Activity 5 Part 2

Ian M. McConihay

College of Science, Engineering and Technology, Grand Canyon University

CST-150: C# Programming I

Mark Smithers

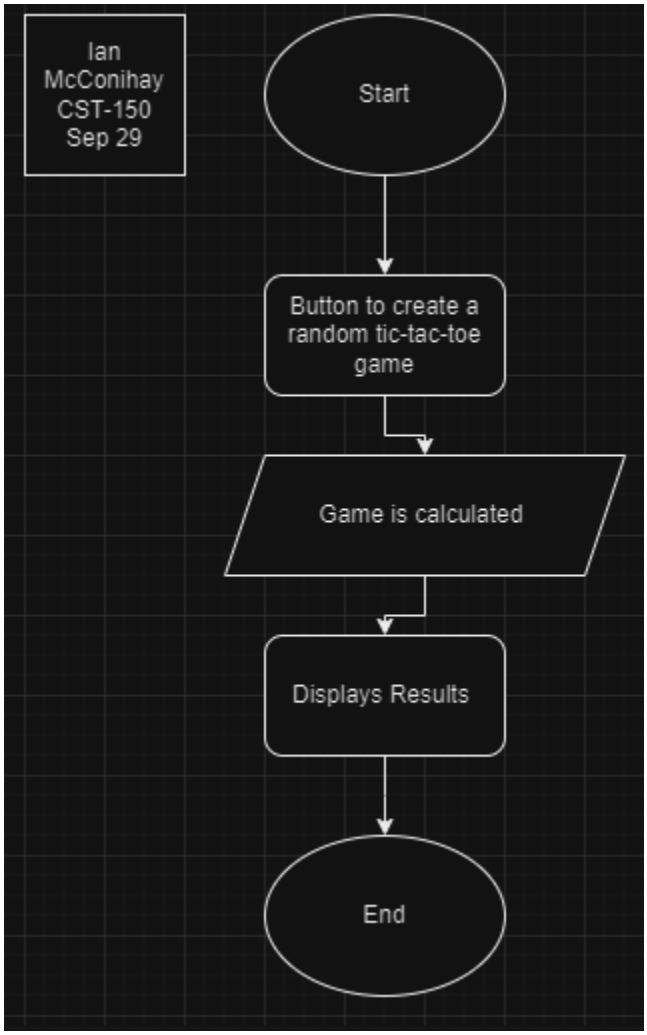
September 29, 2024

Video Link:

https://www.loom.com/share/94275610135a453badfb413eeb25475c?sid=44af4e89-311c-42fe-869f-f43092d32017

