Project: Othello

객체지향프로그래밍

2023202059 김나희

1. Introduction

   본 프로젝트는 고전 전략 보드게임인 오델로(Othello)를 기반으로 하여, C++과 Qt 프레임워크를 사용해 GUI 기반 애플리케이션으로 구현하는 것을 목표로 한다. 오델로는 흑돌과 백돌이 교대로 착수하며 상대방의 돌을 뒤집는 규칙을 기반으로 하며, 단순한 룰 속에 다양한 전략적 사고를 요구하는 게임이다.

   이 프로젝트에서는 단순한 텍스트 기반이 아닌, 사용자가 시각적으로 직관적으로 게임을 진행할 수 있도록 다양한 GUI 요소를 포함하였다. 메인화면, 게임 설정, 실제 게임 플레이, 게임 종료 및 리플레이 기능에 이르기까지 게임의 전체 흐름을 하나의 애플리케이션 내에서 자연스럽게 구현하는 것이 주요 목표였다.

   또한, 각 턴의 상태 저장 및 리플레이, 되돌리기 기능까지 포함하여, 객체지향 프로그래밍의 원리와 GUI 프로그래밍의 융합을 실제 프로젝트를 통해 학습하는 것이 핵심 과제이다. 본 보고서에서는 구현한 기능을 기준으로, 각 항목별로 설명 및 실제 결과 화면과 함께 자세히 기술하고자 한다.

2. 필수 구현 사항

2-1 화면 구성

본 프로젝트는 사용자의 흐름에 따라 전환되는 4가지 주요 화면으로 구성되어 있으며, 각 화면은 직관적인 UI로 설계되었다.

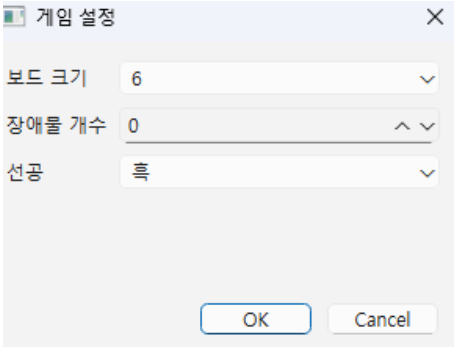
1. 메인 화면
   - 프로그램 실행 시 가장 먼저 표시되는 초기 화면이다.
   - '새 게임', '이어하기', '리플레이', '종료' 버튼이 명확하게 배치되어 있어 사용자는 자연스럽게 다음 행동을 선택할 수 있다.
   - "이어하기"와 "리플레이" 버튼은 조건을 만족할 때에만 활성화되며, 그렇지 않을 경우 비활성화 상태로 유지된다.
   - 오른쪽 하단에 본인의 학번과 이름(2023202059 김나희)이 회색 글씨로 표시된다.

<이어하기, 리플레이 비활성화>

<이어하기, 리플레이 활성화>

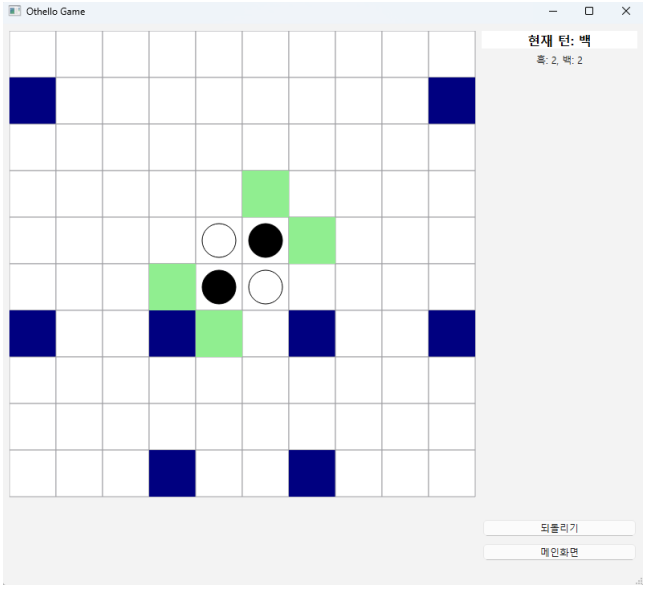
2. 게임 설정 화면
   - QDialog 기반의 모달 대화상자 형태로 구현되었으며, "보드 크기", "장애물 개수", "선공 플레이어" 세 가지 설정이 가능하다.
   - 보드 크기는 6×6, 8×8, 10×10, 12×12 중 하나를 선택할 수 있으며, 장애물 개수는 0~16까지 설정 가능하다.
   - 선공은 "흑", "백", "무작위" 중에서 선택 가능하며, 무작위 선택 시 QRandomGenerator를 사용해 동적으로 결정된다.
   - "확인" 버튼 클릭 시 게임 설정이 저장되며 게임 화면으로 전환되고, "취소" 클

릭 시 설정 없이 메인 화면으로 복귀한다.

