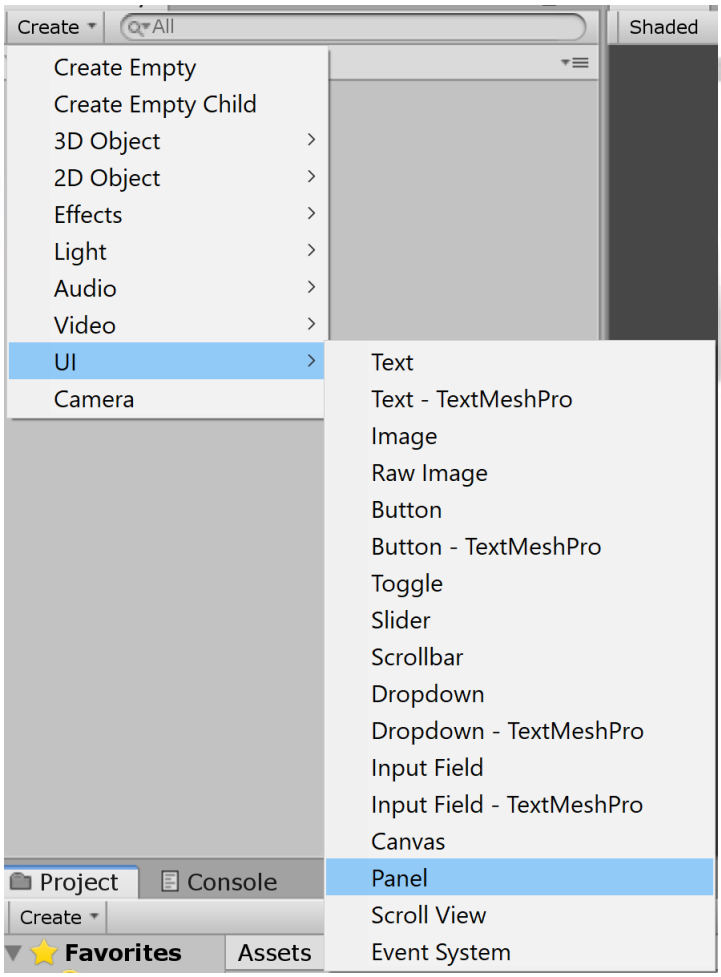
¡PROGRAMAMOS CON UNITY!

1. CREAMOS EL TABLERO DEL TICTACTOE O TRES EN RAYA

Creamos un nuevo proyecto con estilo 2D, una vez estemos en la pantalla inicial de Unity.

Para que nuestro juego tenga un sitio donde “estar”, debemos crear un tablero, para ello en el panel créate, buscaremos la opción **UI/Panel**.

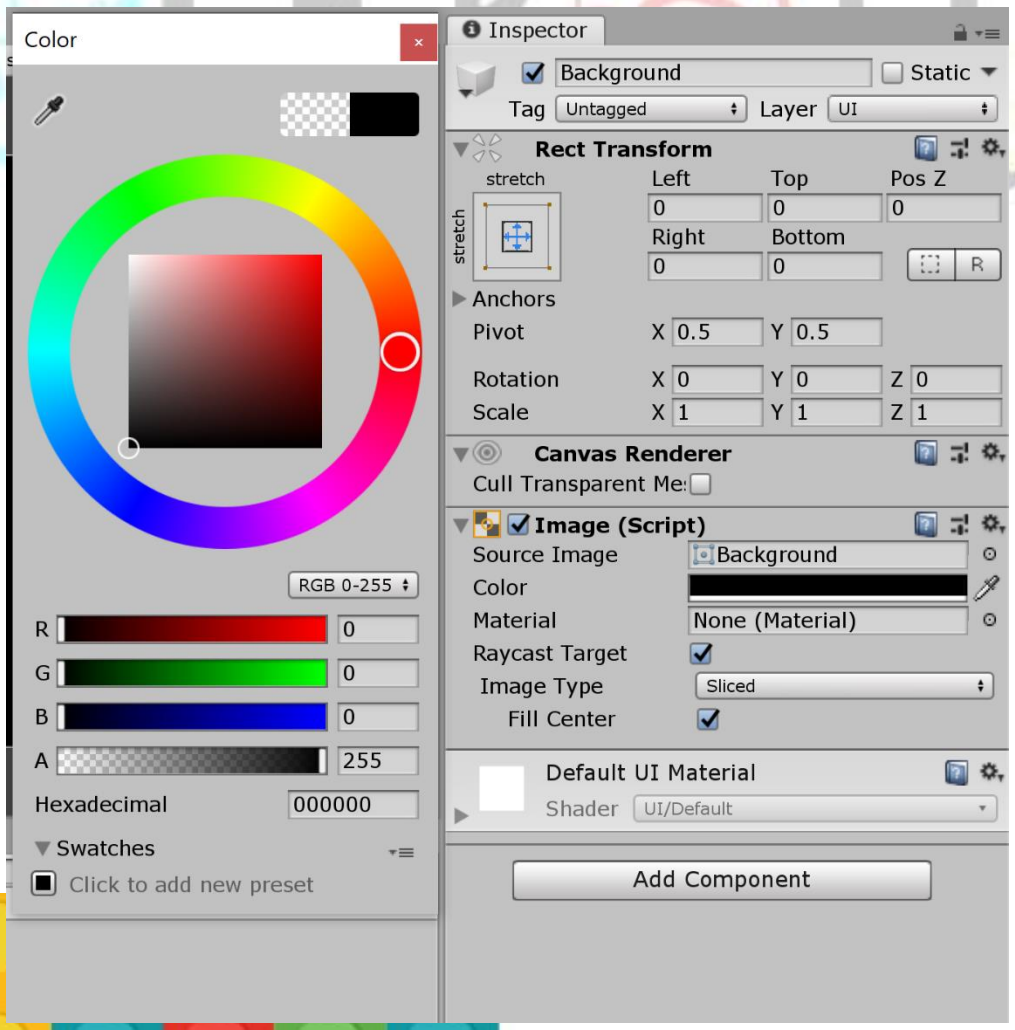


Esto nos crea un nuevo objeto llamado panel que como observamos es hijo de un Canvas, también nos aparece un EventSystem en la jerarquía.

Nos colocaremos de modo que veamos nuestro panel en nuestra pantalla, para ello usaremos la rueda del ratón para quitar el zoom y pulsándola para posicionarnos en el centro del panel.

Seleccionamos el panel y observamos el inspector.

