**Tic Tac Toe**

1. 프로젝트 목표, 실험 환경에 대한 설명

이번 프로젝트 목표는 Tic-Tac-Toe라는 게임을 openframework를 이용하여 구현하는 것이다. Tic-Tac-Toe라는 게임은 두 명의 이용자가 참여하는 보드게임으로 두 명의 플레이어가 ‘O’ , ‘X’ 기호를 번갈아가며 보드 판 위에 두는 게임이다. 이때 승리하는 조건은 보드판의 크기가 5\*5라면 한 플레이어가 해당 플레이어의 기호를 가로,세로 또는 대각선으로 5개 연속으로 채우면 이기게 된다.

구체적으로 5\*5 Tic-Tac-Toe게임을 구현할려고 하며 게임 방식은 두명의 플레이어가 번갈아 가며 기호를 두고자 하는 보드 칸에 마우스를 클릭하면 해당 기호가 화면에 나타나는 방식으로 구현할려고 한다. 게임결과는 콘솔창에 출력하도록 하였다.

프로젝트 환경은 openframework을 다운받아 그 파일 내에 있는 projectGenerator.exe.를 실행해 파일을 생성하였다. tictactoe라는 파일을 생성하였고 그 파일 내 src파일 안에 (main.cpp, ofApp.cpp, ofApp.h)에 코드를 구현하였다.

2. 각 변수에 대한 설명

<ofApp.h 파일>

1> char int SIZE : 틱택토 보드판의 크기

2> const char EMPTY: 처음에 보드판을 초기화시킬 기호

3> const char PLAYER1: 플레이어1의 기호

4> const char PLAYER2: 플레이어2의 기호

5> vector<vector<char>> board : 각 플레이어들의 기호(진행상황)을 담을 이중벡터

6> char currentPlayer : 현재 어떤 플레이어 차례인지 나타내는 변수

7>bool gameOver : gameOver인지 아닌지 체크하는 flag변수

8>float cellSize : 각 보드판의 칸 크기를 나타내기 위한 변수 (가로 사이즈), ofGetWidth를 이용하여 가로사이즈를 정하였다.

9>float cellSize\_y : 각 보드판의 칸 크기를 나타내기 위한 변수 (세로 사이즈), ofGetHeiht를 이용하여 세로 사이즈를 정하였다.

3. 각 함수에 대한 설명

<ofApp.h 파일>

1> drawBoard():

보드판을 그려주는 함수. 가로세로 선을 그려주고 board배열을 확인하여 각 플레이어들이 둔 기호(’X’ 또는 ‘O’)들을 그림으로 그려준다. 이 떄 cellsize 와 cellsize\_y 변수를 고려하여 해당 크기에 맞게 보드판을 출력하도록 한다.

2> isMoveValid(int row, int col) :

보드판에서 마우스클릭을 받을때 마우스클릭이 보드판 범위 내에 있는지 혹은 보드판이 비어있는지 체크해주는 함수이다.

3> isBoardFull() :

보드판이 플레이어들의 기호로 다 채워져있는지 확인하는 함수이다.

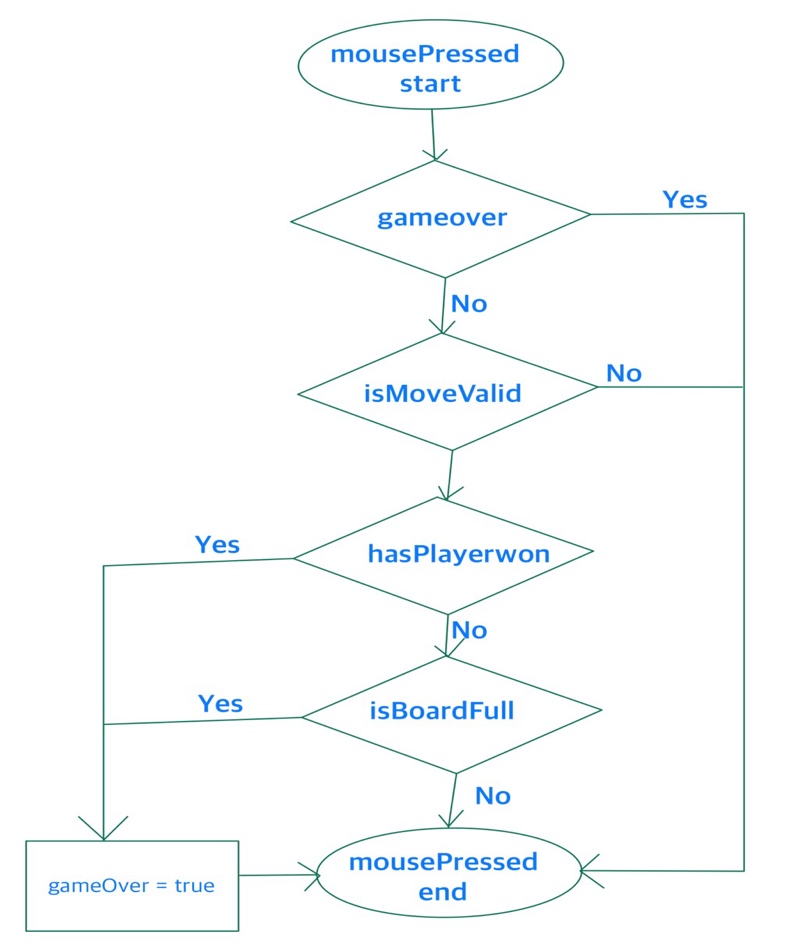
4> hasPlayerWon(char player) :

