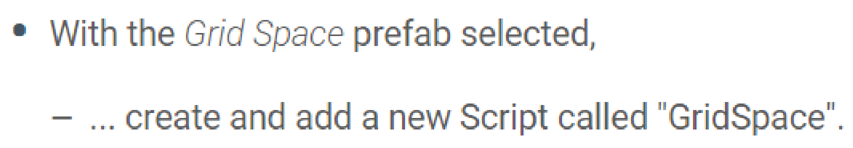
Unity TicTacToe Tutorial Update

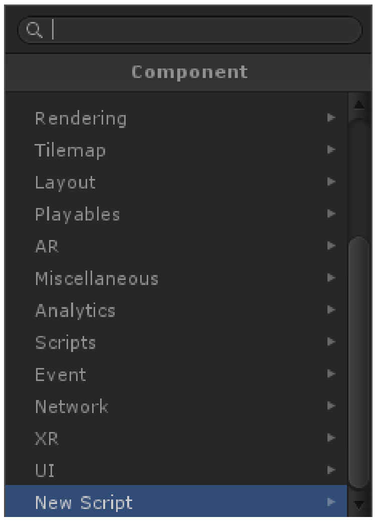
Step 4: Foundation game play

There is clarification necessary to create the GridSpace C# script in the default PC installation as of 3/6/2018.



- With the *Grid Space* prefab selected,
  - ... create and add a new Script called "GridSpace".

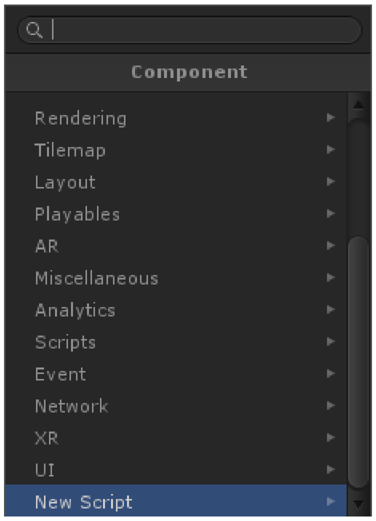
When creating the GridSpace script, select the GridSpace, press the Add Component button, and choose "New Script":



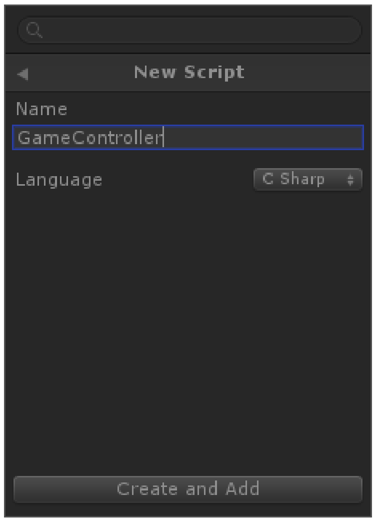
Name it "GridSpace" and click the "Create and Add" button. Then continue as instructed.

Step 5: Controlling the game

When creating the GameController script, select the GameController, press the Add Component button, and choose "New Script":



Name it “GameController” and click the “Create and Add” button:



Name
GameController

Language C Sharp

Then continue as instructed.

06. Win conditions and taking turns

Side note: There’s a better way to implement the EndTurn function.

Define a private field to list all of the 3-in-a-row indices:

```
private int[,] rows = new int[8, 3] { { 0, 1, 2 }, { 3, 4, 5 }, { 6, 7, 8 }, { 0, 3, 6 },
{ 1, 4, 7 }, { 2, 5, 8 }, { 0, 4, 8 }, { 2, 4, 6 } };
```

Then EndTurn() can be implemented more elegantly as:

```
        public void EndTurn()
    {
        for (int i = 0; i < rows.GetLength(0); i++)
        {
            if (buttonList[rows[i, 0]].text == playerSide && buttonList[rows[i, 1]].text
== playerSide && buttonList[rows[i, 2]].text == playerSide)
            {
                GameOver();
                return;
            }
        }
        ChangeSides();
    }
```

(The author's code makes my eyes hurt.)

09. Restarting the game

The author's Restart Button position puts it above the visible area in my application in Play mode given the default Canvas dimensions I have (width:1148, height:550).

One solution:

Restart Button gets repositioned to Pos X = 400, Pos Y = 0.