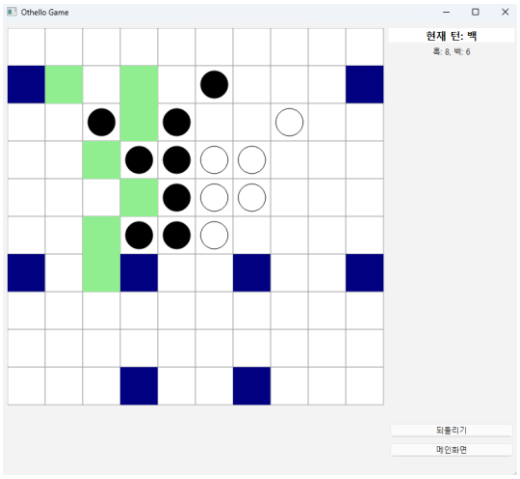


<새게임을 누르면 나오는 설정창>

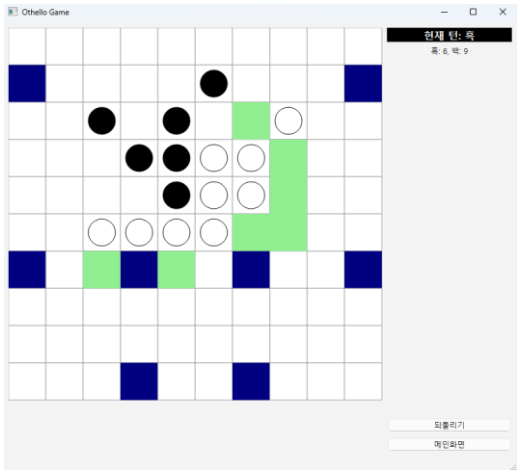
3. 게임 화면
   - 오델로 게임이 실제로 진행되는 핵심 화면이다.
   - 구성 요소:
     - 보드 UI: 설정한 보드 크기에 따라 정렬되며, 각 셀은 경계선으로 명확히 구분된다.
     - 상태 표시창: 현재 턴, 흑/백 돌 개수 표시, 배경색으로 현재 턴 강조
     - 메인화면 버튼: 게임 도중 메인화면으로 돌아갈 수 있는 버튼
     - 되돌리기 버튼: 이전 상태로 되돌아가는 기능 제공
   - 장애물은 진한 남색으로, 착수 가능한 위치는 연두색 배경으로 표시된다.
   - 마우스 클릭으로 착수 가능 위치만 반응하도록 설정되어 있으며, 착수 시 돌이 놓이고 상대 돌이 뒤집힌다.



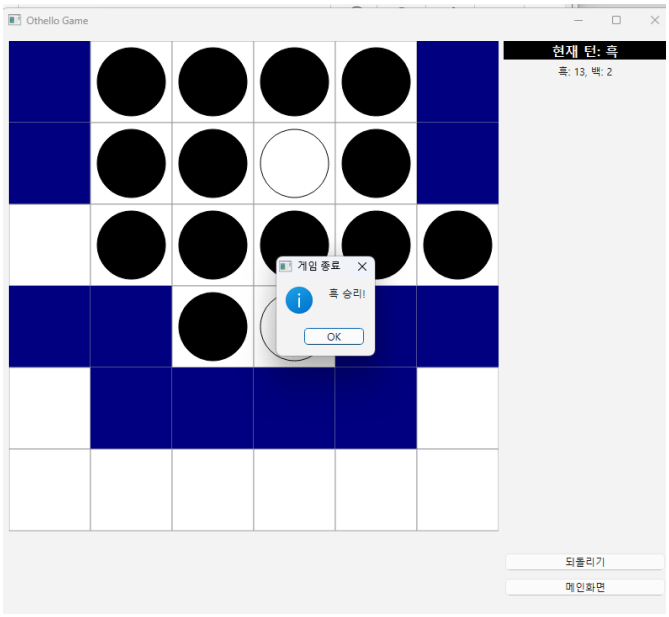
<설정 후 확인 OK 누른 후 게임 창>



<백 턴일 때, 라벨 흰색>

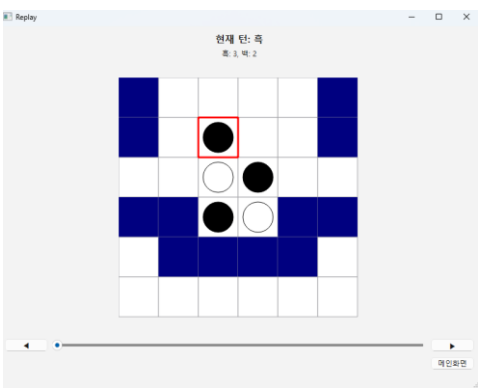


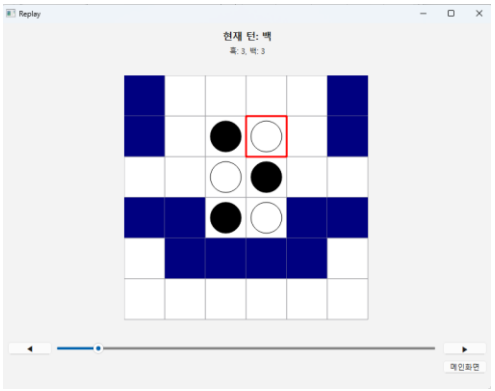
<흑 턴일 때, 라벨 검은색>



<게임 끝났을 때>

4. 리플레이 화면
   - 게임 종료 후 저장된 게임을 다시 확인할 수 있는 전용 화면이다.
   - 슬라이더 또는 이전/다음 버튼을 통해 턴을 자유롭게 이동할 수 있다.
   - 보드에는 해당 턴까지의 돌 배치 상태가 반영되며, 해당 턴에 착수한 위치는 빨간 테두리로 강조된다.
   - 착수 가능한 위치는 리플레이 중 표시되지 않으며, 단순 관람용 모드로 작동한다.
   - 하단의 '메인화면' 버튼을 통해 언제든지 메인화면으로 복귀 가능하다.