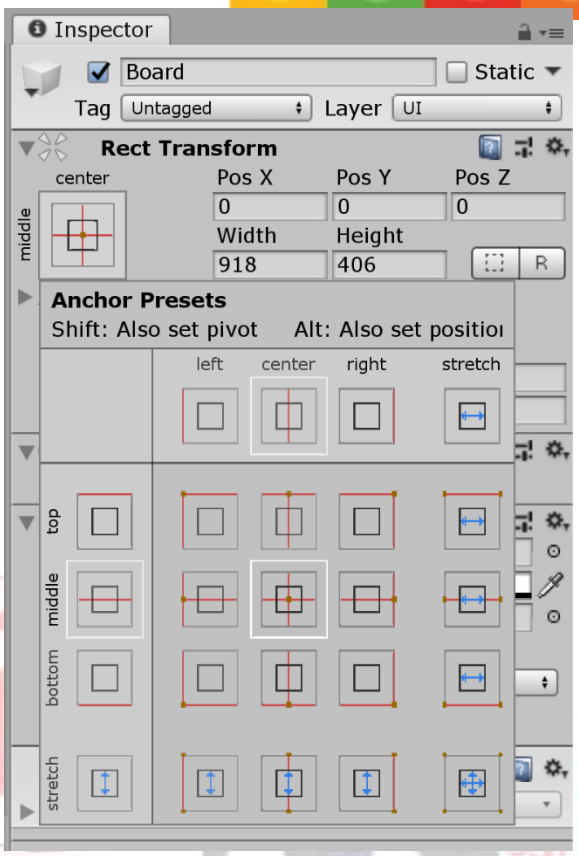
- Cambiamos el nombre por “Background”.
- Cambiamos las propiedades del color del fondo poniéndole en lugar de negro, 100% opacidad.



Ahora, tenemos un panel negro donde trabajar, a continuación, comenzaremos a crear las líneas de nuestro tablero de TicTacToe.

Para hacer esto, debemos crear otro panel y poner las propiedades correspondientes.

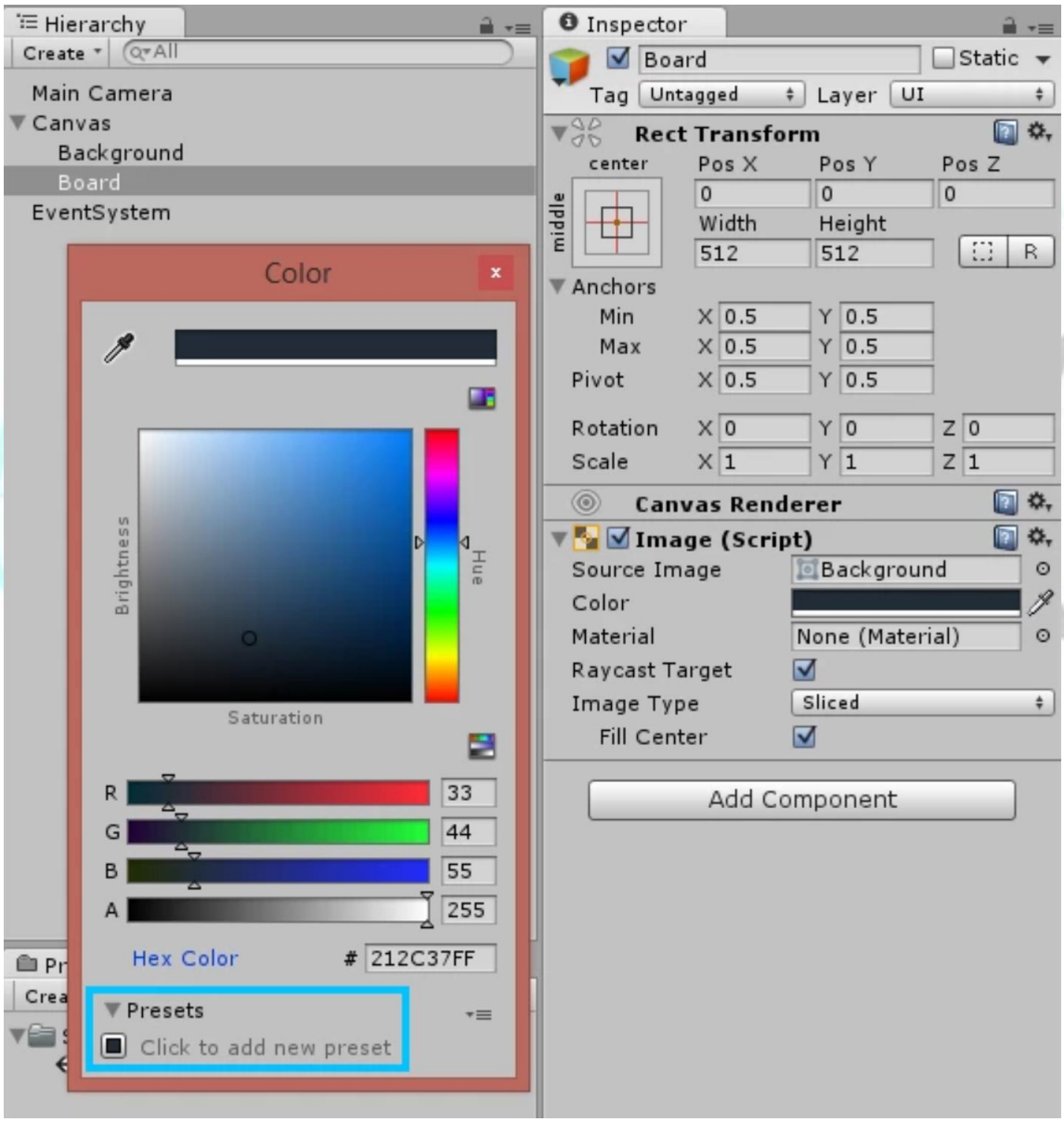
- Crearemos un nuevo UI panel. Create/UI/Panel
- Con el nuevo panel seleccionado cambiaremos su nombre por “Board”
- Pulsaremos en la opción de “Anchor” y dejando pulsadas las teclas “Shift” y “Alt”, seleccionaremos la opción del medio.



Esto nos permite cambiar al centro de nuestro GameObject, como podemos observar, ahora nos aparecen 4 pequeños triángulos en el centro de nuestro canvas.
Ahora haremos que parezca más un tablero, selecciona el objeto del tablero.

- Establece el parámetro **RectTransform’s Width** en 512.
- Establece el parámetro **RectTransform’s Height** en 406.
- En las opciones de la imagen seleccionaremos las propiedades del color, en esta ocasión vamos a ponerlo de un color azul muy oscuro (33, 44, 55, 255) con 100% opacidad.