Bug: A draw doesn't activate the Restart Button in the author's code. I believe the author meant to place the activation of the Restart Button in the common place that both scenarios call, namely "SetGameOverText". So move the following line from GameOver() to SetGameOverText(string):

10. Whose turn is it?

To fit on my canvas,

- Player X position: Pos X = 400, Pos Y = 170
- Player O position: Pos X = 400, Pos Y = 170

Note: Setting player colors seems awfully complicated for what is does. New Player objects? Really? If we must use them, I prefer this code:

```
    void SetPlayerColors()
    {
        Player activePlayer = (playerSide == "X") ? playerX : playerO;
        Player inactivePlayer = (playerSide == "X") ? playerO : playerX;
        activePlayer.panel.color = activePlayerColor.panelColor;
        activePlayer.text.color = activePlayerColor.textColor;
        inactivePlayer.panel.color = inactivePlayerColor.panelColor;
        inactivePlayer.text.color = inactivePlayerColor.textColor;
    }
```

The problem mentioned here doesn't show up in my code, which has a number of edits I've made. There are other improvements that could be made to clean up the code's organization, but here's what I have for GameController after step 10:

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;


[System.Serializable]
public class Player
{
    public Image panel;
    public Text text;
}

[System.Serializable]
public class PlayerColor
{
    public Color panelColor;
    public Color textColor;
}

public class GameController : MonoBehaviour
{
    public Text[] buttonList;
    public GameObject gameOverPanel;
    public Text gameOverText;
    public GameObject restartButton;
    public Player playerX;
    public Player playerO;
    public PlayerColor activePlayerColor;
    public PlayerColor inactivePlayerColor;

    private string playerSide;
    private int moveCount;
    private int[,] rows = new int[8, 3] { { 0, 1, 2 }, { 3, 4, 5 }, { 6, 7, 8 }, { 0, 3,
6 }, { 1, 4, 7 }, { 2, 5, 8 }, { 0, 4, 8 }, { 2, 4, 6 } };

    void Awake()
    {
        SetGameControllerReferenceOnButtons();
        InitGame();
    }

    void InitGame()
    {
        playerSide = "X";
        moveCount = 0;
        gameOverPanel.SetActive(false);
        restartButton.SetActive(false);
        SetPlayerColors();

        for (int i = 0; i < buttonList.Length; i++)
        {
            buttonList[i].text = "";
        }
        SetBoardInteractable(true);
    }

    void SetGameControllerReferenceOnButtons()
    {
        for (int i = 0; i < buttonList.Length; i++)
        {
buttonList[i].GetComponentInParent<GridSpace>().SetGameControllerReference(this);
        }
    }

    public string GetPlayerSide()
    {
        return playerSide;
    }

    public void EndTurn()
    {
        moveCount++;
        for (int i = 0; i < rows.GetLength(0); i++)
        {
            if (buttonList[rows[i, 0]].text == playerSide && buttonList[rows[i, 1]].text
== playerSide && buttonList[rows[i, 2]].text == playerSide)
            {
                GameOver();
                return;
            }
        }
        if (moveCount >= 9)
        {
            SetGameOverText("It's a draw!");
        }
        ChangeSides();
    }

    void ChangeSides()
    {
        playerSide = (playerSide == "X") ? "O" : "X";
        SetPlayerColors();
    }

    void SetPlayerColors()
    {
        Player activePlayer = (playerSide == "X") ? playerX : playerO;
        Player inactivePlayer = (playerSide == "X") ? playerO : playerX;
        activePlayer.panel.color = activePlayerColor.panelColor;
        activePlayer.text.color = activePlayerColor.textColor;
        inactivePlayer.panel.color = inactivePlayerColor.panelColor;
        inactivePlayer.text.color = inactivePlayerColor.textColor;
    }

    void GameOver()
    {
        SetBoardInteractable(false);
        SetGameOverText(playerSide + " Wins!");
    }
    void SetGameOverText(string value)
    {
        gameOverText.text = value;
        gameOverPanel.SetActive(true);
        restartButton.SetActive(true);
    }

    public void RestartGame()
    {
        InitGame();
    }

    void SetBoardInteractable(bool toggle)
    {
        for (int i = 0; i < buttonList.Length; i++)
        {
            buttonList[i].GetComponentInParent<Button>().interactable = toggle;
        }
    }

}
```