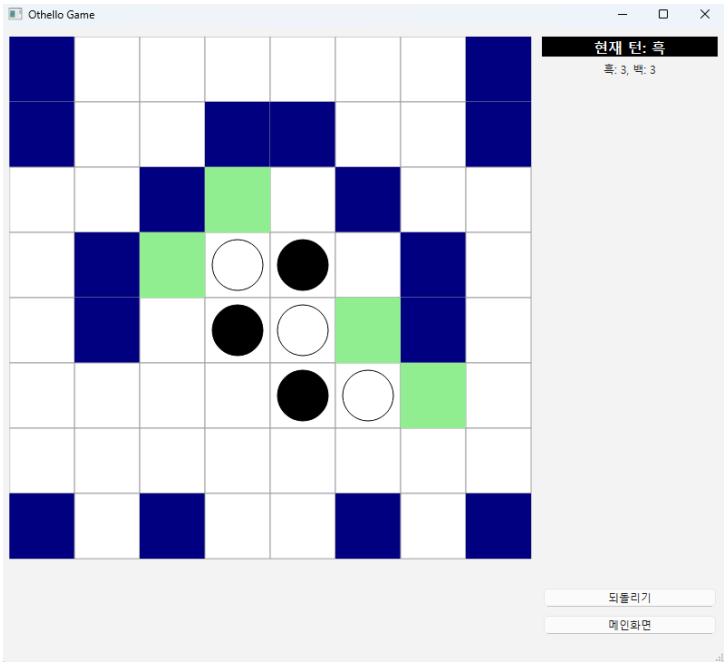


<화살표나 슬라이드를 조절하여 턴 조절>

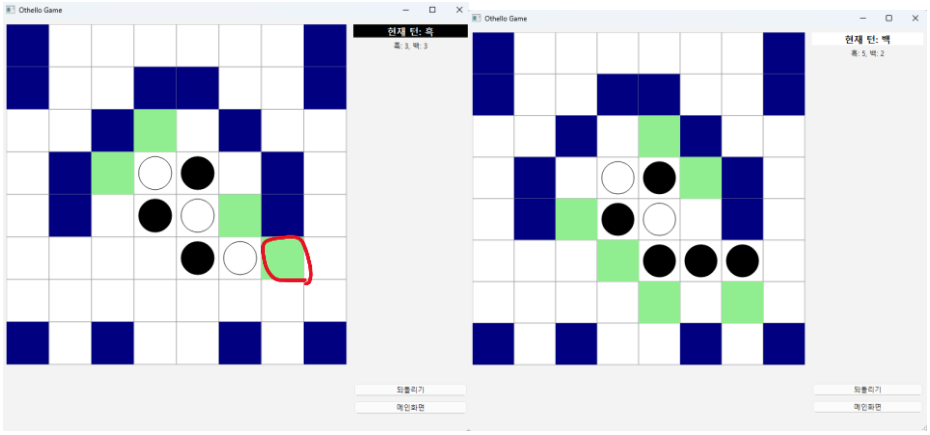
2-2 게임 규칙 구현

오델로의 게임 규칙은 간단하지만 정밀한 상태 관리가 요구된다. 본 프로젝트에서는 착수 가능 판단, 돌 뒤집기, 턴 패스 처리, 게임 종료 및 승패 판정 등 오델로의 핵심 규칙들을 정확하게 구현하였으며, GUI 인터페이스와 유기적으로 연동되도록 설계하였다.

1. 착수 가능 여부 판단
   - 각 턴마다 updateValidMoves() 함수를 통해 현재 플레이어가 둘 수 있는 모든 위치를 탐색한다.
   - isValidMove(row, col, currentPlayer) 함수는 8방향 탐색을 수행하며, 감싸는 형태가 가능한지 판단한다.
   - 탐색 기준은 다음과 같다:
     - 상대 돌이 중간에 있고
     - 그 뒤에 자신의 돌이 위치한 경우
   - 착수 가능 위치는 validMoves에 저장되며, OtheloBoard::setValidMoves()로 초록색 사각형으로 표시된다.
   - 사용자가 착수 가능한 칸이 아닌 위치를 클릭한 경우에는 무시되도록 처리되었다.

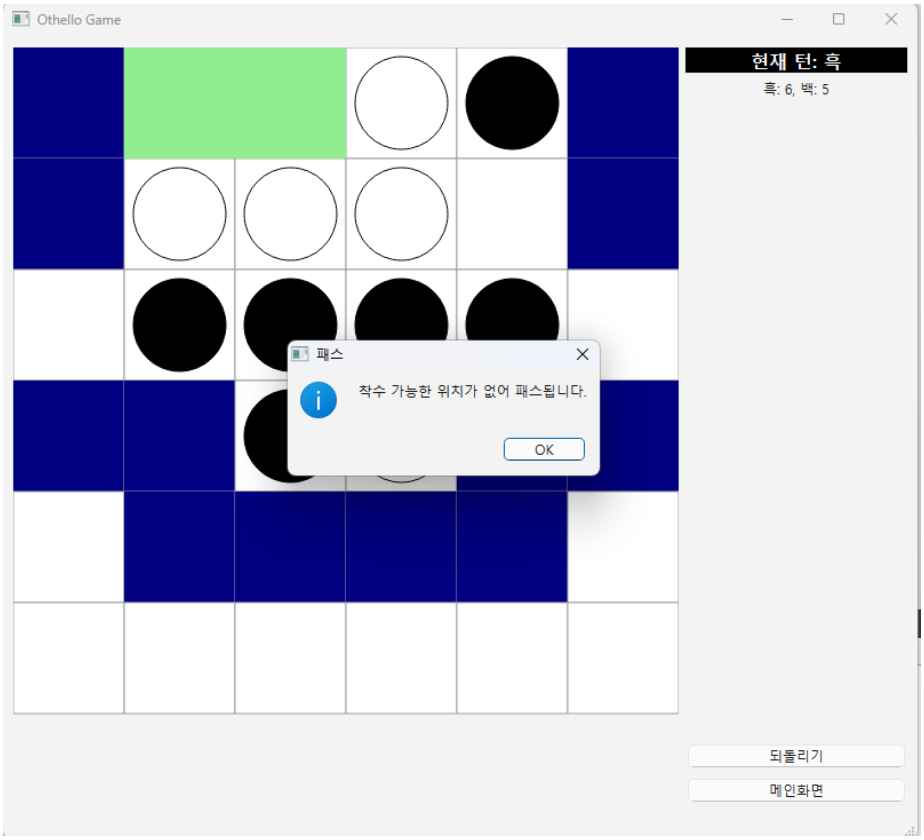


2. 상대 돌 뒤집기 처리
   - 유효한 위치에 돌을 두면 placeStone() 함수로 해당 위치에 돌을 배치하고,
   - 이어서 flipStones() 함수를 통해 8방향을 재탐색하면서 감싸진 상대 돌을 현재 플레이어의 돌로 변경한다.
   - 이 처리는 매 방향마다 중간의 상대 돌들을 리스트로 저장한 후, 현재 플레이어의 돌을 만났을 때 뒤집는 방식으로 구현되었다.