게임을 이긴 플레이어가 있는지 체크해주는 함수이다. 가로줄, 세로줄, 대각선줄로 채워져있는지 보드판을 체크한다.

4.

1) 플로우차트

이번 프로젝트의 플로우차트는 아래와 같다. 사용자들의 마우스 클릭을 입력으로 받은 후 다음과 같은 방식으로 게임이 진행된다.



mousePressed 함수에서는 사용자의 마우스클릭을 받는 함수이다. 이때 입력을 받고 gameover인 경우에는 바로 mousePressed함수를 끝낸다. 하지만 gameover가 아니고 게임이 진행중일 떄는 isMoveValid함수를 호출해 사용자의 마우스클릭한 좌표가 유효한지 체크해준다. 유효하지 않은 경우에는 바로 mousePressed함수를 종료한다. 그다음에 hasPlayerwon 함수를 호출해 방금 받은 마우스클릭 후 해당 사용자가 승리할 조건을 만족하는지 체크해준다. 그래서 hasPlayerwon이 만족된다면 해당 플레이어가 이겼다는 메시지를 콘솔에 호출해주고 gameOver = true로 설정한후 mousePressed함수를 종료한다. 하지만 hasPlayerwon이 만족이 안될경우 즉 누구도 승리하지 않았을때는 isBoardFull 함수를 호출하여 보드판이 다 찼는지 확인한다. 보드판이 다 찬 경우에는 게임이 비겼다는 것을 의미하며 그랬을 경우 마찬가지로 게임이 비겼다는 결과를 콘솔창에 출력해주고 gameOver를 true로 할당하며 mouserPressed를 끝내준다. 그리고 나머지의 경우, 즉 이긴사람이 존재하지 않고 보드판이 다 차지도 않았을 때는 아무 변화없이 mousePressed함수를 종료한다.

2) 자료구조 및 알고리즘

1> 이중벡터

board를 표현하기 위해 실습 때 사용하지 않았던 새로운 자료구조인 이중벡터를 사용하였다. 벡터 자료구조는 동적배열을 나타내는 자료구조로 일반 배열과 달리 배열의 크기를 자동으로 조정해준다. 이번 프로젝트에서는 보드판의 크기를 초기화할 때 resize()함수를 사용하여 SIZE(여기서는 5)만큼 벡터의 크기를 조정하였고 각 행들도 SIZE만큼 벡터를 구성해 이중벡터로 board를 나타내었다. 이렇게 하면 기존에 메모리할당을 위해 쓰던 malloc과 같은 함수와 메모리해제를 위한 free같은 함수없이 배열을 구현할 수 있다.

2> 알고리즘

이번 프로젝트에서 쓰인 주요 알고리즘은 브루트포스 알고리즘이다. 브루트포스 알고리즘이란 모든 경우의 수를 확인하는 알고리즘으로 완전탐색이라고도 한다. 플레이어가 tic-tac-toe 게임을 이겼는지 확인하는 hasPlayerwon함수에서 쓰였다. 이 게임의 승리조건은 가로,세로 혹은 대각선으로 보드판을 같은 기호로 채우는 것이었는데 이때 모든 가로줄,세로줄, 혹은 대각선줄로 같은 기호로 채워져있는지 확인할때 모든 경우를 칸마다 이동하며 확인하였다.

