2023 Spring OOP Assignment Report

과제 번호 : 5
학번 : 20220100
이름 : 박기현
Povis ID : kihyun

명예서약 (Honor Code)

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.
I completed this programming task without the improper help of others.

1. 프로그램 개요

□ 본 과제에서는 C++ GUI(Graphical User Interface) 개발 프레임워크인 Qt framework를 이용하여 2048 퍼즐 게임을 제작한다. Qt Creator 개발 환경을 통해 2048 퍼즐 게임을 제작하여 GUI 제작 경험을 쌓는 것을 목표로 한다.

2. 프로그램의 구조 및 알고리즘

□ **main.cpp**

◆ "game.h"과 "gameui.h"을 헤더 파일을 포함한다.

◆ 게임과 게임 UI를 생성한다.

□ **gameui.h**

◆ <QDebug>, <QGridLayout>, <QLabel>, <QPushButton>, <QMessageBox>, <QKeyEvent>, <QThread>, "game/game.h" 헤더 파일을 포함한다.

◆ BlockUi

- 블록을 나타내는 Ui 클래스

◆ Win

- 승리 시 1초 지연과 GameUi에 승리 메시지 출력 신호를 전달하는 클래스

◆ GameUi

- 게임의 전반적인 Ui를 담당하는 클래스

□ gameui.cpp

- ◆ BlockUi Constructor – QLabel 4x4 배열을 생성하여 블록 값의 따라 맞는 숫자와 블록 색깔을 지정한다. 블록 값은 Game으로부터 전달 받는다. 설정된 블록 배열은 블록 레이아웃에 추가하여 창에 출력한다.
- ◆ BlockUi Destructor – 생성한 QLabel 4x4 배열을 동적 할당 해제한다.
- ◆ Win Constructor – GameUi 클래스의 객체를 전달 받는다.
- ◆ Win::run – 승리 조건이 성립하는 경우 1초 간 지연시키고, GameUi에 승리 메시지 출력 신호를 전달한다.
- ◆ GameUi Constructor – Game 클래스의 객체를 전달 받는다. 메인 레이아웃, 블록 레이아웃, 여백 레이아웃(점수, 복구 버튼, 나가기 버튼) QLayout 객체를 생성하고, 정렬을 설정한다. 점수 QLabel 객체를 생성하여 폰트를 설정하고, Game으로부터 전달 받은 점수를 출력한다. 복구 버튼과 나가기 버튼 QPushButton 객체를 생성하여, 정렬 및 폰트를 설정하고, 복구 버튼을 눌렀을 때와 나가기 버튼을 눌렀을 때의 신호를 연결한다. 생성된 복구 버튼과 나가기 버튼 객체는 여백 레이아웃에 추가한다. Win 객체를 생성하여 승리했을 경우의 신호와 승리 메시지 출력 함수를 연결한다. GameUi가 정상적으로 생성되면 game을 시작하고, 블록을 출력한다.
- ◆ GameUi Destructor – 생성자에서 생성한 객체를 동적 할당 해제하여 메모리 누수를 방지한다.
- ◆ GameUi::print_block – 새 BlockUi 객체를 생성하여 현재 상황의 블록을 출력한다.
- ◆ GameUi::score_update – 점수 Label의 점수를 업데이트한다.
- ◆ GameUi::lose_message – 패배 QMessageBox 객체를 생성하여 패배 문구와 점수를 출력하고, 종료한다.
- ◆ GameUi::win_message – 승리 QMessageBox 객체를 생성하여 승리 문구와 점수를 출력하고, 종료한다.
- ◆ GameUi::resotre_button_clicked – 복구 버튼을 누른 경우 수행되는 함수로, 복구 QMessageBox 객체를 생성하여 복구 여부 문구와 남은 복구 기회를 출력하고, 사용자의 Yes 또는 No 선택에 따라 동작을 수행한다. Yes를 누른 경우, 이전 버퍼에 기록된 보드판이 있고, 복구 기회가 남아있는 경우 이전 보드판으로 복구를 진행한다. 만약 이전 버퍼에 기록된 보드판이 없거나, 복구 기회가 남아있지 않은 경우에는 오류 메시지를 출력하고, 돌아간다. No 버튼을 누른 경우에도 역시 돌아간다.

- GameUi::exit_button_clicked – 나가기 버튼을 누른 경우 수행되는 함수로, 나가기 QMessageBox 객체를 생성하여 나가기 여부 문구를 출력하고, 사용자의 Yes 또는 No 선택에 따라 동작을 수행한다. Yes를 누른 경우에는 프로그램을 종료하고, No를 누른 경우에는 돌아간다.
- GameUi::keyPressedEvent – 방향키를 누른 경우 수행되는 함수로, 만약 패배 조건에 부합하는 경우에는 이후 동작을 수행하지 않고, 패배 문구 출력 함수를 호출한 뒤 종료한다. 그렇지 않은 경우에는 방향키에 따라 다음 동작을 수행한다. 동작을 수행하고 난 뒤, 새롭게 블록을 출력하고, 점수를 업데이트한다. 만약 승리 조건에 부합하는 경우에는 승리 Thread에서 확인하고, 신호를 전달하여 1초 지연 후 승리 메시지를 출력하고 종료한다.

□ game.h

- <iostream>, <fstream>, <cstdlib>, <ctime>, <conio.h> 헤더 파일을 포함한다.
- Block
  - 블록 값을 저장하는 클래스
- Board
  - 4x4 블록을 저장하는 클래스
  - 새 블록 값을 생성하고, 방향키에 따른 동작을 수행하는 함수를 멤버 함수로 가진다.
- Game
  - 전반적인 게임과 게임 보드 판을 관리하는 클래스
  - 복구를 위한 이전 보드 판을 버퍼에 저장하고, 점수와 복구 기회, 진행 여부와 승패 여부를 멤버 변수로 가진다. 또한, 과정을 출력하는 출력 파일 역시 멤버 변수로 가진다.
  - 게임을 시작하고, 진행하며, 전반적인 게임에 대한 함수를 멤버 함수로 가진다.