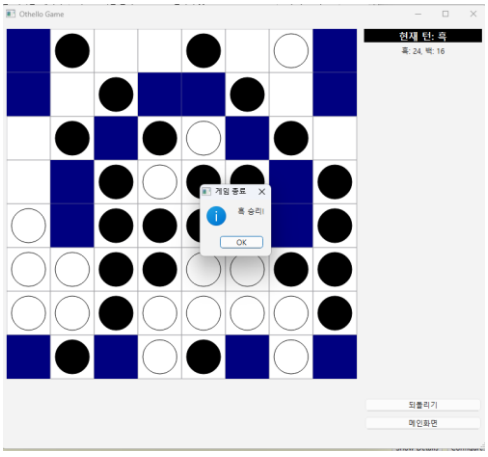
<빨간색으로 칠해진 부분에 돌을 놓았을 때>

3. 착수 불가 시 턴 패스 처리
   - 착수 가능한 위치가 없는 경우 switchTurn() 내에서 자동으로 턴이 전환된다.
   - 만약 두 플레이어 모두 착수 불가 상태가 되면, checkGameOver()가 호출되어 게임이 종료된다.
   - 패스 발생 시에는 사용자에게 QMessageBox로 알림이 표시된다.

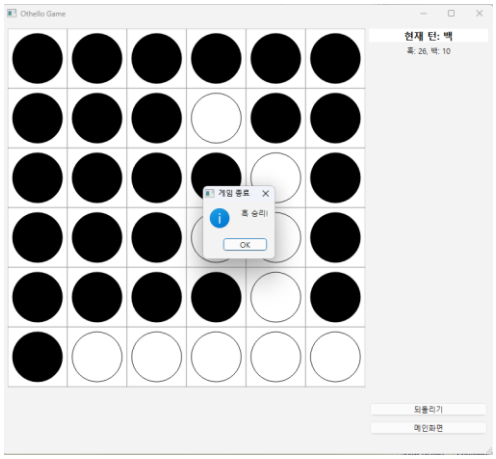


<착수 가능한 위치가 없어 턴이 패스>

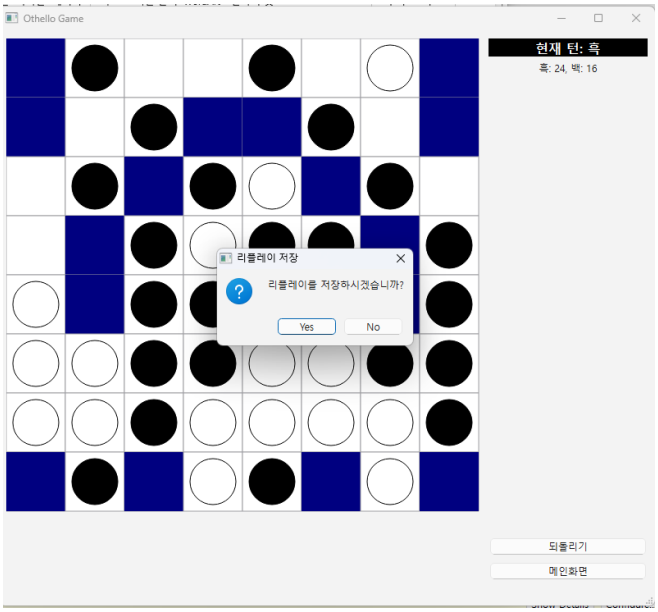
4. 게임 종료 및 승패 판정
   - 다음 두 조건 중 하나라도 만족하면 게임이 종료된다:
     1. 보드가 전부 채워졌을 경우
     2. 양쪽 플레이어 모두 착수 불가한 경우
   - checkGameOver() 함수에서 현재 보드의 흑/백 돌 개수를 카운트하여 승패 또는 무승부를 판정하고,
   - QMessageBox를 통해 최종 결과를 사용자에게 안내한다.
   - 이후, 리플레이 저장 여부를 묻고 메인화면으로 복귀한다.