3) 시간/공간복잡도

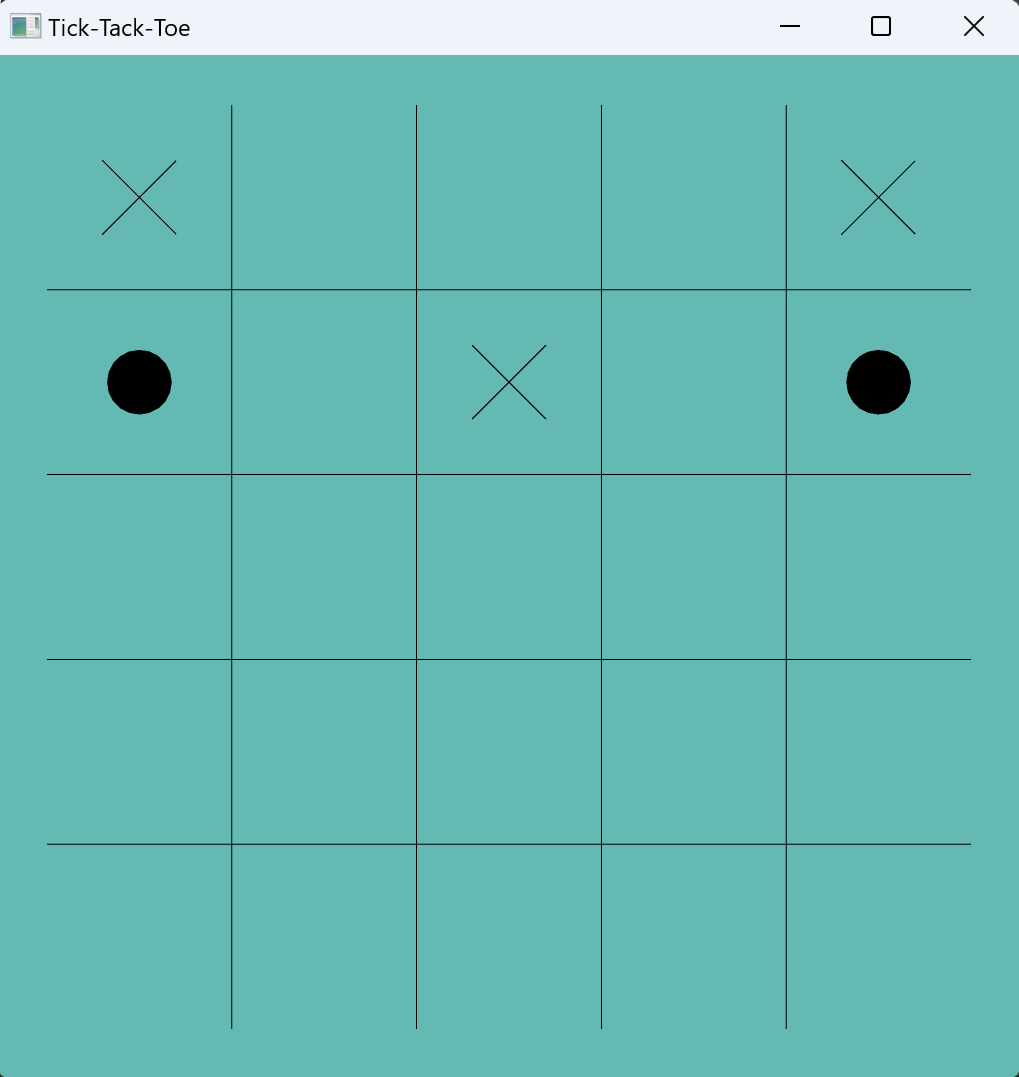
공간복잡도 : 여기서 board배열을 사용할 때 쓰인 공간복잡도는 배열판의 크기를 n이라고 하면 가로\*세로이므로 O(n^2)이다.

시간복잡도: hasPlayerwon함수는 기호가 5개가 채워져있는지 가로줄을 확인한다고 가정하면 첫번째 가로줄부터 마지막 가로줄까지 체크하고 가로줄의 개수가 n이라고 하면 시간복잡도는 O(n)이다. 그리고 세로줄도 마찬가지 방법이므로 전체적인 시간복잡도는 O(n)이라고 할 수 있다.