Una vez puesto el color lo guardaremos como un preset para así asegurarnos que tenemos este color en el futuro.



Llegados a este punto y con el panel en su sitio y colocado, debemos empezar a construir las separaciones de nuestro panel. Para ello

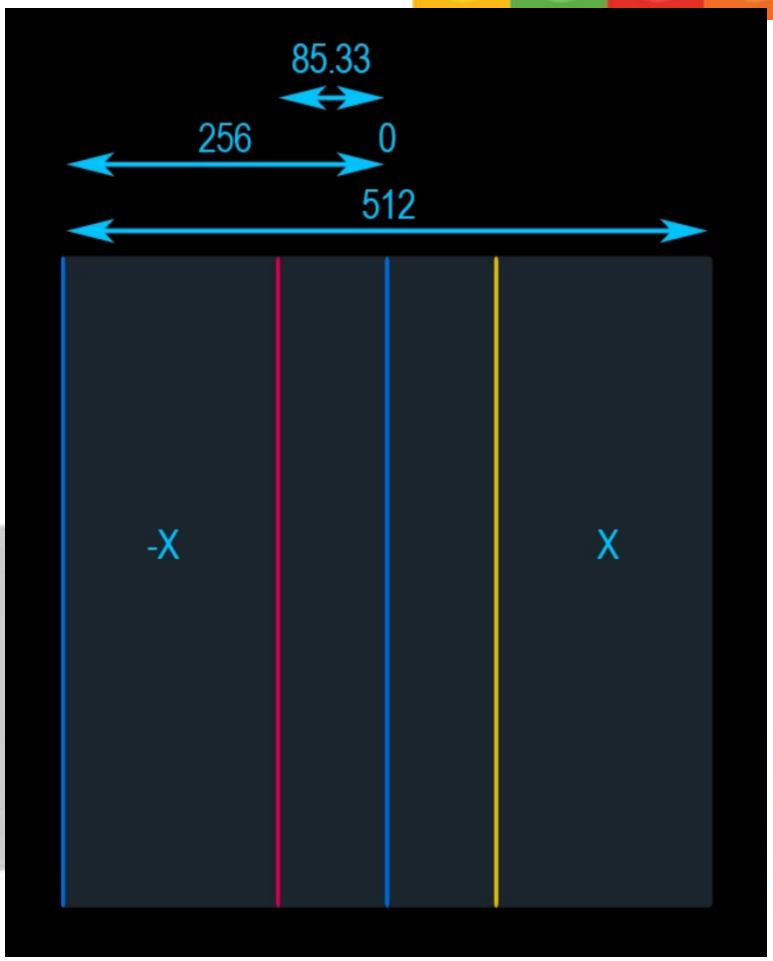
separaremos nuestro tablero en 9 espacios, lo haremos usando el panel de UI.

Para crear nuestra cuadricula:

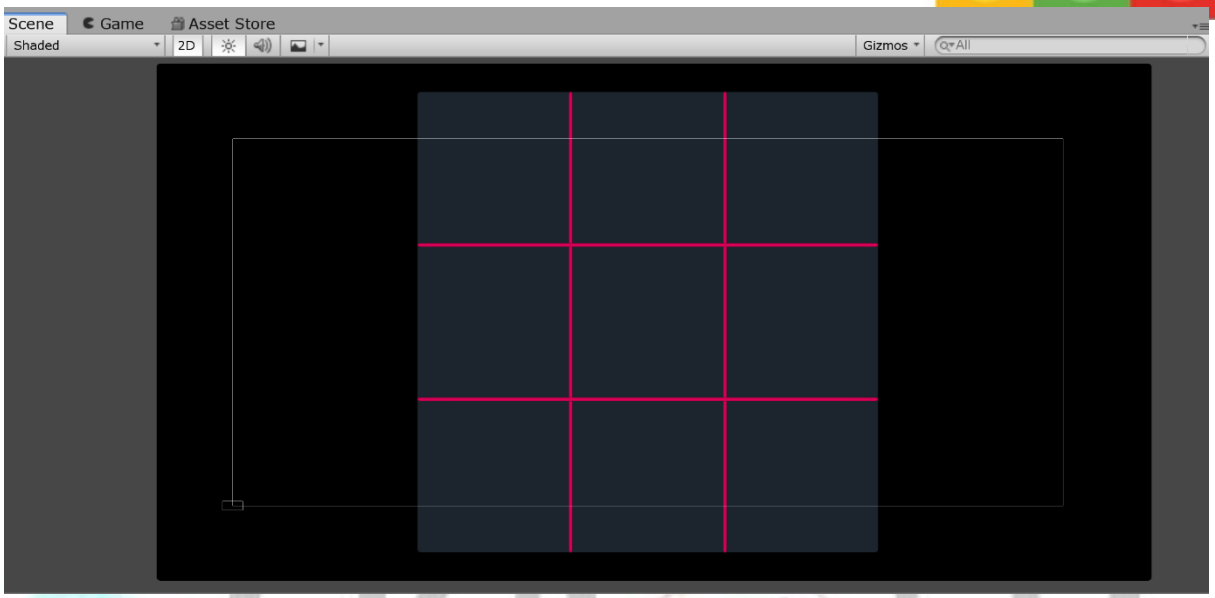
- Creamos un nuevo panel como hemos aprendido a hacer hasta ahora.
- Renombramos nuestro objeto nuevo como “Grid”, y colocamos el Anchor, pivot y Position en el centro.
- El parámetro Width lo establecemos en 5
- El parámetro Height lo establecemos en 512
- Ponemos la Position X en -85.33
- Pondremos un color rosa (255, 0, 102, 255).
- Colocamos el color en los preset para poder usarlo en las siguientes líneas de separación.

¿Cómo obtuvimos el número mágico -85.33 para el valor de Posición X? Esto se hace simplemente con las matemáticas básicas.

El tablero tiene 512 píxeles por 512 píxeles. Si simplemente dividiéramos 512 por 3 obtendríamos un número alrededor de 170 (pronto veremos este número nuevamente). Sin embargo, cuando se trata de posicionamiento en la escena o en el espacio de la pantalla, el mundo comienza en un punto de origen y el valor de las posiciones se aleja de ese punto, positivo en una dirección y negativo en la otra. El tablero del juego está en el centro de la pantalla con una posición X de 0 en su centro y la posición X de -256 y 256 en cada una de las esquinas. ¡Queremos que la línea de la cuadrícula sea 1/3 del camino de 0 a 256, o 256 dividido por 3, lo que equivale a 85.33!



Ahora haremos el resto de las líneas de la cuadricula. Recordad que las posiciones en X cambiarán entre 85.33 y -85.33 tanto en la posición X como en la posición Y.