□ game.cpp

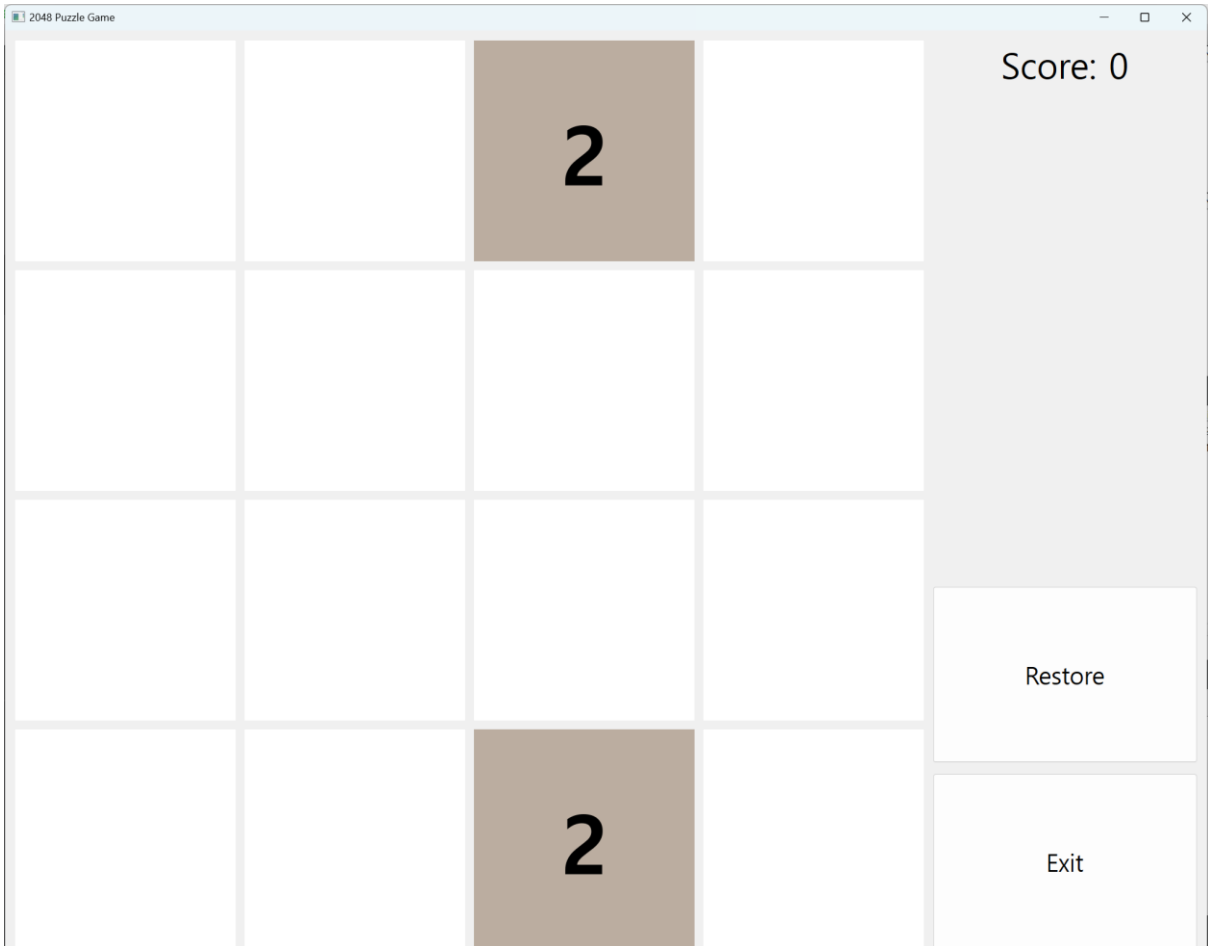
- Block Constructor – 블록 값을 0으로 초기화한다.
- Board Constructor – 4x4 블록을 생성하고, 모든 블록 값을 0으로 초기화한다.
- Board Destructor – 4x4 블록을 동적 할당 해제한다.

- Board::new_board – 새롭게 보드 판이 생성된 경우, 랜덤한 두 위치에 2 블록을 생성한다. 이때 과정 출력을 위해 생성된 위치를 반환한다.
- Board::new_value – 게임이 진행될 때마다 비어있는 랜덤한 위치에 일정 확률로 2 또는 4 블록을 생성한다. 이때 과정 출력을 위해 생성된 값과 위치 배열을 반한한다.
- Board::left – 왼쪽 방향키가 눌렸을 때 수행되는 함수로, 왼쪽으로 이동하는 게임 로직을 수행한다. 이때 더 이상 움직일 수 없는지, 블록이 생성될 수 있는 자리가 없는지를 확인하고, 병합이 이루어졌다면, 병합되어 생성된 블록 값의 총합을 점수 반영을 위해 반환한다.
- Board::right – 오른쪽 방향키가 눌렸을 때 수행되는 함수로, 동작 원리는 위와 같다.
- Board::up – 위쪽 방향키가 눌렸을 때 수행되는 함수로, 동작 원리는 위와 같다.
- Board::down – 아래쪽 방향키가 눌렸을 때 수행되는 함수로, 동작 원리는 위와 같다.
- Board::block_value – 보드 판 위치에 따른 블록 값을 반환한다.
- Board::change_value – 보드 판 해당 위치의 블록 값을 변환한다.
- Game Constructor – 과정 출력을 위한 파일을 열고, 새 게임 보드 판을 생성한다.
- Game Destructor – 사용한 게임 보드 판을 동적 할당 해제하고, 파일을 닫는다.
- Game::game_start – 게임 시작 시, 랜덤한 위치 두 곳에 2 블록을 생성한다.
- Game::game_continue – 게임 진행 시, 입력된 방향키에 맞는 함수를 수행하고, 게임 진행 여부와 게임 승패 여부를 업데이트한다. 게임이 진행되는 경우 비어있는 랜덤한 위치에 일정한 확률로 2 또는 4 블록을 생성한다.
- Game::block_value – 보드 판 위치에 따른 블록 값을 반환한다.
- Game::get_score – 현재 점수를 반환한다.
- Game::get_restore_chance – 현재 복구 기회를 반환한다.
- Game::use_chance – 복구 기회를 사용한다.
- Game::continue_or_end_check – 진행 여부를 확인한다.
- Game::lose_or_win_check – 승패 여부를 확인한다.