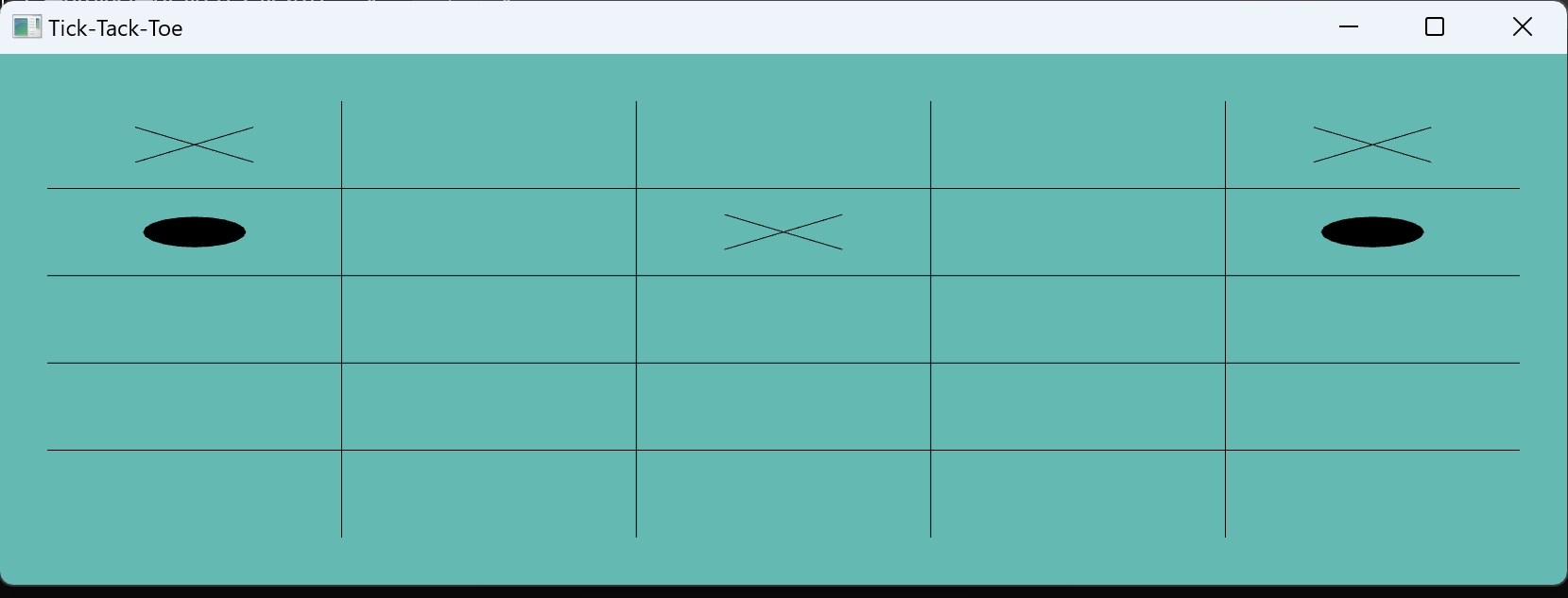
5.

이번 게임을 구현하면서 창의적으로 구현할려 했던 점은 창의 크기에 맞게 보드판의 크기와 모양을 재구성하는 것이었다. 보통 기본 창의 크기에 맞게 보드판이 출력이 되는데 이 때 창의 크기를 바꿨을때 아무런 기능을 구현하지 않은 상태라면 보드판의 모양이 잘리는 현상이 생길 것이다. 그래서 창의 크기를 조절하더라도 게임을 즐길 수 있게 보드판의 모양을 그 크기에 맞게 다시 보이도록 코드를 구현하였다. 구현방법은 ofGetWidth()와 ofGetWidth()를 사용하였다. 각각의 함수는 창의 너비와 높이를 반환하는 함수이다. 따라서 이 두 함수를 이용하여 각 칸의 크기를 다시 정의하여 그려주도록 하였다. 칸 안에 있는 기호도 마찬가지 방법으로 작동하게 하였다.