Al final, debe quedarnos algo parecido a esto.

2. CREAMOS LAS PIEZAS DEL TICTACTOE O TRES EN RAYA

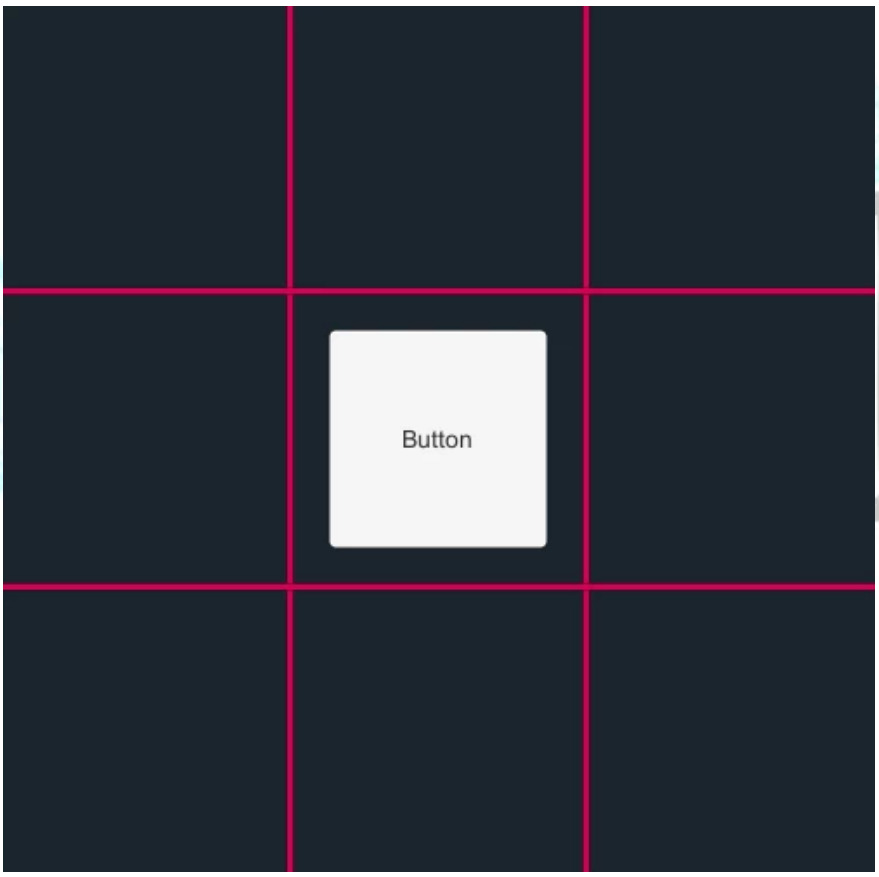
Con nuestro tablero de juego básico configurado, ahora necesitamos encontrar una manera para que nuestros jugadores interactúen con el juego. El jugador debe poder hacer clic en un cuadrado de la cuadrícula y asignar su valor, ya sea "X" u "O", a ese espacio.

Una de las mejores maneras de obtener la interacción de "clic" de un usuario es usar un botón de UI. Los botones pueden detectar un clic y realizar una acción o un conjunto de acciones en respuesta a la entrada. Convenientemente, los botones UI vienen con un elemento de texto adjunto como elemento secundario. Una manera fácil y conveniente de mostrar el valor del jugador, ya sea "X" u "O", es simplemente agregar el movimiento del jugador directamente al elemento de texto UI adjunto como un carácter de texto.

- Creamos un Botón Create > UI > Button
- Con el botón GameObject seleccionado,
    - Renombramos el objeto como “Grd Space”

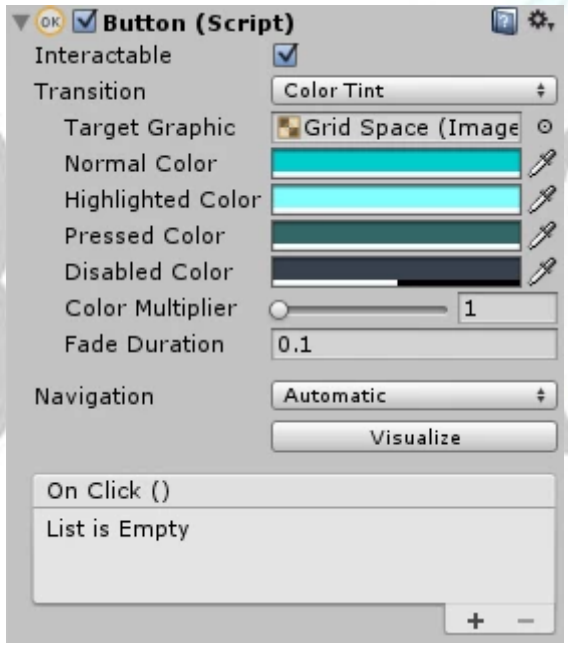
- Pondremos en “Width” en 128
- Pondremos en “Height” en 128
- Reseteamos el “RectTransform”

Esto, nos creará un Botón en medio de nuestro tablero.



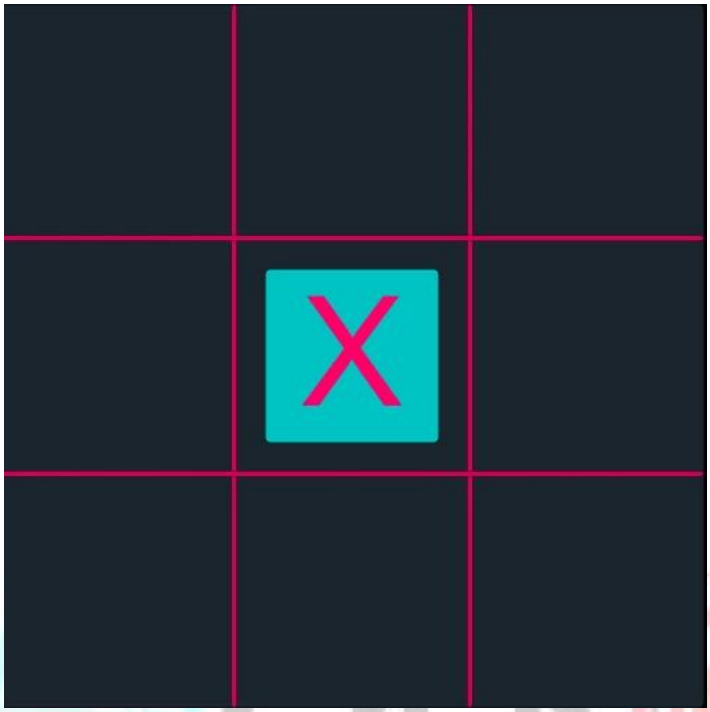
El siguiente paso es establecer el estilo del elemento del botón UI. Haremos esto configurando el color de los elementos de transición en el componente Button y el componente de color Text en el elemento de texto de la interfaz de usuario secundaria.