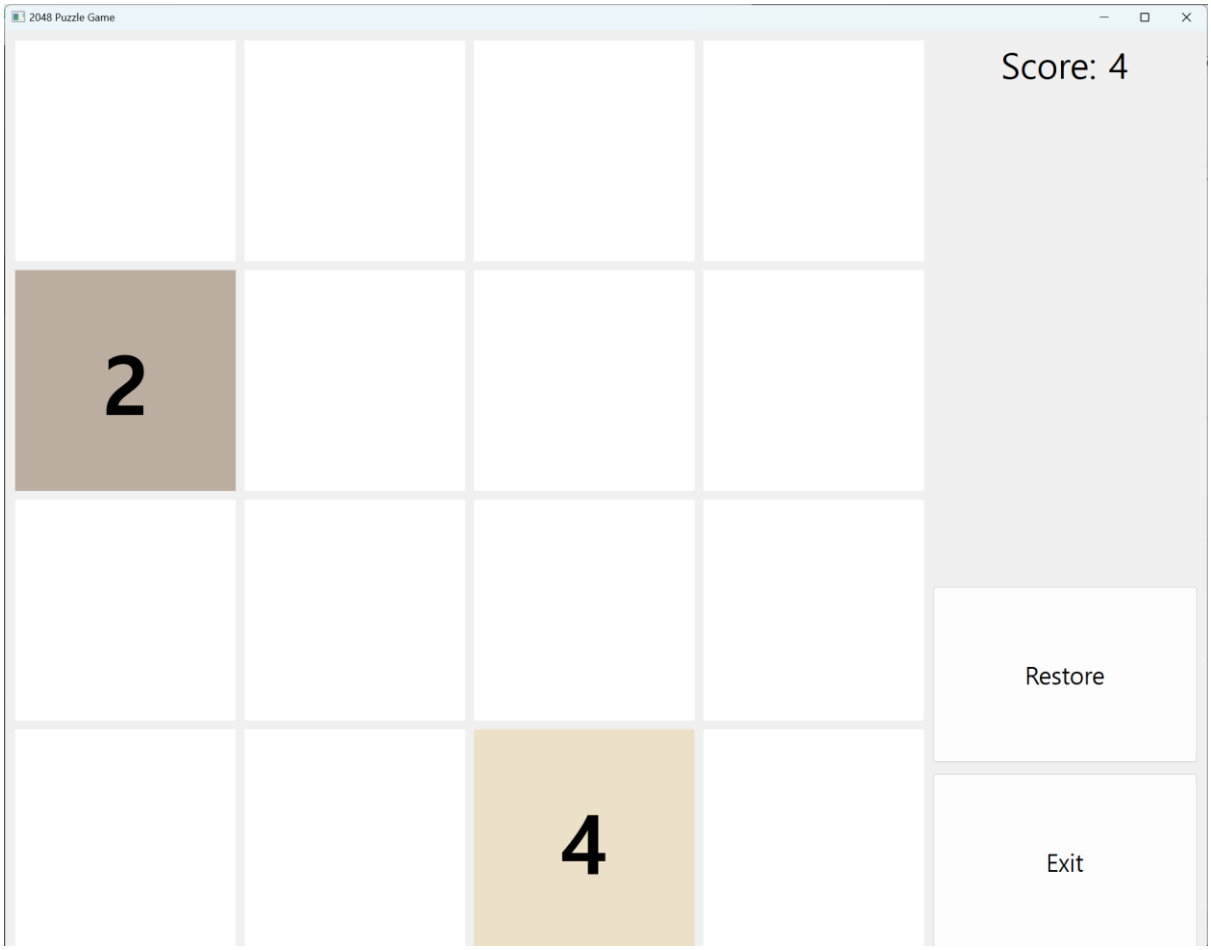
- Game::check_previous_board – 이전 보드 판이 버퍼에 존재하는지 확인한다.
- Game::restore_board – 버퍼에 저장된 이전 복구 판으로 게임 보드 판을 바꾼다.
- Game::add_score – 점수를 추가한다.
- Game::subtract_score – 복구 시 점수를 이전 점수로 되돌린다.
- Game::restore_state – 현재 상태(진행 여부 및 승패 여부)를 업데이트한다.

[프로그램 실행 예제]