<둘 다 놓을 수 없는 경우>



<가득 채워져서 끝나는 경우>

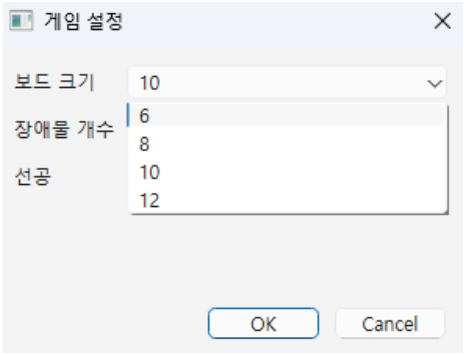


<리플레이 여부 묻기>

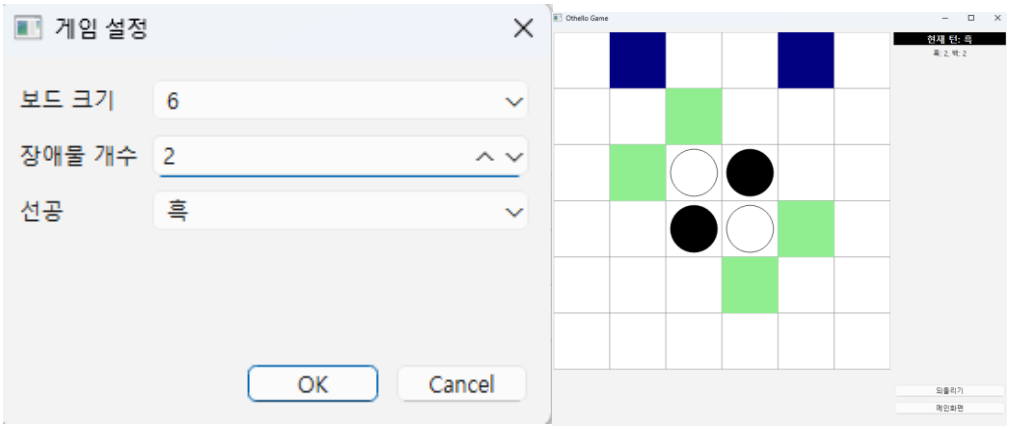
2.3 게임 설정 기능

게임 시작 전 사용자로부터 원하는 환경 설정을 받아, 이에 따라 게임 보드를 구성할 수 있도록 SettingsDialog 클래스를 구현하였다. 이 설정 창은 QDialog 기반 모달 창으로 구성되며, 아래 세 가지 항목을 입력받는다:

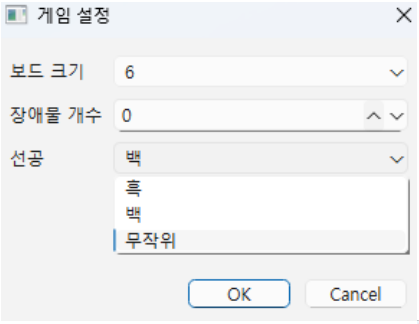
1. 보드 크기 설정
   - 사용자에게 6×6, 8×8, 10×10, 12×12의 짝수 크기 중 하나를 선택할 수 있도록 QComboBox로 구성되어 있다.
   - 선택된 보드 크기는 getBoardSize() 함수를 통해 Gamewindow로 전달되며, initializeBoardUI()에서 해당 크기만큼 boardState를 2차원 벡터로 생성한다.
   - 보드의 그래픽 표현도 해당 크기에 맞게 조절되며, 칸 수에 따라 UI가 자동 확장되도록 설정되었다.

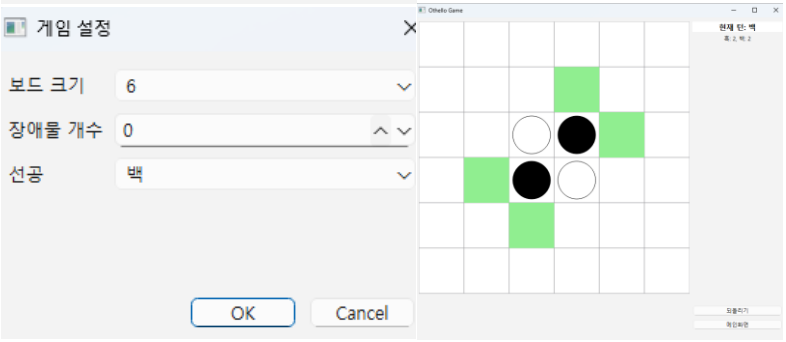


2. 장애물 개수 및 배치
   - QSpinBox를 사용하여 장애물 개수를 0~16 사이로 선택 가능하도록 하였고, 짝수 개수 단위로 좌우 대칭 배치를 구현하였다.
   - 실제 배치는 Gamewindow::placeObstacles() 함수에서 이루어지며, 중앙의 초기 돌 4개 위치는 제외하고, 좌우 대칭이 되도록 무작위 좌표를 반복 탐색하여 배치한다.
   - 이미 돌이나 장애물이 있는 자리는 피해서 배치되며, 최대 1000회까지 시도하여 무한 루프를 방지한다.
   - 장애물은 진한 **남색(navy)**으로 칠해져 시각적으로 구분 가능하다.



3. 선공 플레이어 설정
   - 사용자는 선공 플레이어로 "흑", "백", "무작위" 중에서 선택 가능하며, QComboBox로 구성되었다.
   - 설정 결과는 getFirstPlayer()를 통해 전달되며, 무작위 선택 시 QRandomGenerator를 사용하여 흑 또는 백을 랜덤하게 결정한다.
   - 게임 화면에서는 이 값이 firstPlayer 변수로 설정되어 첫 번째 턴의 주체가 결정된다.
   - 현재 턴은 상태창 상단(turnLabel)에 실시간으로 표시되며, 텍스트와 배경 스타일로 시각적으로 강조된다.





<선공 백으로 설정>

<백 턴>

4. 기타 UI 동작
   - "확인" 버튼 클릭 시 설정이 저장되며 Gamewindow가 실행되고,
   - "취소" 버튼 클릭 시 설정 내용은 반영되지 않고 메인 화면으로 복귀한다.
   - 모든 입력값은 설정 후 Gamewindow 생성자에 인자로 전달되어 반영된다.