- Selecciona el objeto Grid Space en la jerarquía.
- Con Grid Space seleccionado
    - Cambiamos el Normal Color a azul (0, 204, 204, 255).
    - El "Highlighted Color" lo cambiamos a un azul más claro (128, 255, 255, 255)
    - La opción de "pressed Color" la pondremos con un azul verdoso oscuro (51, 102, 102, 255)
    - Por último la parte de "Disabled Color" tendrá un azul oscuro (55, 66, 77, 255)



Ahora en la ventana de "Hierachy" abriremos nuestro botón y haremos doble clic en la opción de texto.

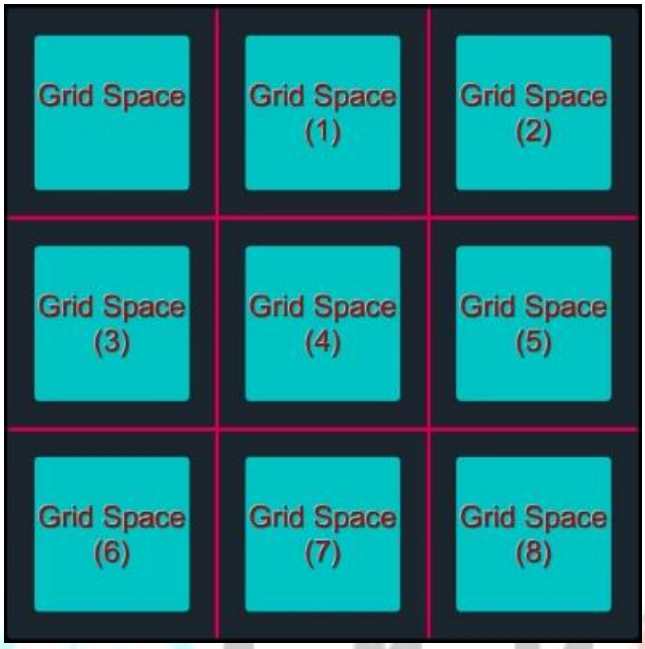
- Cambiaremos el texto por la letra "X"
- El tamaño debe ser 111.
- El color del texto, siguiendo la estética de nuestro panel lo pondremos del mismo color (en mi caso de color rosa (255, 0, 102, 255)) si lo hemos guardado, con dar clic al color es suficiente.



Ahora podemos guardar todo esto como un “Prefab”

- Creamos una nueva carpeta en el “Project Window”
- La llamamos “Prefabs”
- Seleccionamos el “Grid Space” en la ventana de jerarquía y la arrastramos hasta la carpeta que acabamos de crear.

Ahora tenemos nuestro botón de cuadrícula prefabricado. Necesitamos duplicar este botón ocho veces, para un total de nueve botones. Estos se colocarán alrededor del tablero de juego en forma de cuadrícula. Unity creará un nombre único agregando un número al final del nombre de cada GameObject. En la Ventana de Jerarquía, seleccione el Objeto de Juego de Espacio de Cuadrícula y duplíquelo 8 veces.



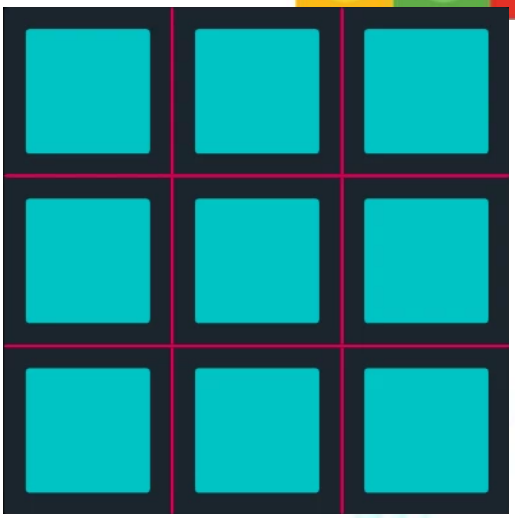
Hay dos formas fáciles de mover estos botones a su lugar. Una es establecer los valores directamente en RectTransform del botón de la interfaz de usuario. El otro es arrastrar los botones en la vista de escena y usar las funciones de colocación y ajuste para alinear los botones.

Para mostrar, por ejemplo, cómo establecer la posición de un botón utilizando directamente RectTransform del botón:

- Seleccionamos el “Grid Space” que queramos editar
    - Vamos cambiando las posiciones de “X” o “Y” a -170 o 170

Ahora borraremos la “X” del texto de nuestros botones, para que quede en blanco (la “X” nos ayuda a colocar mejor los botones si lo vamos a colocar arrastrando los elementos).

Al final nuestro panel de juego tiene que quedar algo así:



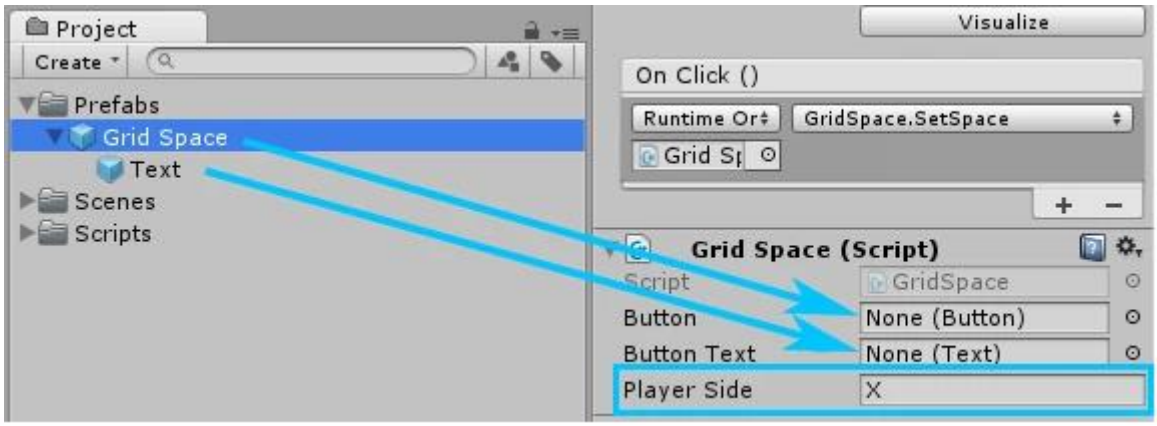
Guardaremos nuestro proyecto. Con esto ya tendríamos todo lo que
es la interface creada, en la siguiente parte, comenzaremos a crear
nuestra programación y una IA para poder jugar.