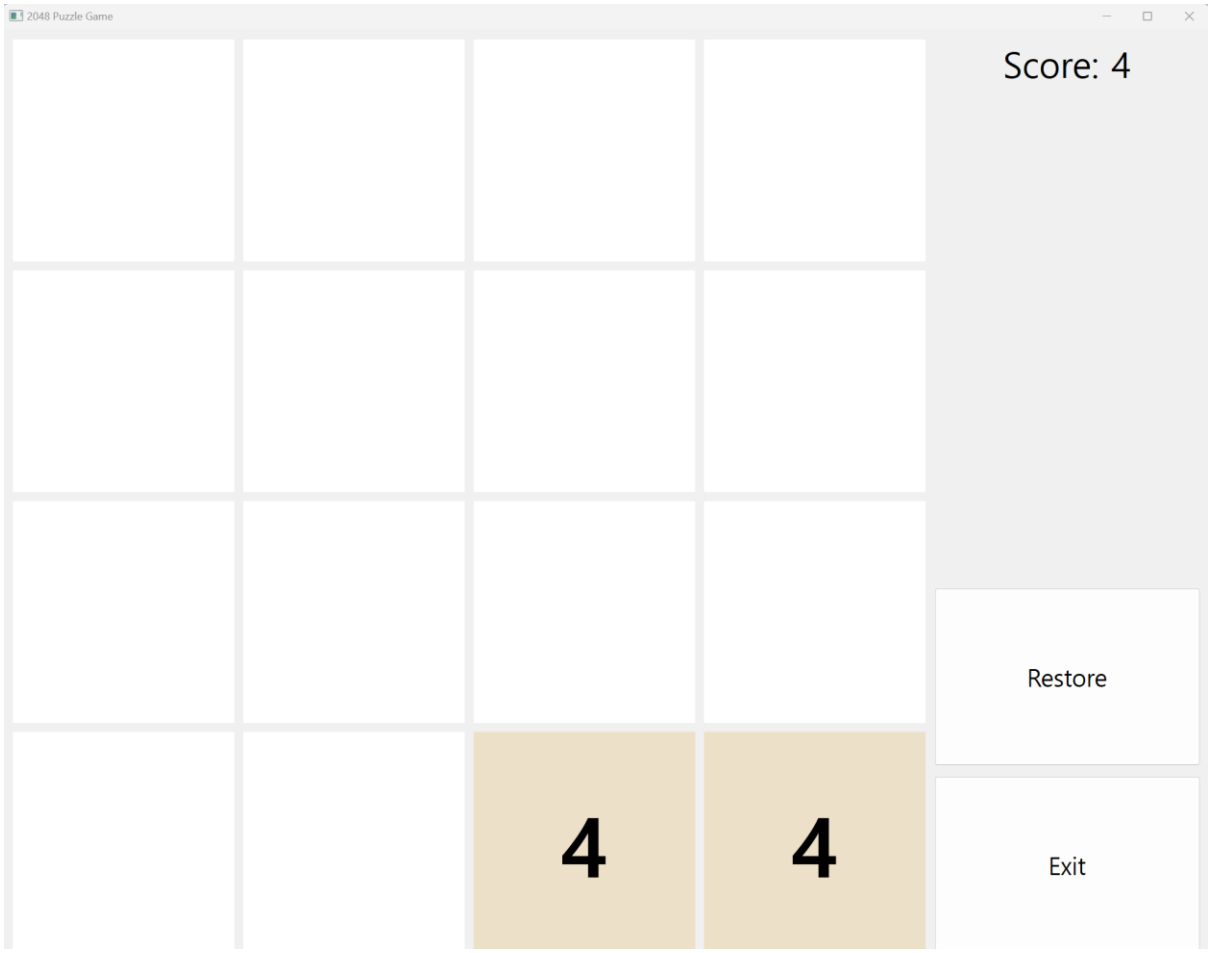
<처음 실행한 경우>



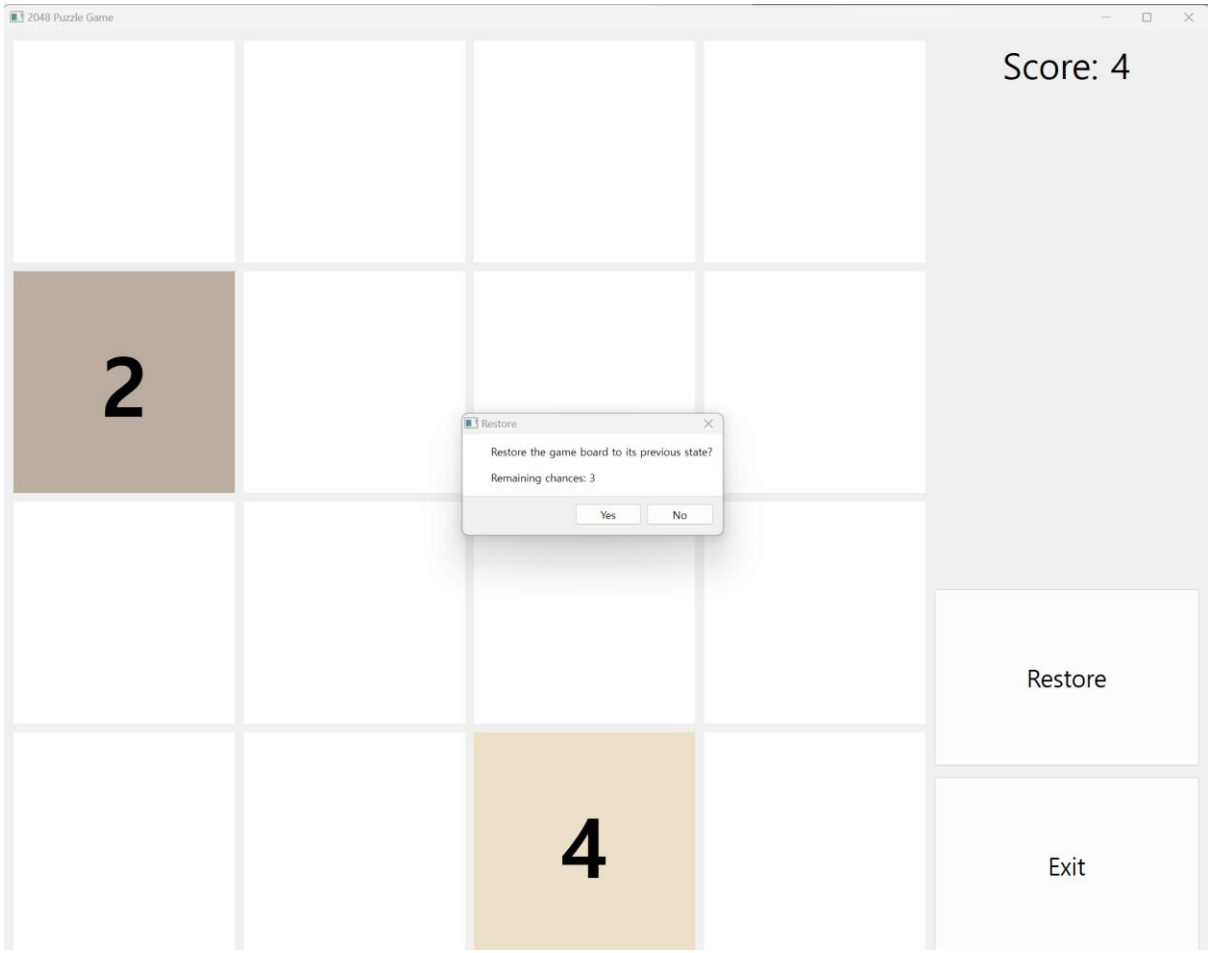
<아래 방향키를 눌러 기존의 블록이 병합되고, 새로운 2 블록이 생성된 경우>



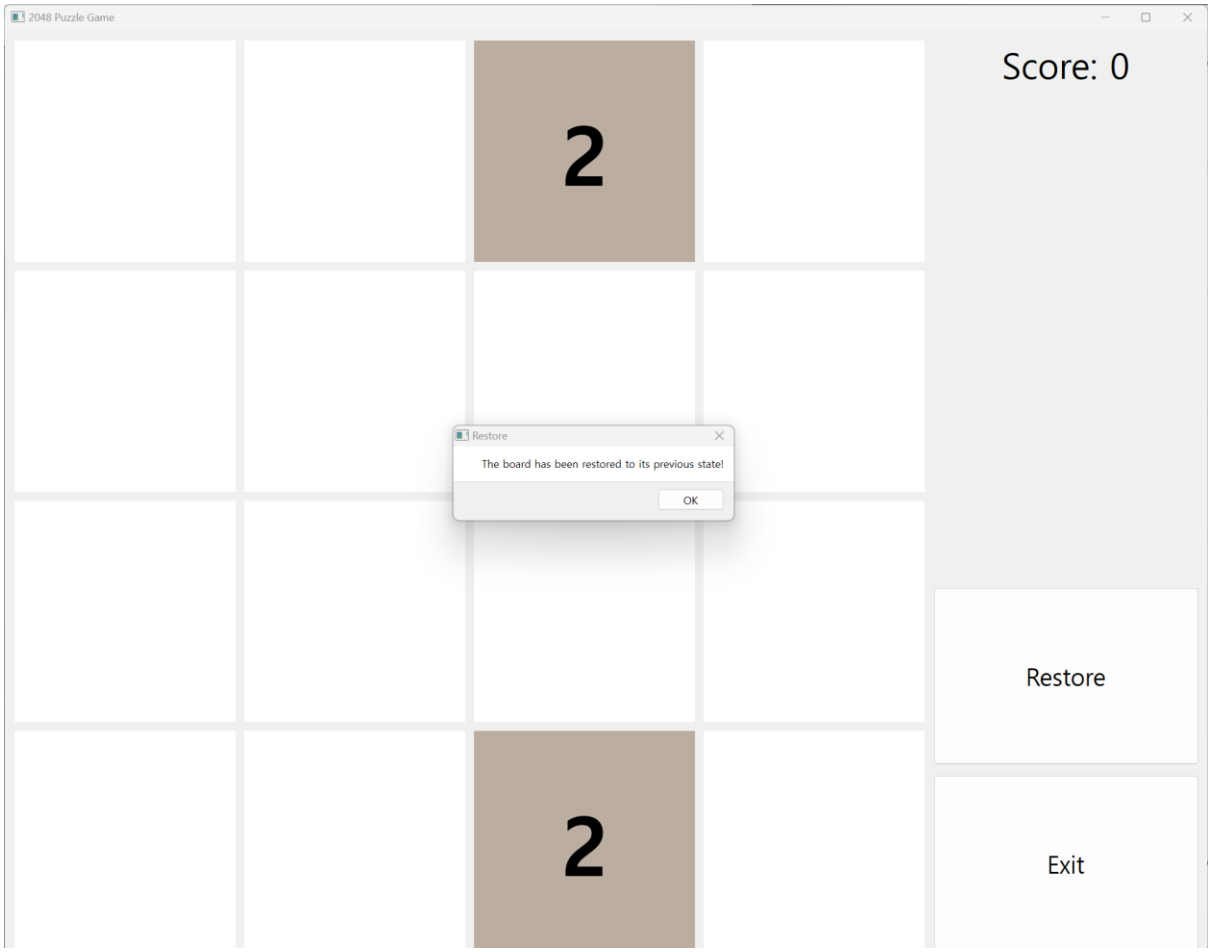
<아래 방향키를 눌러 기존의 블록이 병합되고, 새로운 4 블록이 생성된 경우>



