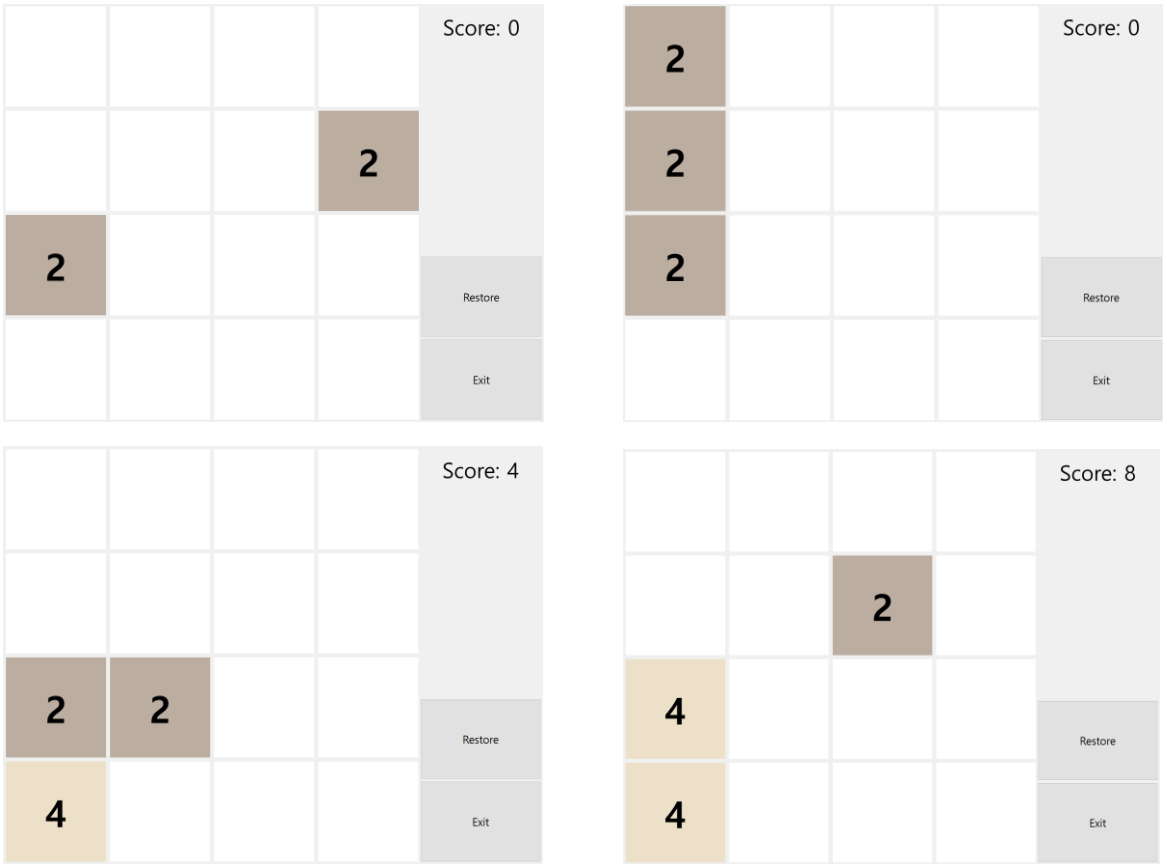
3. 이동 후 n 블록이 생성되는 위치 x행 y열: **GENERATE x y n**
   A. 이동 후 4 블록이 2행 1열에 생성되었을 경우 ➔ **GENERATE 2 1 4**

4. x행 y열에서 n 블록으로의 병합이 발생할 경우: **MERGE x y n**
   A. 2개의 4 블록이 2행 2열에서 8 블록으로 합쳐질 경우 ➔ **MERGE 2 2 8**

5. 되돌리기 버튼을 누르고 기회가 n번 남을 경우: **RESTORE n**
   A. 기회가 2번 남은 상태에서 되돌리기를 누를 경우 ➔ **RESTORE 1**