3. CONTINUAMOS TICTACTOE

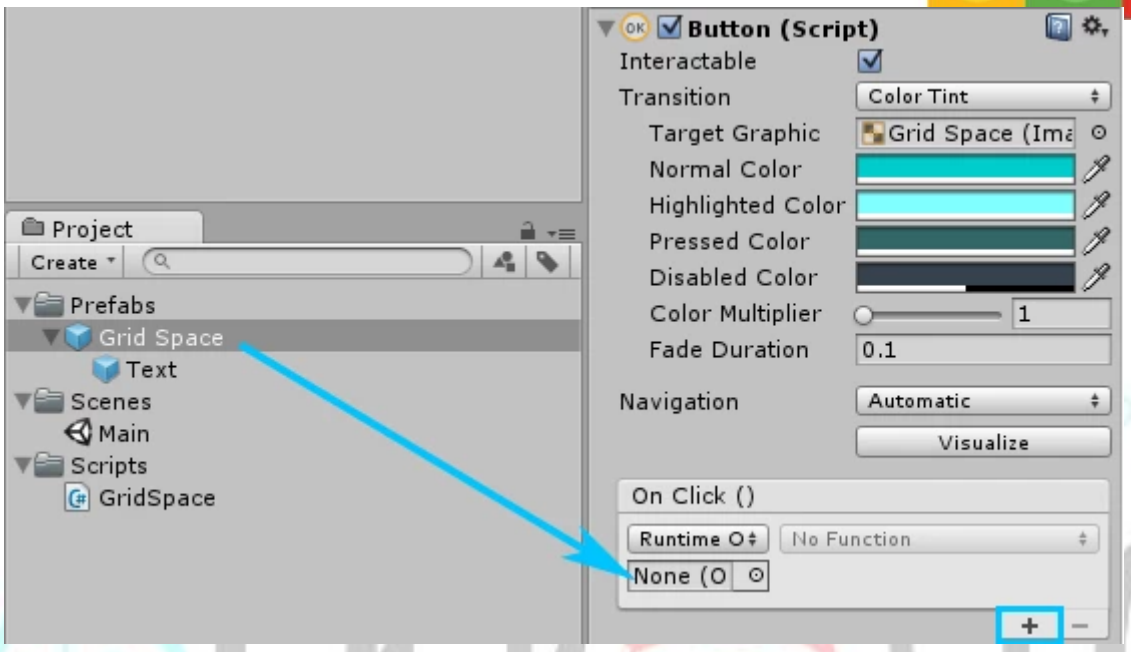
Para empezar con el apartado de programación comenzaremos creando una carpeta en nuestra "Project Window", a la cual llamaremos "Scripts". Dentro de esta carpeta crearemos un nuevo Script que llamaremos "GridSpace", haremos doble clic en el Script para abrir el editor de código.

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;
public class GridSpace : MonoBehaviour
{
    public Button button;
    public Text buttonText;
    public string playerSide;
    public void SetSpace()
    {
        buttonText.text = playerSide;
        button.interactable = false;
    }
}
```

Una vez creado nuestro Script seleccionamos nuestro “Grid Space” en nuestra Project window, arrastraremos nuestro prefab hasta la opción “Button”. Acto seguido arrastramos el hijo “Text” hasta la propiedad de “Button text”. En el apartado de “Player Side”, debemos poner “X”.

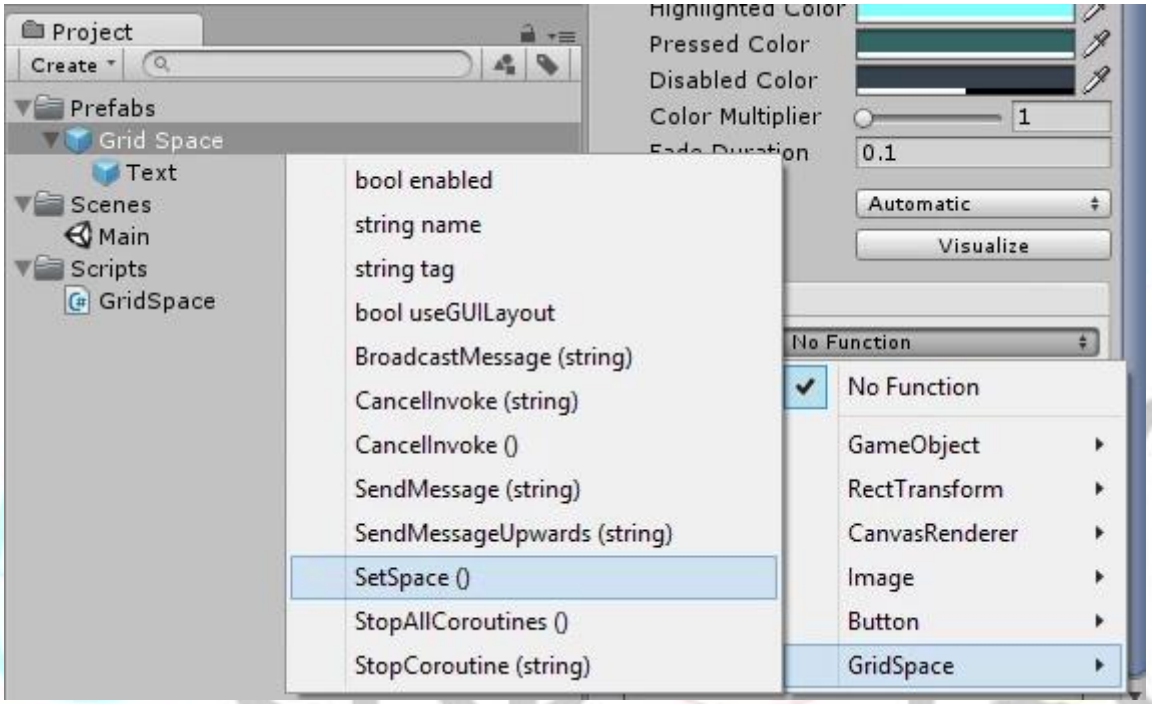


Ahora volveremos a seleccionar nuestro “Grid Space”. añadimos una nueva opción pulsando sobre el símbolo “+”, donde nos aparece la opción “None” arrastramos nuestro prefab..



Con el prefabricado Grid Space seleccionado,

- Con el componente Botón, en la lista desplegable de funciones, seleccione GridSpace> SetSpace.
- Guarda la escena
- Dale al modo reproducción y pulsa en los espacios de la cuadricula



Ahora necesitamos convertir este comportamiento básico en un juego.

Para hacer esto, necesitaremos mover el control a un lugar central creando un Game Controller. El controlador de juego establecerá el lado del jugador inicial, ya sea "X" u "O", hará un seguimiento de quién es el turno y cuando se hace clic en un botón, envía el lado del jugador actual, busca un ganador y cambia de lado o finaliza el juego.

Necesitaremos un nuevo script para hacer esto, así que creemos uno adjunto a su propio GameObject.

- Clic derecho en la ventana de “Hierachy” Create>Create Empty
- Le pondremos como nombre “GameController”
- Creamos un nuevo Script que se llame “GameController” también, haremos doble clic para editar el código

El script de GameController necesitará conocer todos los botones del juego para que pueda verificar su estado y determinar si ha sido una victoria. Para esto, el script debe contener una colección de todos los botones de Grid Space. Queremos verificar el componente de texto del botón para el lado del jugador para ver si hay 3 valores coincidentes en una fila, así que hagamos una matriz del tipo Texto.

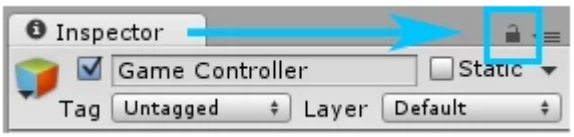
El guion final debería verse así:

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
public class GameController : MonoBehaviour
{
    public Text[] buttonList;
}
```