Github: https://github.com/Ian-McConihay/CST-150

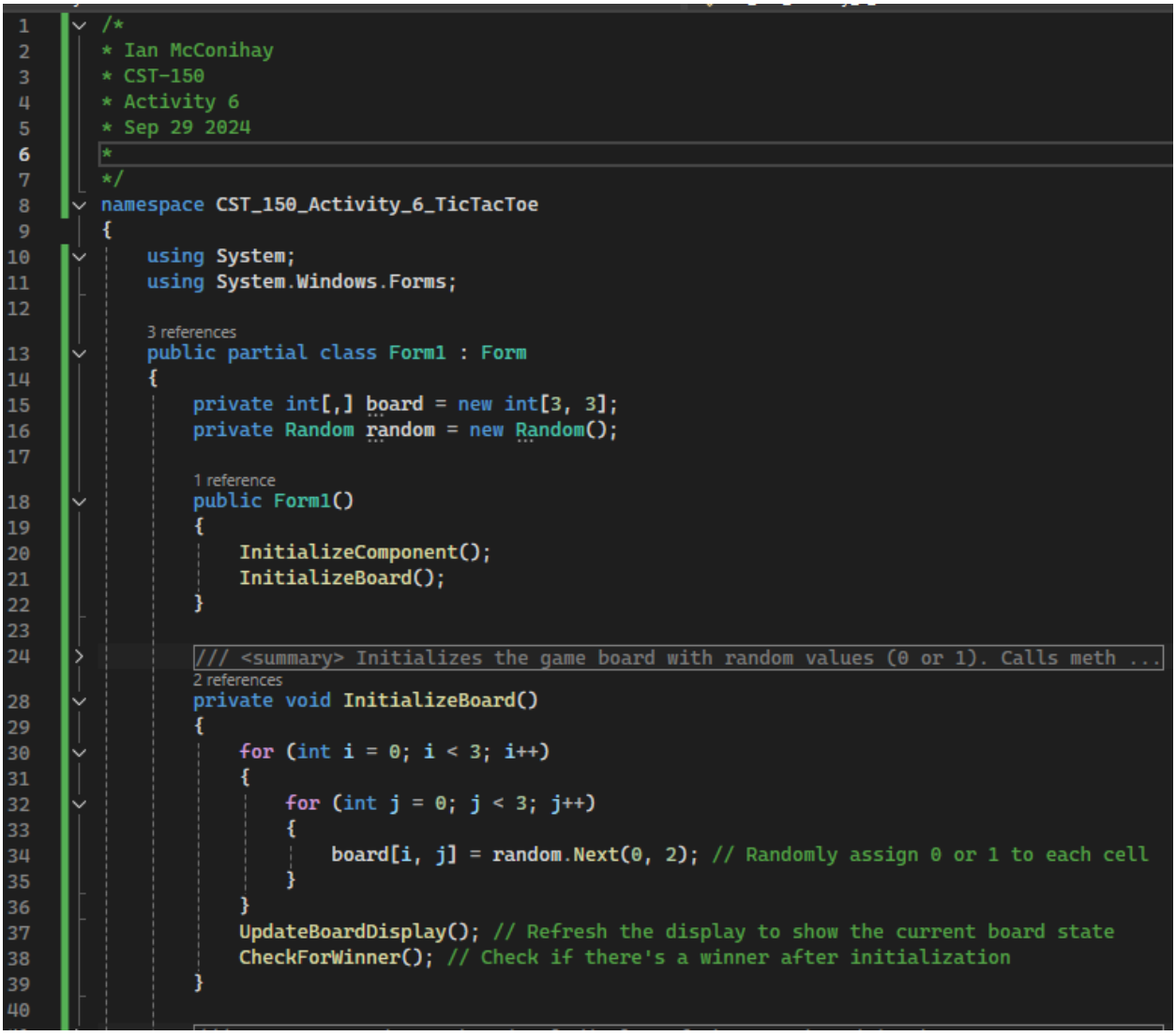
Flowchart



The flow chart for this activity describes a tic tac tow generator. The Idea is to push a button to have a new game be played and the results displayed. This is not an interactive game but more to see the results.