1) 초기 상태의 창



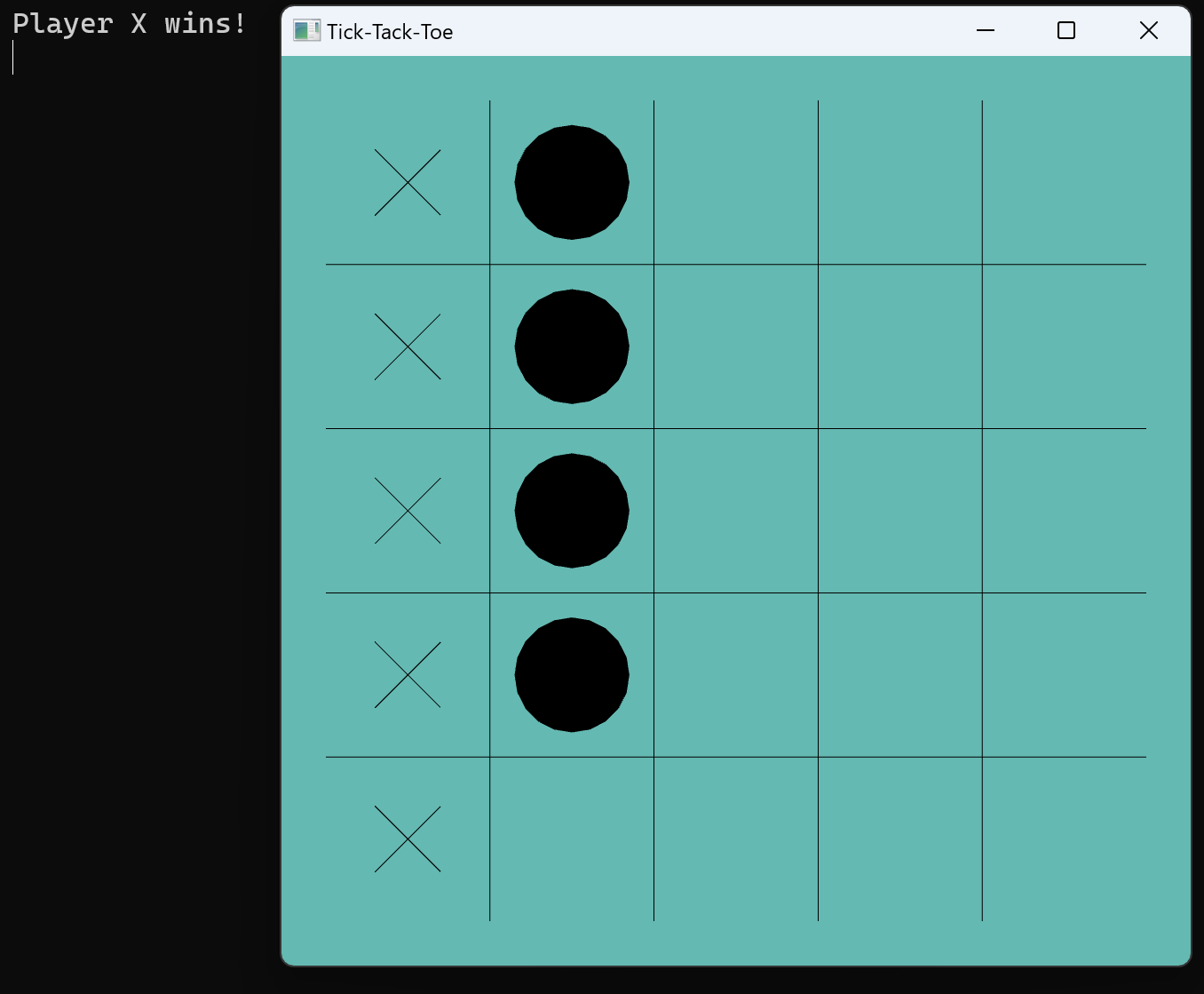
2) 크기를 바꾼 후의 창