2.4 게임 상태 표시

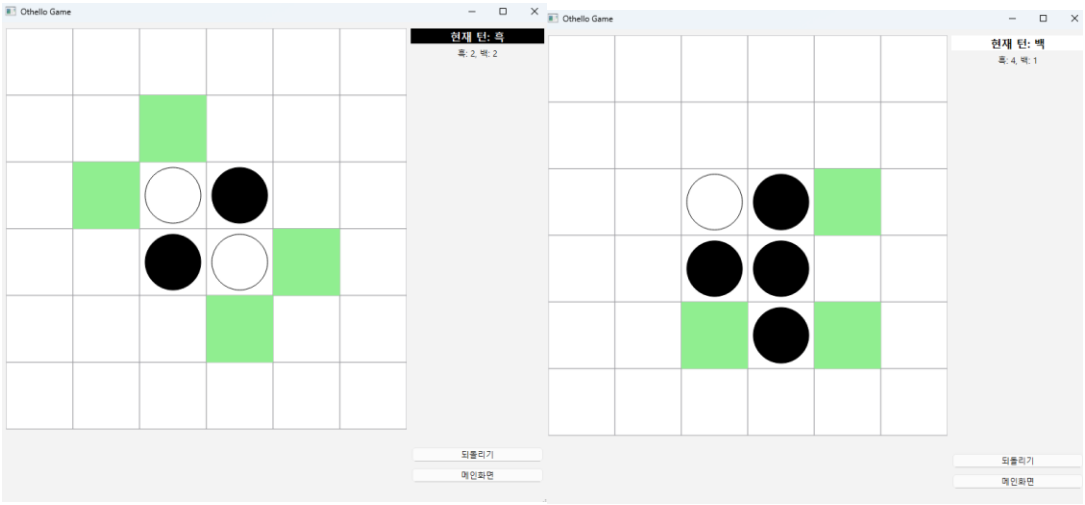
게임의 각 턴마다 사용자에게 실시간으로 게임 상태를 시각적으로 제공하기 위해 상태 표시창을 구현하였다. 이 상태창은 게임 화면 상단에 배치되며, 현재 턴 정보, 돌 개수, 착수 가능 위치 등을 실시간으로 표시한다. 다음과 같은 방식으로 구현되었다:

1. 현재 플레이어의 차례 표시
   - Gamewindow 클래스의 firstPlayer 변수를 기반으로 현재 차례를 판단한다.
   - updateStatus() 함수에서 상태 표시창 위젯 turnLabel의 텍스트를 갱신하며, "현재 턴: 흑" 또는 "현재 턴: 백" 식으로 출력된다.
   - 배경색과 폰트 스타일은 굵게 지정되어 있으며, 사용자 시선에 잘 들어오도록 중앙 정렬 처리하였다.

2. 각 플레이어의 돌 개수 표시

- 매 턴마다 현재 보드 상태를 순회하며 흑돌과 백돌의 개수를 계산한다.
- updateStatus() 함수에서 돌 개수를 카운트한 뒤, countLabel 위젯에 "흑: 10, 백: 8"과 같은 형식으로 출력한다.
- 돌을 착수하거나 뒤집을 때마다 자동으로 갱신되며, 게임 종료 시에도 최종 결과 판정에 사용된다.

3. 착수 가능한 위치 표시
   - 각 턴마다 updateValidMoves() 함수가 호출되어 현재 플레이어가 착수 가능한 좌표들을 탐색한다.
   - 이 위치 정보는 OtheloBoard::setValidMoves()로 전달되며, 보드 위에 연두색 사각형으로 시각적으로 표시된다.
   - 유효하지 않은 위치(= 표시되지 않은 곳)는 클릭해도 아무 동작도 발생하지 않도록 처리되었다.
   - 표시된 착수 가능 위치는 턴이 전환될 때마다 자동으로 갱신된다.



2.5 리플레이 기능

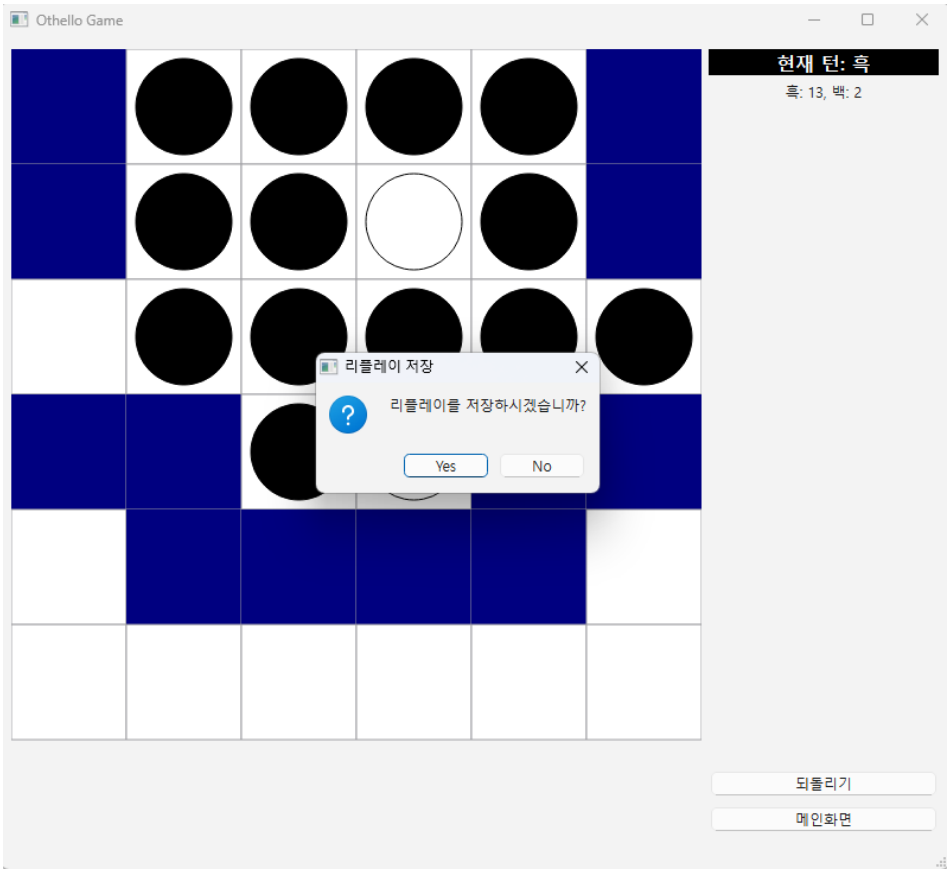
본 프로젝트에서는 게임이 종료된 후, 사용자 요청에 따라 해당 게임을 턴 단위로 재생할 수 있는 리플레이 기능을 구현하였다. 이를 위해 게임 도중 각 턴의 상태를 저장하고, 별도의 리플레이 전용 화면에서 이를 재구성하여 보여준다.