6. 이동 후 점수: **SCORE n**
   A. 이동 후에 점수가 72점일 경우: **SCORE 72**

예시) 좌상단->우상단->좌하단->우하단 순서로 움직임



INITIAL 2 4 3 1

LEFT

GENERATE 1 1 2

SCORE 0

DOWN

MERGE 4 1 4

GENERATE 3 2 2

SCORE 4

LEFT

MERGE 3 1 4

GENERATE 2 3 2

SCORE 8

처음 블록이 생성되는 것을 기록할 때는 숫자가 작은 행에서 생성된 위치를 우선적으로 기록하고, 행이 같을 경우 숫자가 작은 열에서 생성된 위치를 우선적으로 기록한다. 예를 들어, 그림과 같이 3행 1열과 2행 4열에서 첫 블록들이 생성되었을 경우 2행 4열이 행의 숫자가 작으므로 **INITIAL 2 4 3 1**로 기록한다.

병합이 여러 번 발생하였을 경우 숫자가 작은 행에서 발생한 병합을 우선적으로 기록하고, 행이 같을 경우 숫자가 작은 열에서 발생한 병합을 우선적으로 기록한다. 예를 들어, 2행 3열에서 병합이 발생하고 2행 2열에서 병합이 발생했을 경우

MERGE 2 2 n

MERGE 2 3 n'

의 순서로 기록된다.

하나의 이동에 대한 기록 순서는 이동 -> 병합 -> 생성 -> 점수로 정한다. 예를 들어 왼쪽으로 이동할 경우

LEFT

MERGE ... (발생한 병합 순서대로 쭉 기록)

GENERATE ... (이동 후 생성된 위치 기록)

SCORE ... (점수 기록)

방향키를 입력하였지만 해당 방향으로의 이동이 불가능 했을 경우에는 방향만 기록한다. 예를 들어 왼쪽과 오른쪽을 순서대로 눌렀는데 왼쪽으로의 이동이 불가능한 상황이었다고 해보자. 그럴 때는

LEFT

RIGHT

MERGE ...

GENERATE...

SCORE ...

로 기록하면 된다.

점수의 변화가 발생하지 않아도 이동 후에는 점수를 기록해야 한다. 점수를 기록하지 않아도 되는 상황은 해당 방향으로의 이동이 불가능한 경우밖에는 없다.

LEFT

GENERATE …

SCORE n

RIGHT

GENERATE …

SCORE n (MERGE가 발생하지 않아서 점수 변화가 없어도 기록해야 함)

RIGHT (이동이 불가능하기 때문에 점수 기록 X)

UP

MERGE …

GENERATE …

SCORE n'

TIPS

1. Qt 공식 문서 (https://doc.qt.io/)에 라이브러리 사용법에 대한 정리가 잘 되어 있으므로 해당 사이트를 통해 정보를 얻는 것을 추천한다.

2. 파일 추가나 파일 삭제를 하고 컴파일 에러를 겪는다면 Clean & Build를 해보자. Clean을 함으로써 이전에 프로젝트를 빌딩할 때 생성되었던 object 파일이나 바이너리 파일들을 삭제할 수 있다. Clean을 하고 Build를 하게 되면 clean한 상태에서 프로젝트를 다시 빌드하게 되므로 변경된 상태로부터 새로이 프로젝트를 빌드할 수 있게 된다.

3. .pro 파일은 Qt build system인 qmake에 사용되는 파일이다. 프로젝트를 제대로 빌드하기 위해서는 .pro 파일에 본인이 사용하는 .cpp 파일들과 .h 파일들을 명시해야 한다. 예를 들어, 본인이 game/mycode.cpp와 game/mycode.h를 추가했다면 .pro 파일의 SOURCES와 HEADERS에 추가된 파일들을 적어줘야 컴파일이 진행된다.

4. 레이아웃에 버튼을 추가하고 키보드 방향키를 누르면 게임판의 블록들이 움직이지 않을 수 있다. 이는 버튼을 추가했을 때 키보드의 focus가 버튼 쪽으로 설정되기 때문이다. 이를 해결하기 위해서는 QWidget 클래스의 focus policy 관련 문서를 읽어보는 것을 추천한다.