Application Screenshots

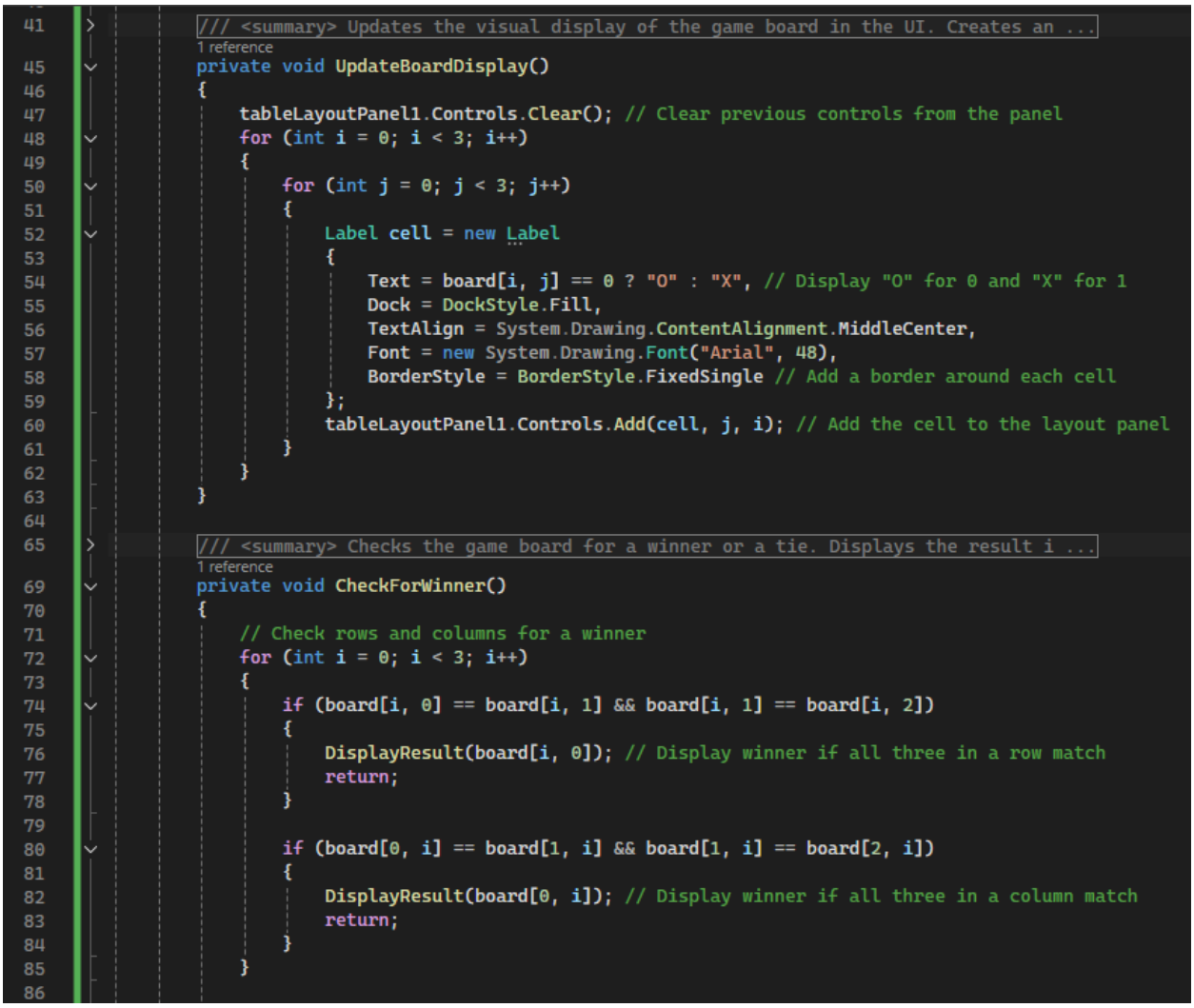
Figure 1: Code



```
/*
* Ian McConihay
* CST-150
* Activity 6
* Sep 29 2024
*
*/
namespace CST_150_Activity_6_TicTacToe
{
    using System;
    using System.Windows.Forms;

    3 references
    public partial class Form1 : Form
    {
        private int[,] board = new int[3, 3];
        private Random random = new Random();

        1 reference
        public Form1()
        {
            InitializeComponent();
            InitializeBoard();
        }

        /// <summary> Initializes the game board with random values (0 or 1). Calls meth ...
        2 references
        private void InitializeBoard()
        {
            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    board[i, j] = random.Next(0, 2); // Randomly assign 0 or 1 to each cell
                }
            }
            UpdateBoardDisplay(); // Refresh the display to show the current board state
            CheckForWinner(); // Check if there's a winner after initialization
        }
```

In this screenshot we can see the citation. After that we Initialize the board. This initializes the game board with random values (0 or 1). Calls methods to update the display and check for a winner.

Figure 2: Code



```
41      /// <summary> Updates the visual display of the game board in the UI. Creates an ...
        1 reference
45      private void UpdateBoardDisplay()
46      {
47          tableLayoutPanel1.Controls.Clear(); // Clear previous controls from the panel
48          for (int i = 0; i < 3; i++)
49          {
50              for (int j = 0; j < 3; j++)
51              {
52                  Label cell = new Label
53                  {
54                      Text = board[i, j] == 0 ? "O" : "X", // Display "O" for 0 and "X" for 1
55                      Dock = DockStyle.Fill,
56                      TextAlign = System.Drawing.ContentAlignment.MiddleCenter,
57                      Font = new System.Drawing.Font("Arial", 48),
58                      BorderStyle = BorderStyle.FixedSingle // Add a border around each cell
59                  };
60                  tableLayoutPanel1.Controls.Add(cell, j, i); // Add the cell to the layout panel
61              }
62          }
63      }
64
65      /// <summary> Checks the game board for a winner or a tie. Displays the result i ...
        1 reference
69      private void CheckForWinner()
70      {
71          // Check rows and columns for a winner
72          for (int i = 0; i < 3; i++)
73          {
74              if (board[i, 0] == board[i, 1] && board[i, 1] == board[i, 2])
75              {
76                  DisplayResult(board[i, 0]); // Display winner if all three in a row match
77                  return;
78              }
79
80              if (board[0, i] == board[1, i] && board[1, i] == board[2, i])
81              {
82                  DisplayResult(board[0, i]); // Display winner if all three in a column match
83                  return;
84              }
85          }
86
```