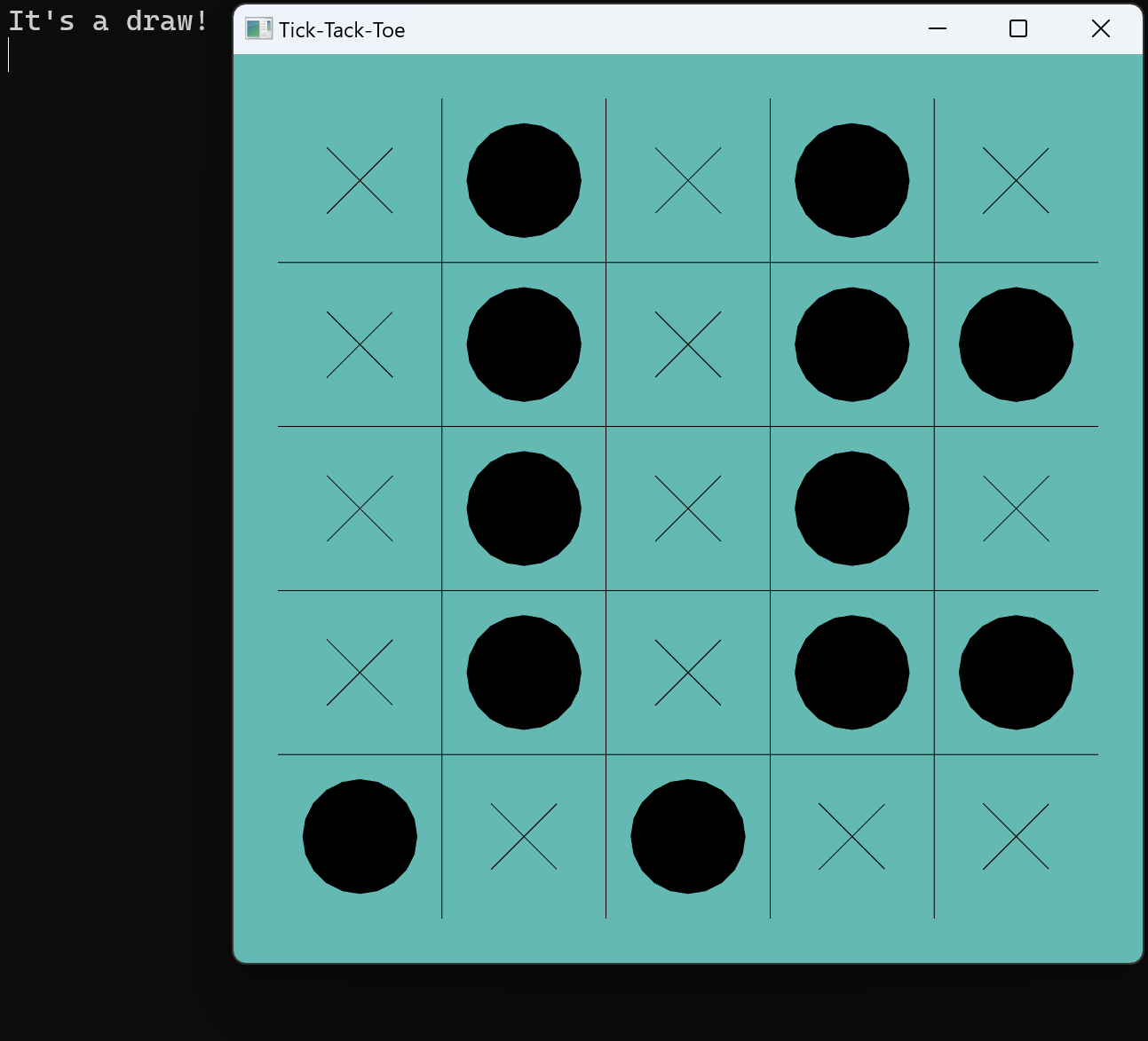
6.

다음은 실제 게임을 실행했을 경우 각 상황에 맞는 화면들을 캡쳐한것이다.