Ahora que hemos creado esta matriz de Lista de botones, debemos completarla con los componentes de Texto de nuestros botones de Espacio de cuadrícula. Recuerde, ordenar aquí es importante. También es importante que vinculemos los GameObjects de texto secundarios con la matriz Lista de botones, no con los GameObjects de Grid Space primarios.

Una forma de llenar la matriz de la Lista de botones en el Controlador de juegos en el Inspector es establecer el tamaño de la matriz en 9 y luego arrastrar cada GameObject de texto, uno a la vez, a la matriz. Esto es un poco tedioso y consume mucho tiempo. Sin embargo, hay una manera más fácil de hacer esto, e implica "bloquear" la ventana del inspector y arrastrar todos los GameObjects a la vez a la matriz.

- Selecciona el botón del candado en Inspector del GameObject



Lo que esto hace es evitar que la ventana del inspector cambie el foco de la selección actual, en este caso, el controlador del juego. Si no hiciéramos esto, cuando, en el siguiente paso, seleccionáramos los Objetos de Juego de Texto secundarios, el foco del Inspector cambiaría a estos

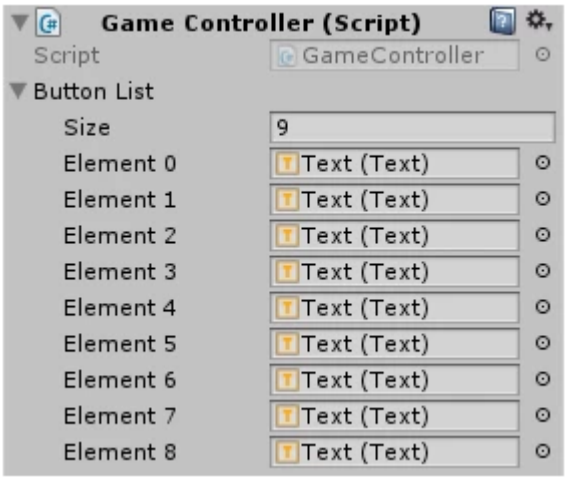
Objetos de Juego de Texto secundarios y no podríamos arrastrarlos a la matriz en el Juego Controlador.

También vale la pena señalar que podemos abrir múltiples ventanas de Inspector, bloqueando solo las que queremos. Esto permite vistas complejas de varios GameObjects en la Jerarquía y Proyecto Windows.

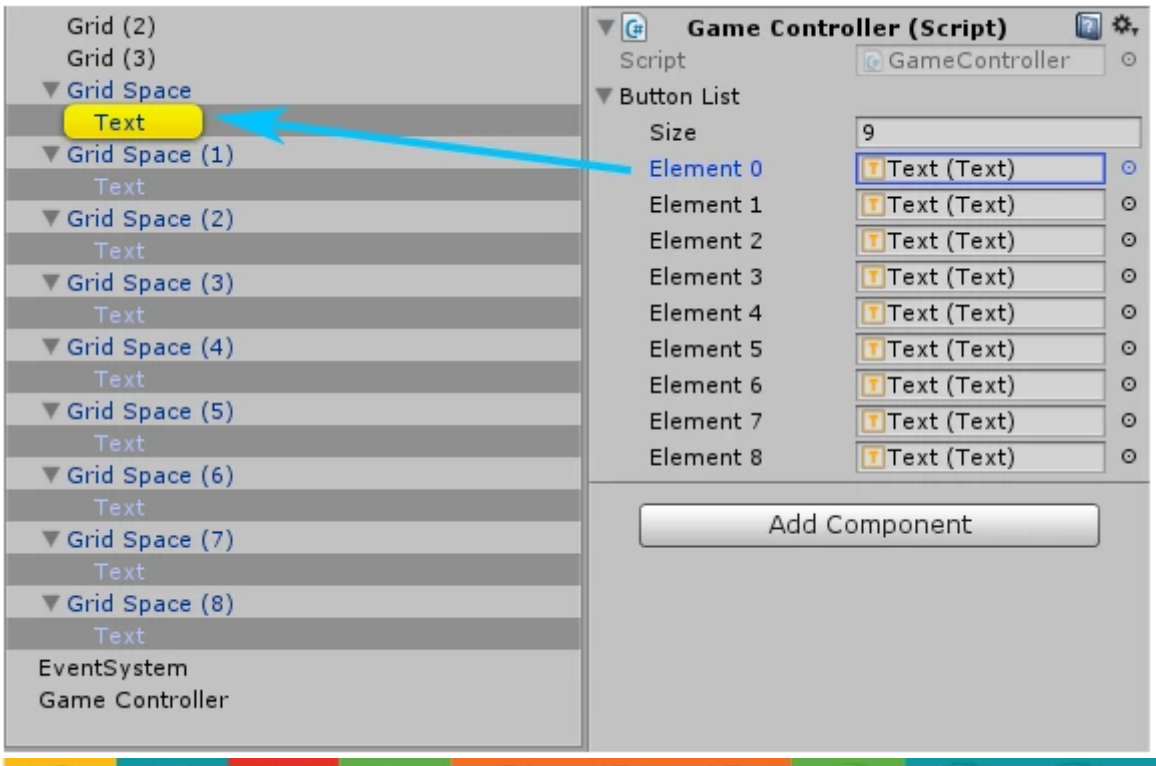
- Abre 9 espacios para los 9 botones en el Inspector haciendo clic en el símbolo “+”

- Arrastra al bloque que le hemos puesto el candado la lista de botones.