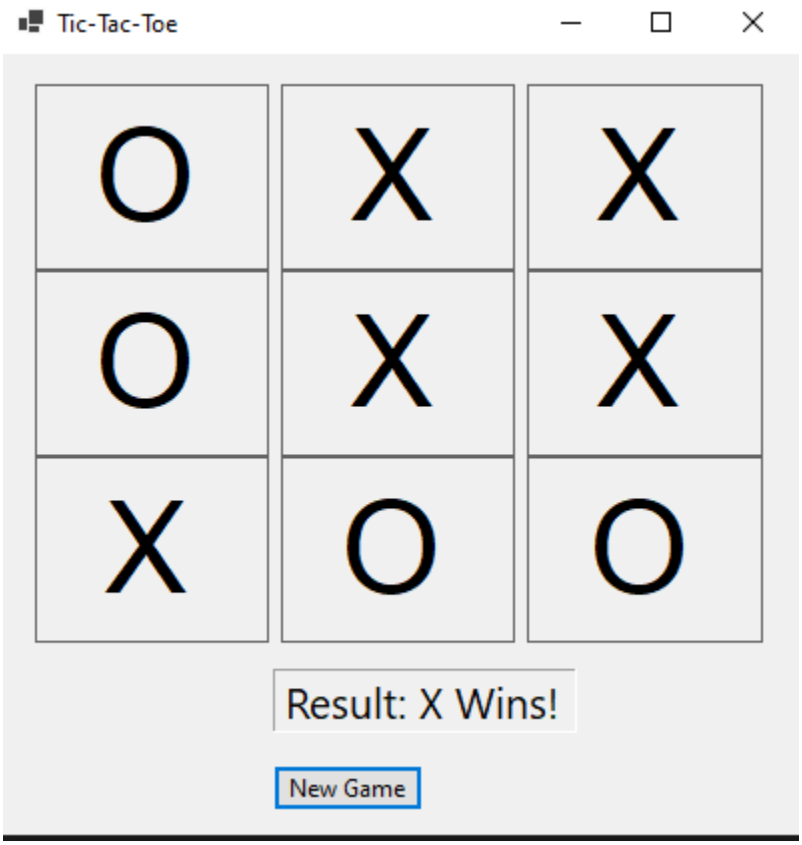
The first method updates the visual display of the game board in the UI. Creates and adds labels for each cell based on the board's values. The CheckForWinner Checks the game board for a winner or a tie. Displays the result if a winner is found or if the game ends in a tie.

Figure 3: Code

```
87          // Check diagonals for a winner
88          if (board[0, 0] == board[1, 1] && board[1, 1] == board[2, 2])
89          {
90              DisplayResult(board[0, 0]); // Display winner if top-left to bottom-right matches
91              return;
92          }
93
94          if (board[0, 2] == board[1, 1] && board[1, 1] == board[2, 0])
95          {
96              DisplayResult(board[0, 2]); // Display winner if top-right to bottom-left matches
97              return;
98          }
99
100         // Check for a tie by ensuring all cells are filled
101         bool isTie = true;
102         foreach (var cell in board)
103         {
104             if (cell == -1) // If there are any unfilled cells, it's not a tie
105             {
106                 isTie = false;
107                 break;
108             }
109         }
110
111         if (isTie)
112         {
113             lblResult.Text = "Result: It's a tie!"; // Display tie result if all cells are filled
114         }
115     }
116
117     /// <summary> Displays the result of the game indicating the winner.
        4 references
121     private void DisplayResult(int winner)
122     {
123         lblResult.Text = winner == 0 ? "Result: O Wins!" : "Result: X Wins!"; // Show who won
124     }
125
126     /// <summary> Starts a new game by re-initializing the game board.
        1 reference
131     private void btnNewGame_Click(object sender, EventArgs e)
132     {
133         InitializeBoard(); // Reset the game board for a new game
134     }
```

This is a continuation of the CheckForWinner method. Then we have DisplayResults that Displays the result of the game indicating the winner. The last method btnNewGame_Click Starts a new game by re-initializing the game board.