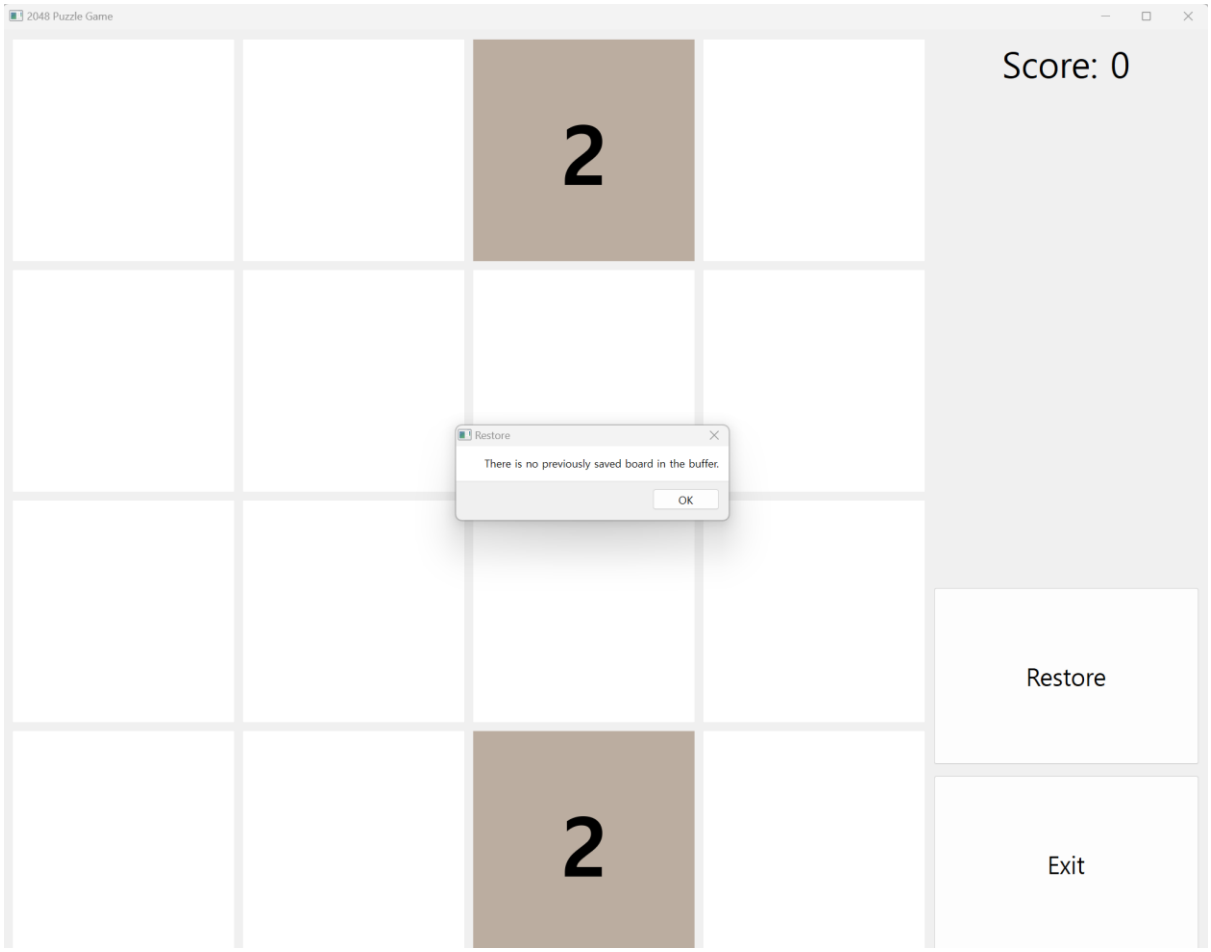
<복구 버튼을 누른 경우>


2048 Puzzle Game
Score: 4
2
Restore
Restore the game board to its previous state?
Remaining chances: 3
Yes
No
4
Restore
Exit

