By: Phillip Hunter

Part 1: Tic-Tac-Toe

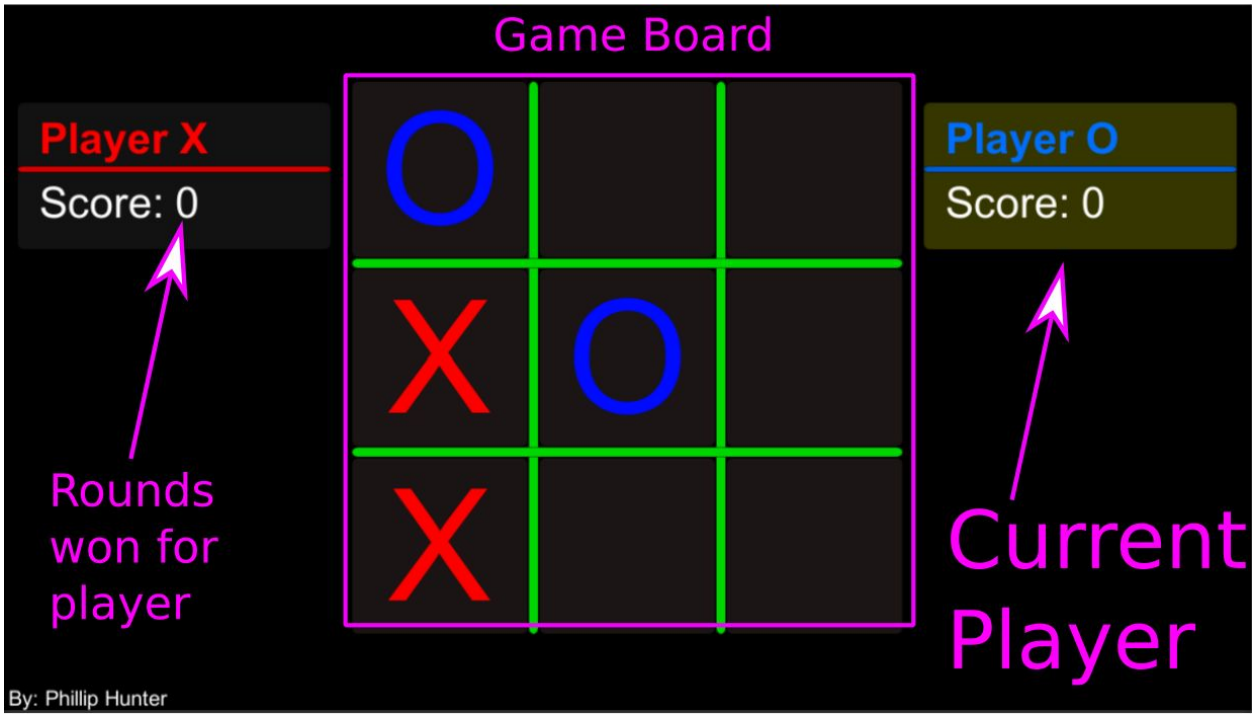
Game Type: Two Player Puzzle Game
How to Play:

- Decide which letter will be assigned to which player.
- The current player will be indicated by a yellowish background behind their score.
- Each player will take turns placing their letter on the board.
- The game will notify the players when the game is over, whether by a player win or draw.
- Players then have the option to restart the game. The game will keep track of the wins of each player.