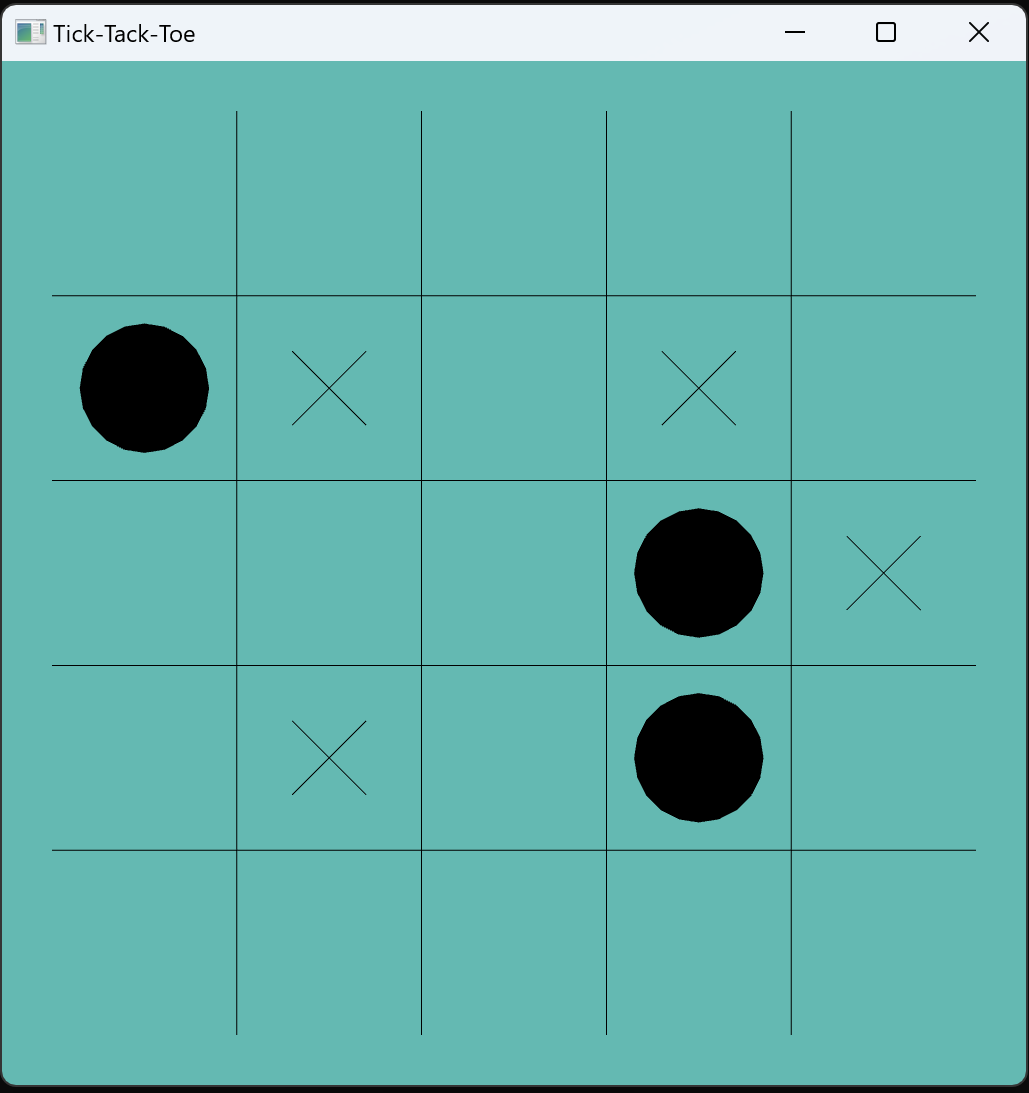
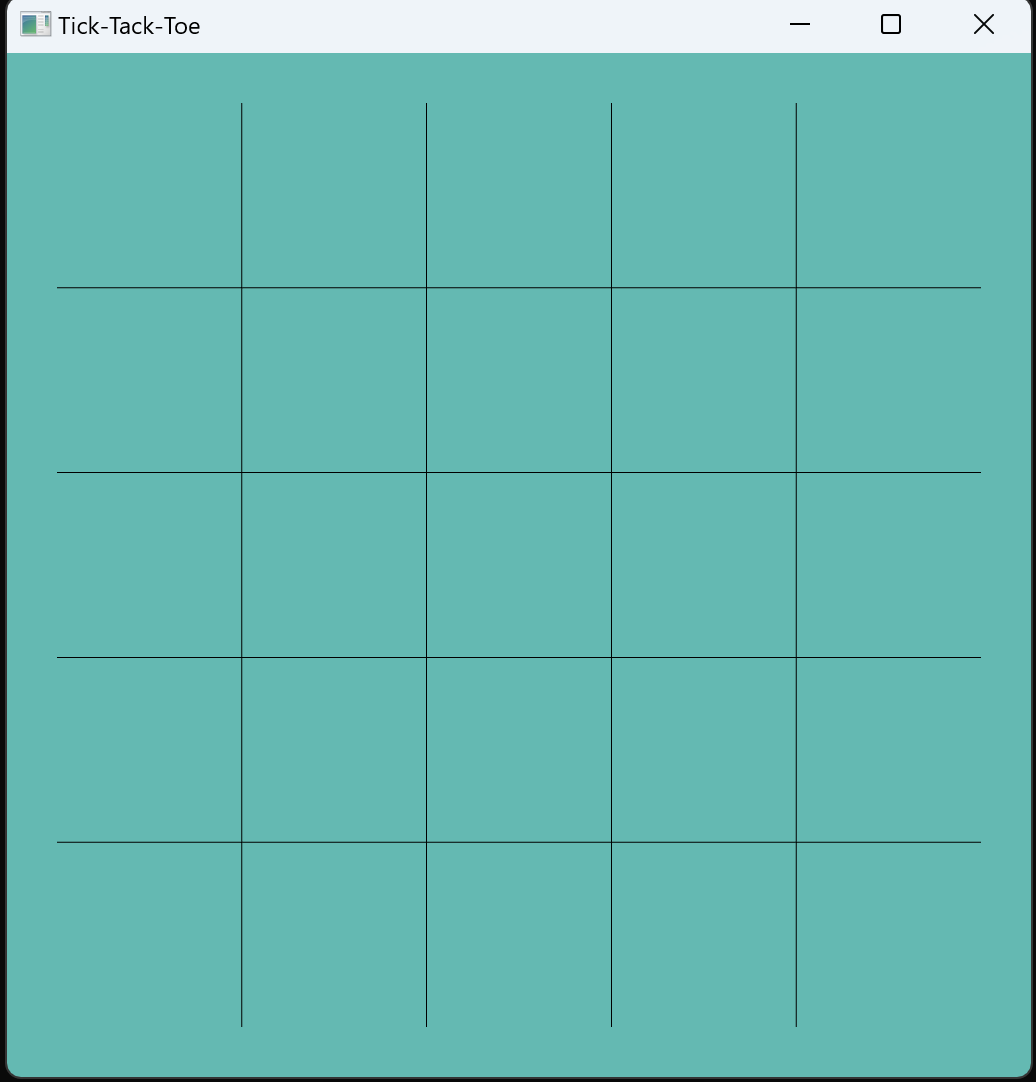
1) player x가 이기는 경우