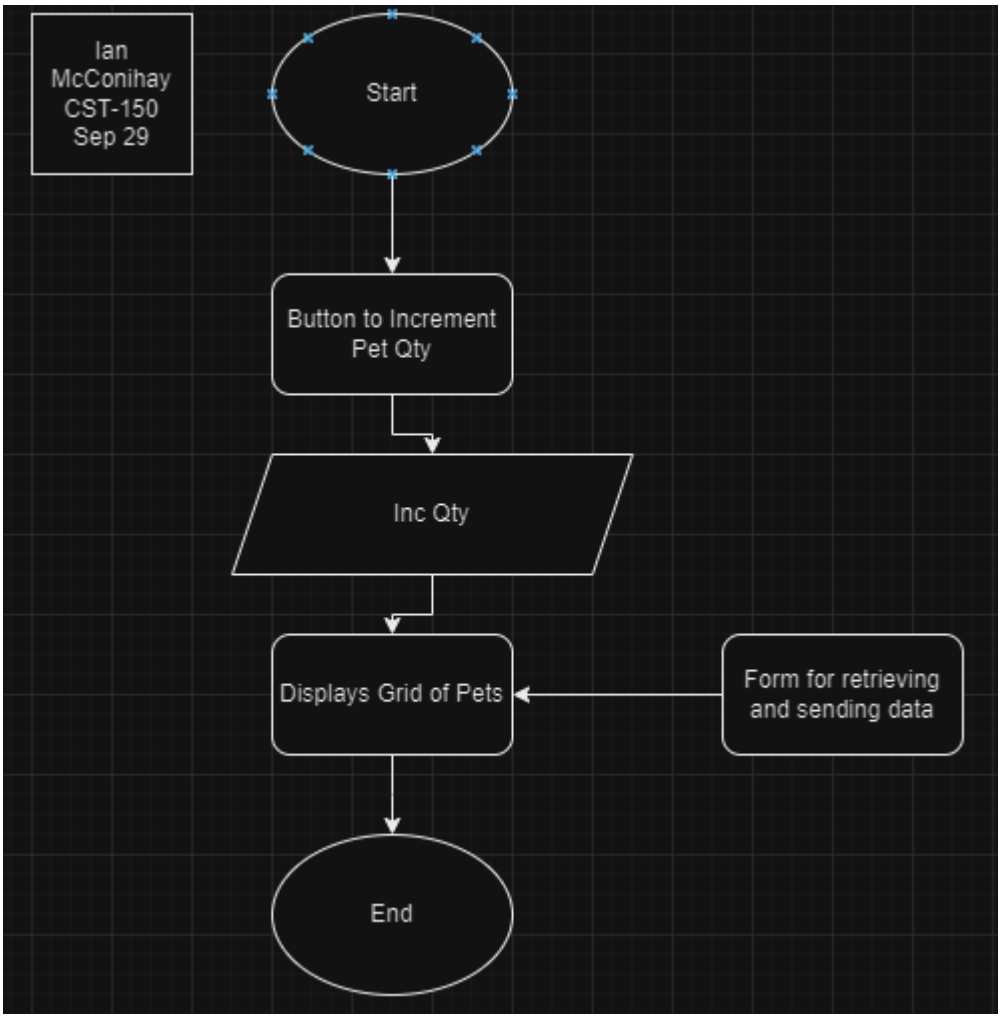
Figure 4: Application Running



The application running shows the button clicked. There is the UI populated in the table layout. There is the label displaying the player results. The Button to create a new game.

Part 2 of Activity 5

Flowchart



Activity 6 part 2 required a flowchart for Activity 7. This is a continuation of Activity 6 part 1. This will be adding a second form to communicate data with the first form.

What was challenging?
This was a tricky challenge dealing with all of the different outcomes.

What did you learn?
I learned about table layouts being used with nested arrays.

How would you improve on the project?
I would like to play the game so probably add the functionality for the player to fill in the boxes.

How can you use what you learned on the job?
Nested arrays and automated result calculations could be used for a variety of uses.