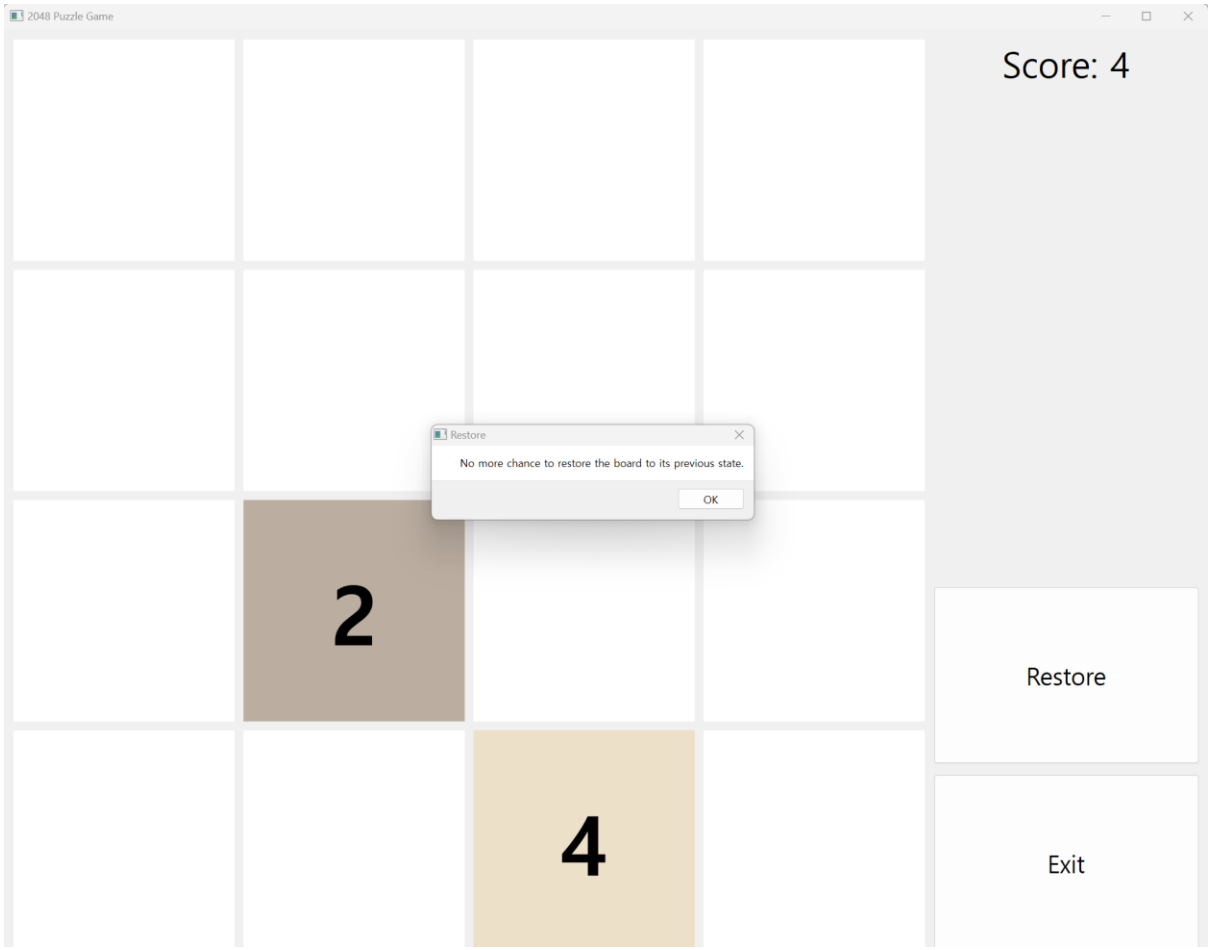
<복구를 진행한 경우>



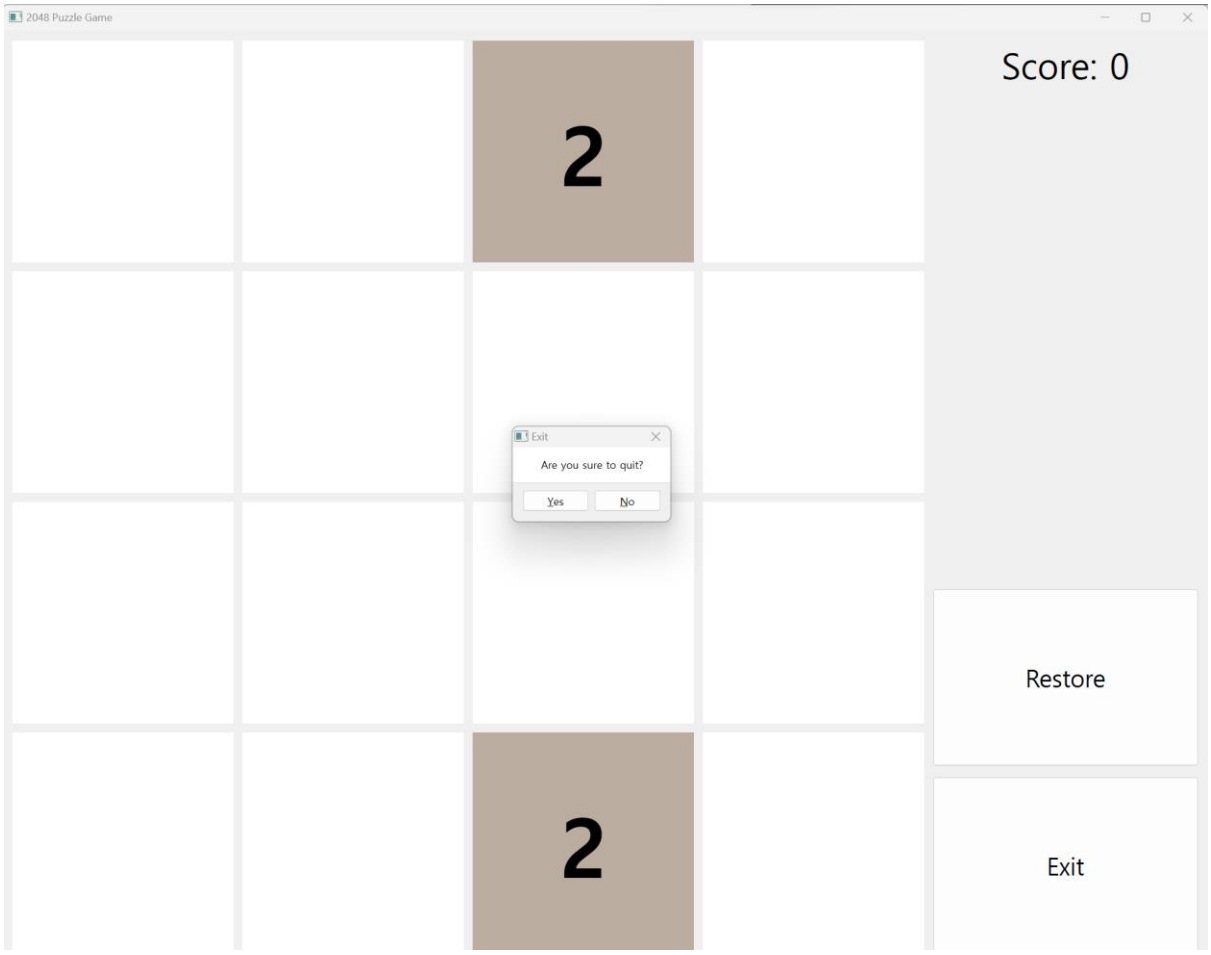
<버퍼에 기록된 보드 판이 없는 경우>


2048 Puzzle Game
Score: 0
2
Restore
There is no previously saved board in the buffer.
OK
Restore
2
Exit

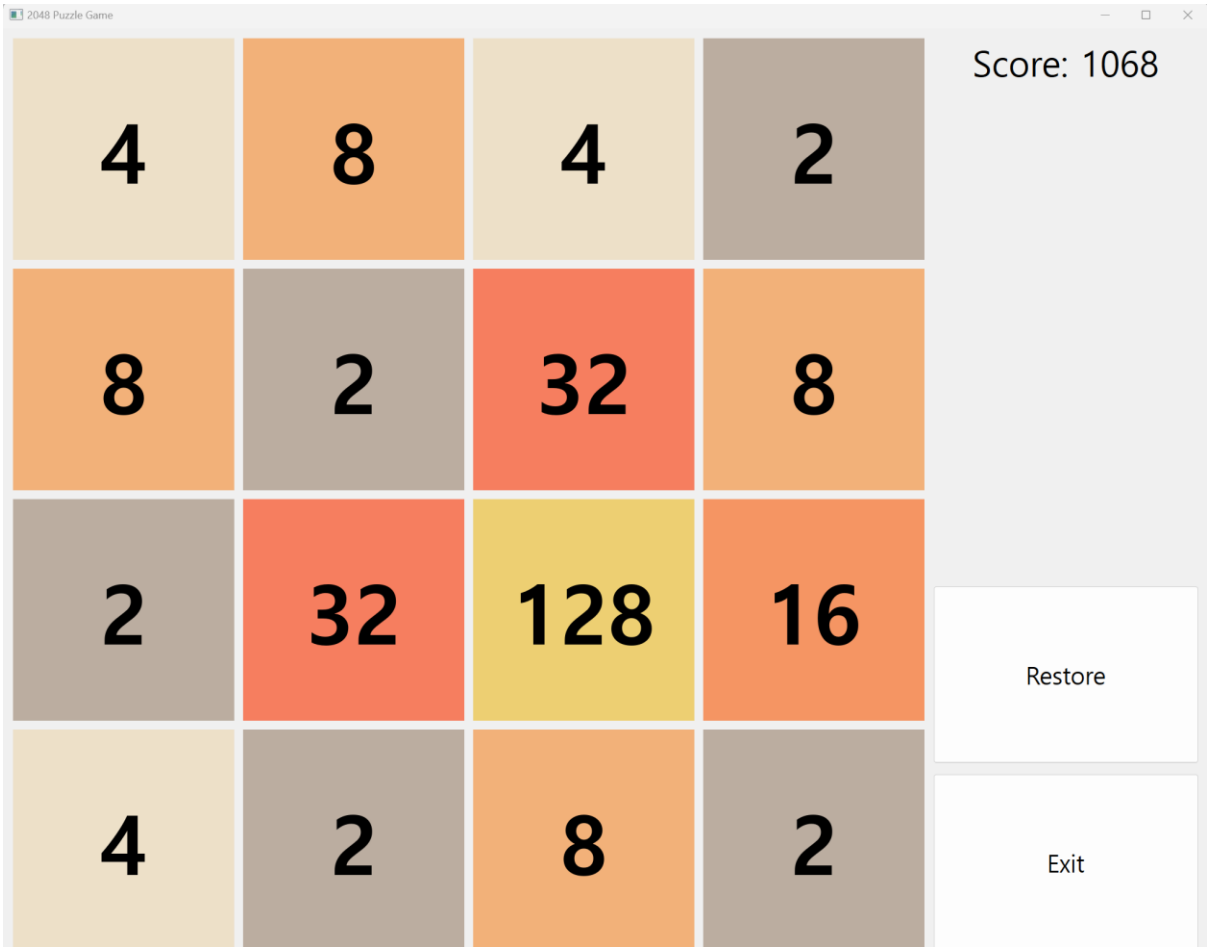
<복구 기회가 없는 경우>


2048 Puzzle Game
Score: 4
Restore
No more chance to restore the board to its previous state.
OK
2
4
Restore
Exit

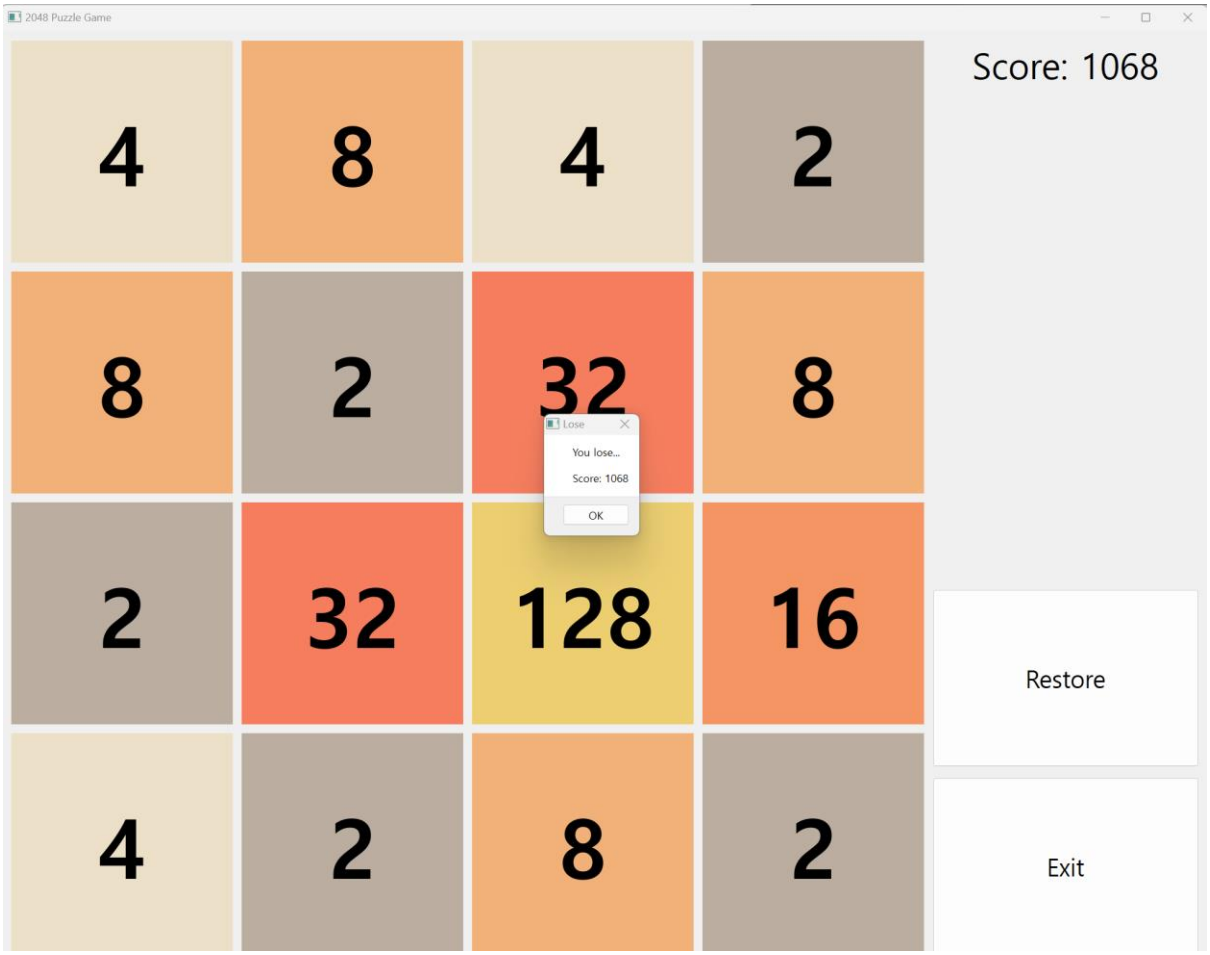
<나가기 버튼을 누른 경우>