---

1. 턴별 보드 상태 저장
   - 게임 중 착수가 이루어질 때마다 ReplayState 구조체를 통해 현재 보드 상태와 착수 정보를 저장한다.
   - 턴의 저장은 Gamewindow::placeStone() 함수에서 수행되며,
     - 놓인 위치(row, col)
     - 돌의 색상(color)

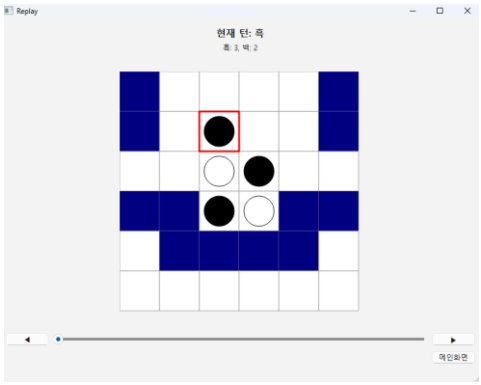
- 보드 상태(board)

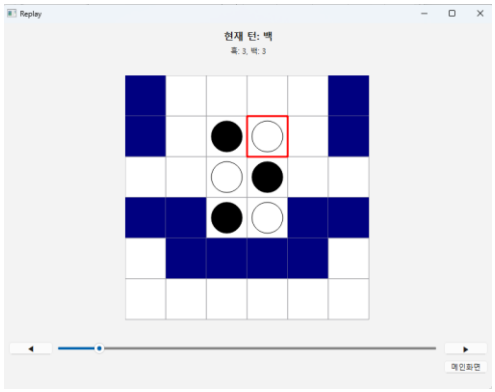
- 정보는 replayStates 벡터에 순차적으로 추가된다.
- 종료 시 gameEnded 시그널로 메인화면에 전달되며, 저장 여부는 사용자에게 팝업으로 묻는다.



2. 면 구성 및 턴 이동 기능
   - 리플레이 화면은 별도의 ReplayWindow 클래스로 구현되며, 다음 요소로 구성된다:
     - 보드 UI (OtheloBoard)
     - 현재 턴 정보 라벨
     - 흑/백 돌 개수 표시 라벨
     - 슬라이더 및 이전/다음 버튼
     - 메인화면 복귀 버튼
   - 사용자는 슬라이더를 조작하거나 ◀/▶ 버튼을 클릭하여 원하는 턴으로 자유롭게 이동할 수 있다.
   - ReplayWindow::showTurn(int index) 함수를 통해 해당 턴의 보드 상태와 착수 위치를 반영한다.
   - 슬라이더는 0부터 마지막 턴까지 범위를 가지며, 조작 시 보드 상태가 즉시 반영된다.

3. 착수 위치 강조 및 착수 가능 위치 제거
   - 리플레이에서는 게임 중처럼 초록색 착수 가능 위치는 표시되지 않으며, 관람 모드로 작동한다.
   - 대신, 해당 턴에 착수한 위치만 빨간색 테두리로 강조되어 흐름을 파악할 수 있게 했다.
   - 이전 턴의 강조는 자동으로 제거되고, 현재 턴만 강조된다.
   - 강조는 OtheloBoard::setHighlight(row, col)로 구현되었으며, 빨간색 테두리로 시각화된다.





4. 기타 리플레이 동작 조건
   - 리플레이는 게임이 종료된 후 사용자 선택으로만 접근 가능하며, 메인 화면의 "리플레이" 버튼이 활성화된다.
   - 게임이 저장되지 않은 경우에는 리플레이 기능이 비활성화된다.

2.6 메인화면 이어하기 기능

게임 도중 사용자가 '메인화면' 버튼을 눌러 게임을 종료하지 않고 나간 경우, 해당 시점의 게임 상태를 저장해두었다가 이후 다시 이어서 플레이할 수 있도록 '이어하기' 기능을 구현하였다. 이를 통해 사용자는 실수로 게임을 종료하더라도, 중단된 시점부터 그대로 이어서 플레이할 수 있다.

1. 저장 조건 및 동작 흐름
   - 게임 화면에서 사용자가 메인화면으로 돌아갈 경우, 게임이 아직 종료되지 않았다면 현재 게임 인스턴스를 MainWindow로 전달하여 저장한다.
   - 이 저장은 requestSave(Gamewindow*) 시그널을 통해 이뤄지며, savedGame 포인터로 유지된다.
   - 저장 시 continueButton을 활성화하여 사용자에게 이어하기 가능 상태임을 시각적으로 알린다.

2. 이어하기 버튼 동작
   - 메인화면에서 "이어하기" 버튼을 클릭하면, 저장된 Gamewindow 인스턴스를 다시 show()하여 중단된 상태부터 게임을 이어간다.
   - 이어하기는 항상 가장 최근에 중단된 게임 한 개만 저장되며, 게임이 종료된 경우에는 저장되지 않는다.
   - 저장된 인스턴스가 없을 경우, 이어하기 버튼은 비활성화 상태를 유지하거나 클릭해도 반응하지 않도록 설정되어 있다.
3. 예외 처리 및 제한사항
   - 사용자가 메인화면으로 이동한 후 새로운 게임을 시작하고 해당 게임을 정상 종료하더라도, 이전에 저장된 미완료 게임은 여전히 이어할 수 있다.
   - 리플레이 기능과는 독립적으로 작동하며, 종료된 게임은 리플레이로만 확인할 수 있다.