Cuando el cursor de arrastre está en la posición correcta, aparecerá un pequeño icono "+" al lado del cursor.



Debemos asegurarnos de que los elementos están colocados en el orden correcto.



Ahora tenemos que avisar al juego de que el jugador ha efectuado un movimiento, para ello debemos informar al controlador de juego que se ha realizado un movimiento y se deben verificar las condiciones de victoria.

Abrimos nuestro Script y colocamos el código.

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

public class GridSpace : MonoBehaviour
{

    public Button button;
    public Text buttonText;

    private GameController gameController;

    public void SetGameControllerReference(GameController controller)
    {
        gameController = controller;
    }

    public void SetSpace()
    {
        buttonText.text = gameController.GetPlayerSide();
        button.interactable = false;
        gameController.EndTurn();
    }
}
```

Probamos que todo funcione de forma correcta.

4. CREANDO TURNOS

Necesitamos probar los espacios de la cuadrícula en el tablero y ver si ha habido una victoria. No tenemos los lados del jugador correctamente configurados, pero necesitaremos verificar la propiedad Text de los espacios de la cuadrícula para ver si los valores de la cadena coinciden con "tres en una línea" y si estos valores coinciden con el lado que se está jugando actualmente. Para hacer esto, tendremos que poner el siguiente código.

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

```
public class GameController : MonoBehaviour
{

    public Text[] buttonList;

    private string playerSide;

    void Awake()
    {
        SetGameControllerReferenceOnButtons();
        playerSide = "X";
    }

    void SetGameControllerReferenceOnButtons()
    {
        for (int i = 0; i < buttonList.Length; i++)
        {

buttonList[i].GetComponentInParent<GridSpace>().SetGameControllerReference(this);
        }
    }

    public string GetPlayerSide()
    {
        return playerSide;
    }

    public void EndTurn()
    {
        if (buttonList[0].text == playerSide && buttonList[1].text == playerSide
&& buttonList[2].text == playerSide)
        {
            GameOver();
        }

        if (buttonList[3].text == playerSide && buttonList[4].text == playerSide
&& buttonList[5].text == playerSide)
        {
            GameOver();

       }

        if (buttonList[6].text == playerSide && buttonList[7].text == playerSide
&& buttonList[8].text == playerSide)

        {
            GameOver();
        }

        if (buttonList[0].text == playerSide && buttonList[3].text == playerSide
&& buttonList[6].text == playerSide)
```

```
        {
            GameOver();
        }

        if (buttonList[1].text == playerSide && buttonList[4].text == playerSide
&& buttonList[7].text == playerSide)
        {
            GameOver();
        }

        if (buttonList[2].text == playerSide && buttonList[5].text == playerSide
&& buttonList[8].text == playerSide)
        {
            GameOver();
        }

        if (buttonList[0].text == playerSide && buttonList[4].text == playerSide
&& buttonList[8].text == playerSide)
        {
            GameOver();
        }

        if (buttonList[2].text == playerSide && buttonList[4].text == playerSide
&& buttonList[6].text == playerSide)
        {
            GameOver();
        }
    }

    void GameOver()
    {
        for (int i = 0; i < buttonList.Length; i++)
        {
            buttonList[i].GetComponentInParent<Button>().interactable = false;
        }
    }
}
```

Ahora, todas las condiciones ganadoras posibles deberían funcionar y cuando se hayan cumplido las condiciones ganadoras, el tablero debería estar desactivado.

Luego, necesitamos cambiar de bando cuando se realiza un turno y poder jugar contra otra persona.

En la actualidad no tenemos realmente un juego. Podemos verificar las condiciones de victoria y cuando obtenemos tres "X" seguidas, el juego termina. Ahora necesitamos poder cambiar de bando. Para

cambiar de lado, necesitaremos verificar de qué lado estamos y alternar los valores del jugador; intercambiando "X" por "O" o viceversa.

Para hacer esto, creemos una función que cambie los lados del jugador.

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

public class GameController : MonoBehaviour
{

    public Text[] buttonList;

    private string playerSide;

    void Awake()
    {
        SetGameControllerReferenceOnButtons();
        playerSide = "X";
    }

    void SetGameControllerReferenceOnButtons()
    {
        for (int i = 0; i < buttonList.Length; i++)
        {

buttonList[i].GetComponentInParent<GridSpace>().SetGameControllerReference(this);
        }
    }

    public string GetPlayerSide()
    {
        return playerSide;
    }

    public void EndTurn()
    {

        if (buttonList[0].text == playerSide && buttonList[1].text == playerSide
&& buttonList[2].text == playerSide)
        {
            GameOver();
        }

        if (buttonList[3].text == playerSide && buttonList[4].text == playerSide
&& buttonList[5].text == playerSide)
        {
            GameOver();
        }
```

```
        if (buttonList[6].text == playerSide && buttonList[7].text == playerSide
&& buttonList[8].text == playerSide)
        {
            GameOver();
        }


        if (buttonList[0].text == playerSide && buttonList[3].text == playerSide
&& buttonList[6].text == playerSide)
        {
            GameOver();
        }

        if (buttonList[1].text == playerSide && buttonList[4].text == playerSide
&& buttonList[7].text == playerSide)
        {
            GameOver();
        }

        if (buttonList[2].text == playerSide && buttonList[5].text == playerSide
&& buttonList[8].text == playerSide)
        {
            GameOver();
        }

        if (buttonList[0].text == playerSide && buttonList[4].text == playerSide
&& buttonList[8].text == playerSide)
        {
            GameOver();
        }

        if (buttonList[2].text == playerSide && buttonList[4].text == playerSide
&& buttonList[6].text == playerSide)
        {
            GameOver();
        }

        ChangeSides();

    }

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

    void GameOver()
    {
        for (int i = 0; i < buttonList.Length; i++)
        {
            buttonList[i].GetComponentInParent<Button>().interactable = false;
        }
```

```
    }
}
```

Ahora, cuando hacemos clic en cualquier espacio, los turnos se alternan entre "X" y "O", y si obtenemos tres "X" o tres "O" seguidas, cumplimos las condiciones de victoria y el juego ha terminado.

¡Nuestro juego está esencialmente terminado! En esta etapa, podemos jugar al Tic-Tac-Toe.