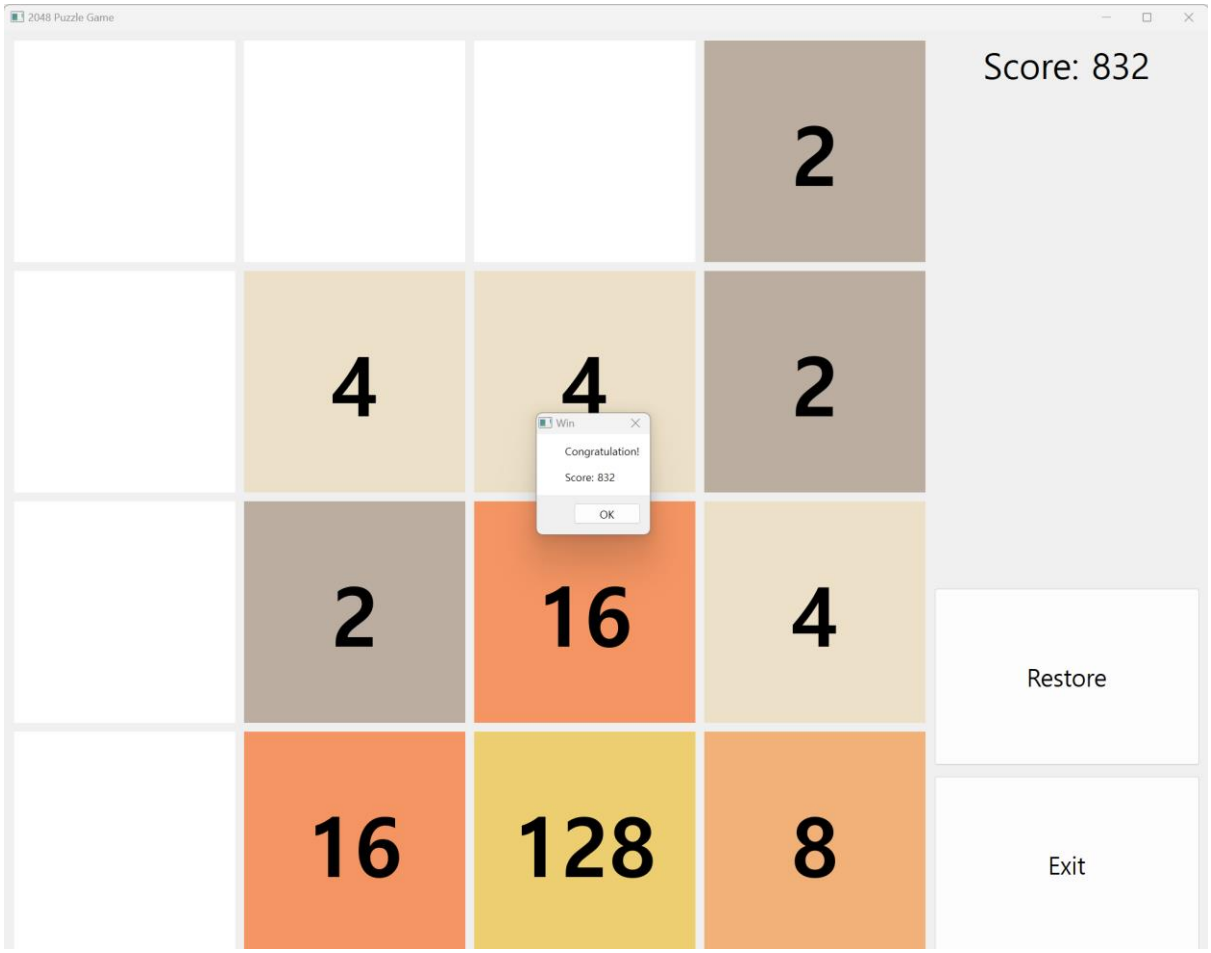
<더 이상 움직일 수 없는 경우 (방향키를 누르기 전)>



<패배한 경우 (방향키를 누른 후)>



<승리한 경우 (2048을 만들지 못한 관계로 128 생성 시 승리 조건으로 변경하여 첨부)>



3. 토론 및 개선

□ 이번 과제에서 처음으로 GUI에 대한 개념을 직접 구현해볼 수 있어서 흥미로운 과제였다. 하지만 Qt를 다루는 법을 익히는 데 많은 어려움을 겪었다. 처음에는 Qt에서 제공되는 헤더 파일이 존재하지 않는다는 오류로 인해 시작조차 하지 못해 많은 어려움을 겪었으나, 경로에 한글이 있으면 문제가 된다는 것을 알고 해결할 수 있었다. 또한, ui파일이 아닌, 코드로 GUI를 구현하는 과정에서 생소한 문법들이 많아 어떤 상황에서 어떤 명령어를 사용하고, Qt에서 제공되는 함수는 무엇이 있으며, 어떻게 했을 때 적용할 수 있는지 하나하나 시도해보면서 구현하는 과정이 시간이 매우 오래 걸리고, 어려움도 있었지만, 교수님께 수업 시간에 스스로 배우는 과정도 중요하다는 말씀 아래 나중에는 스스로 응용할 수 있는 방법을 생각하는 등 마지막 과제인 만큼 첫 번째 과제 때보다 성장한 모습을 느낄 수 있었다고 생각한다. 객체라는 개념이 정확히 이해되지 않았을 때는 private 멤버에 접근하지 못한다는 오류가 떴을 때 어떻게 해결해야 될지 모르겠어 답답함을 느끼고, 막막함도 느꼈지만, 이제는 오히려 그 객체, 클래스라는 개념 뿐만 아니라 data hiding, polymorphism, over ridding 과 같이 다양한 개념들을 종합하여 왜 이런 개념들이 중요한지 전반적으로 이해하0

고 코드를 짜는 데에도 있어 그 경계를 구분하는 법을 잘 익히고 있는 것 같아 한편으로는 큰 뿌듯함을 느낄 수 있었다.

4. 참고 문헌

- □ 조성현 교수님의 <객체지향프로그래밍> 수업 자료
- □ ASSN5 PDF에 첨부된 QT 관련 자료