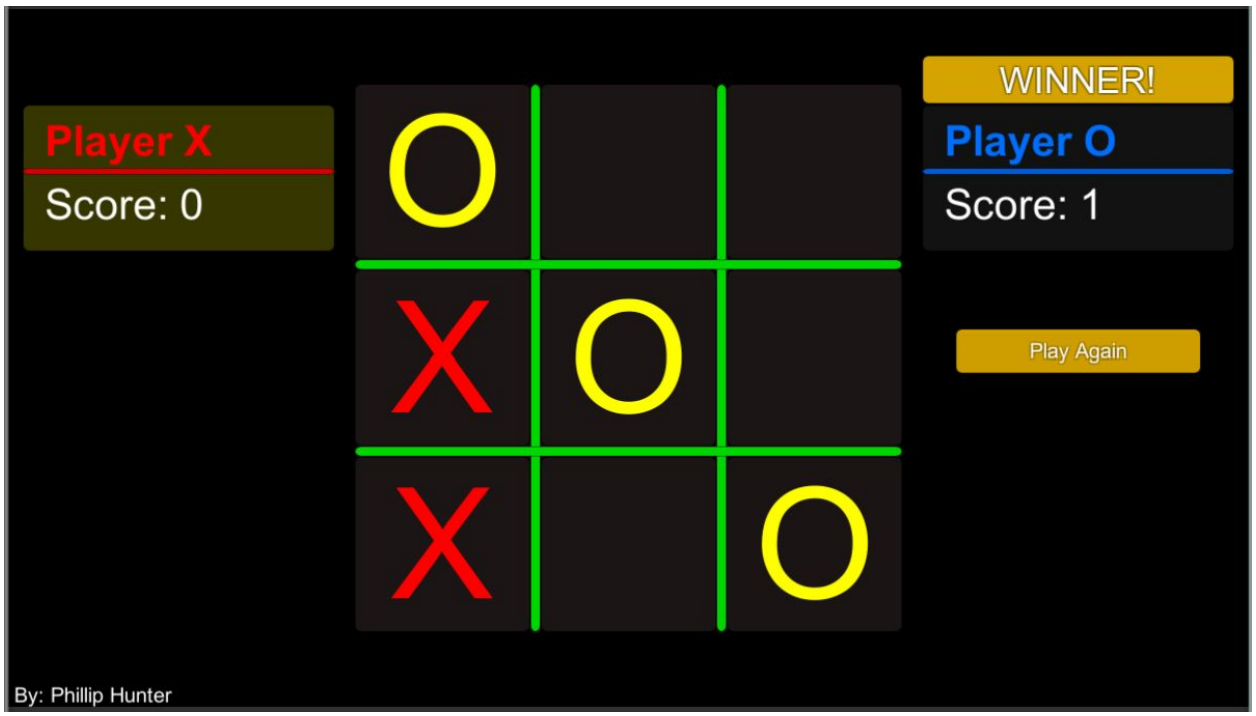
Behind the Scenes:

- The game uses a simple statically created canvas to draw the entire game.
    - Canvas consists of UI elements for game buttons, score indicators, and other graphical functions.
- References to canvas are obtained through serialized fields through the Unity inspector to a game controller object.
- The board status is saved as a character array. This array is updated when buttons are clicked, and is in the logic for determining win conditions.
    - Win condition logic comes down to algorithmically checking rows and columns for a lack of each player's letter. If their letter is not found in either a row, column, or diagonal, they couldn't logically have won this turn.
        - Note: It is possible to apply this win condition logic to a dynamically generated board, consisting of more than 3 rows and columns.
- The UI, as well as the score for the proper player is updated based on winner if a win condition is present.

Diagram of UI elements



Example game won by player O



By: Phillip Hunter

Part 2: Minesweeper

Game Type: Single Player Puzzle Game
How to Play:

- Player clicks a block to clear it. If it is not a mine, it will clear all spaces around it that do not border mines. If it is a mine, the player loses.
- If a cleared space borders at least one mine, it will display a number of how many mines border it.
- The player wins if they uncover every space that is not a mine.

Behind the Scenes:

- Base algorithm for the game logic was provided for the assignment.
- Several changes were made to this algorithm to allow for
    - Dynamic generation of the game.
        - This is implemented by saving Elements as prefabs, and instantiating them at runtime.
        - The X and Y position of each element is now saved as a field of the element class.
        - Usage: The world size can be changed for different difficulties.
    - The first block selected is never a bomb, to prevent the player from automatically losing on the first play.
        - This is implemented by generating bomb locations immediately after the first click, not at launch.
- A timer was added that counts up, in seconds, to display the amount of time the player has taken to complete the level.
    - This timer uses a simple InvokeRepeating call.
- If the game is won or loss, a congratulatory message is shown, telling the player of the outcome.
- An options menu is present, which allows the player to select a difficulty from a choice of three presets, or manually drag the board size and mine frequency sliders to their liking.
    - These menus stop the time from incrementing, as the majority of the game board is hidden and the time should not be increased at this time.
    - These menus and notifications are implemented using separate canvases that are enabled or disabled as needed.

****** Fun fact: The background sky of my minesweeper game changes its time of day depending on the real life time of day. As the time of day moves forward, the directional light changes angle proportionally to this time, allowing the procedural skybox to change the apparent time. This is implemented using simple arithmetic and the computers set time.***

By: Phillip Hunter

Typical layout of a lost game

Options Menu