2) 비기는 경우

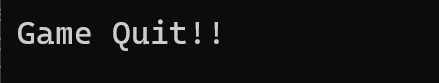


3) 게임을 리셋하는 경우(’r’ 버튼)

4) 게임을 끝내는 경우(’q’버튼)

보드판 창이 꺼지면 아래와 같은 메시지가 콘솔 창에 뜨게된다.



7.

이번 프로젝트를 진행하면서 벡터라는 자료구조와 mousePressed함수, 브루트포스 알고리즘을 활용했고 이에 대해 숙지할 수 있는 좋은 기회였다.

벡터는 동적배열을 자동으로 지원해주는 자료구조로 프로젝트를 진행하면서 메모리 할당 및 해제에 사용자가 신경쓸 필요 없이 본연의 작업에 집중할 수 있어서 편리했던 것 같다. 그리고 벡터 자료구조에 제공되는 여러 편의 기능들이 있어 벡터라는 자료구조 자체를 스택, 큐처럼 사용할 수 있다. 이런 자료구조는 여러모로 사용자에게 도움을 주는 것 같다.

mousePressed 함수는 openframework에서 제공하는 함수이며 마우스클릭을 할때 호출된다. 이때 마우스의 위치(x,y 좌표)와 마우스의 상태를 인자로 받고 이를 이용하여 tic-tac-toe에서 마우스 클릭한 부분에 기호를 둘 수 있게 구현하였다.

브루트포스 알고리즘은 가능한 모든 경우를 직접 확인하는 알고리즘이다. 여기서는 tic-tac-toe 게임에서 승리자가 나왔는지 체크하는 용도로 알고리즘을 활용하였다. 가장 간단하고 직관적인 알고리즘이지만 경우에 따라서 시간복잡도가 필요이상으로 커질수도 있고 비효율적인 방법일수 도 있다. 그래서 이런 알고리즘이 적합한지 생각해봐야되며 이번 프로젝트 같은 경우에는 올바르게 쓰인 것 같다.