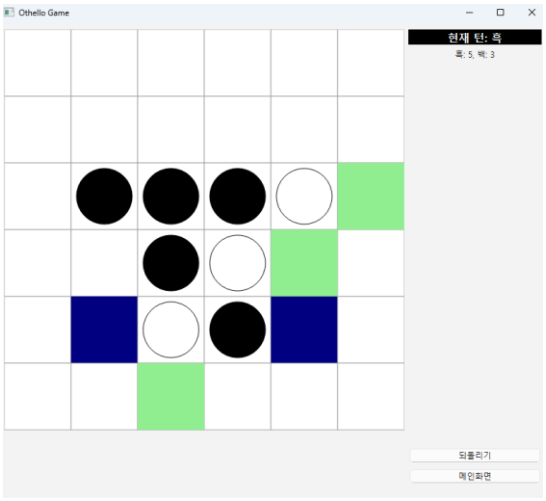
3. 선택 구현 사항

3.1 되돌리기 기능 (Undo)

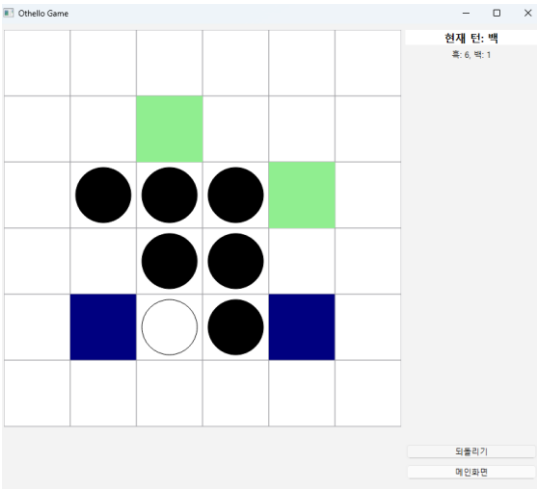
게임 도중 사용자가 실수로 돌을 잘못 놓았을 경우, 이전 턴으로 되돌아갈 수 있도록 Undo 기능을 구현하였다. 이 기능은 오직 게임 중에만 사용할 수 있으며, 보드 상태, 턴 정보, UI 요소들이 이전 상태로 정확하게 복원된다.

1. 동작 개요
   - 게임 중 한 턴을 진행한 뒤, '되돌리기' 버튼을 클릭하면 해당 턴 이전 상태로 복구된다.
   - 돌 배치 상태, 현재 플레이어, 착수 가능 위치, 상태 표시창(턴/돌 개수) 등 모

든 정보가 함께 되돌아간다.



<되돌리기 전>



<되돌린 후>

2. 구현 방식

- Gamewindow 클래스에서 undoStack이라는 QStack<GameState> 구조를 사용해 이전 상태를 저장한다.
- 각 턴 착수 직전에 saveCurrentState()를 호출하여 현재 상태를 스택에 저장한다.
- restorePreviousState() 함수가 호출되면 스택에서 가장 최근의 상태를 꺼내 복원하고, UI도 동기화한다.
- ReplayState도 한 턴 제거되어 리플레이 기록과의 일관성도 유지된다.

3. UI 연동 및 반영

- 되돌리기 버튼은 게임 화면 하단에 배치되어 있으며, 클릭 시 즉시 동작한다.
- 되돌리기 후에는:
    - 보드 위 돌 배치(setBoard)
    - 착수 가능 위치(setValidMoves)
    - 현재 턴 정보 및 돌 개수(updateStatus)
    - 리플레이 기록(replayStates.removeLast)

      가 모두 갱신된다.

4. 제한 조건

- 되돌리기 기능은 게임이 진행 중일 때만 가능하며,
- undoStack이 비어있는 경우(= 되돌릴 상태가 없는 경우)에는 아무 동작도 하지 않는다.
- 리플레이 모드에서는 되돌리기 버튼이 비활성화되며, 관람 전용으로 작동한다.

4. 결론

본 프로젝트는 오델로 게임의 규칙을 기반으로, Qt 기반 GUI 프로그램 설계 및 객체지향적 구조 구현을 목표로 하여 수행되었다. 메인화면부터 게임 설정, 실제 플레이, 리플레이 및 되돌리기 기능까지 포함된 완성형 오델로 애플리케이션을 개발하면서, 사용자 인터페이스 구성, 이벤트 처리, 상태 저장 및 복원과 같은 다양한 프로그래밍 기술을 실습할 수 있었다.

구현 과정에서 특히 신경 쓴 부분은 다음과 같다: 착수 가능 여부 판단 및 돌 뒤집기 로직의 정확성, UI와 로직의 실시간 동기화 (돌 개수, 현재 턴 등), 되돌리기 및 리플레이 기능을 통한 게임 흐름 추적과 복원

이번 프로젝트를 통해 단순한 게임 로직을 넘어서, GUI 환경에서의 복잡한 상태 관리, 사용자 중심 인터페이스 설계, 클래스 간 역할 분리 및 연동 구조에 대한 실전 경험을 얻을 수 있었다.

결과적으로, 게임의 규칙적 완성도뿐 아니라, 사용자의 편의성과 직관적인 조작을 함께 고려한 설계를 이루어낸 점에서 객체지향 프로그래밍 학습의 실제 적용 예시로서 매우 유익한 프로젝트였다고 평가할 수 있다.