Introducción a la programación de videojuegos

Sebastián Díaz Arancibia

Universidad Técnica Federico Santa María - 2023

Tabla de contenido

01.

Scene Manager

Encargado de administrar las escenas del videojuego: cambiar de una escena a otra, consultar en qué nivel se encuentra el jugador, etc.

BRAND NAME

Scene Manager

Al **completar un nivel** de un juego de acción, al presionar el botón Play en el Main Menu, **al perder** en un juego de carreras, **al comenzar** la los créditos de un juego de aventuras. En todos estos casos, se están **utilizando cambios de escena**.

Muchos videojuegos los utilizan para **anunciar el Game Over** como en Donkey Kong Country 3: Dixie Kong 's Double Trouble!, Super Mario 64.

Otros videojuegos los utilizan para **mostrar al jugador** a los luchadores disponibles en juegos como en Street Fighters IV.

En el caso de The Legend of Zelda: Breath of the Wild, **cada vez que hay una pantalla de carga**, es porque está cambiando de escena.

Cambiar una escena significa que se destruirán todos los GameObjects de una escena y se cargarán otros GameObjects completamente nuevos en una escena nueva.

Scene Manager

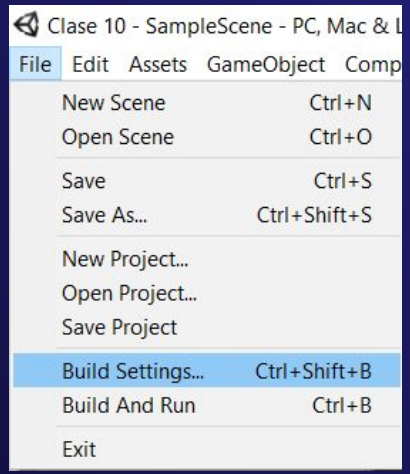
En Unity existe un responsable de gestionar las escenas, esto quiere decir que podremos acceder a información importante y manipularla según nuestros requerimientos. Este responsable se llama **SceneManager**.

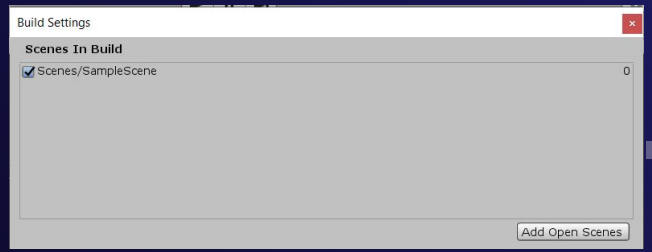
SceneManager es una clase proporcionada por Unity que se utiliza para gestionar escenas en un proyecto.

Una escena en Unity **es un contenedor** que contiene objetos, luces, cámaras y otros elementos de un juego.

La clase **SceneManager** permite cargar, descargar, cambiar y administrar las escenas dentro del proyecto.

Scene Manager





Para ver las escenas que Unity cargará en nuestro juego, debemos entrar a la **configuración de la build** o ejecutable.

En la parte superior de la ventana emergente, se pueden arrastrar las escenas del proyecto desde la ventana Project hasta la ventana de Build Settings, donde aparecen listadas las escenas, indicando: Un **ticket** si **la escena será incluída en la build**, el **nombre** junto a la ruta de la escena y luego su **número identificador**.

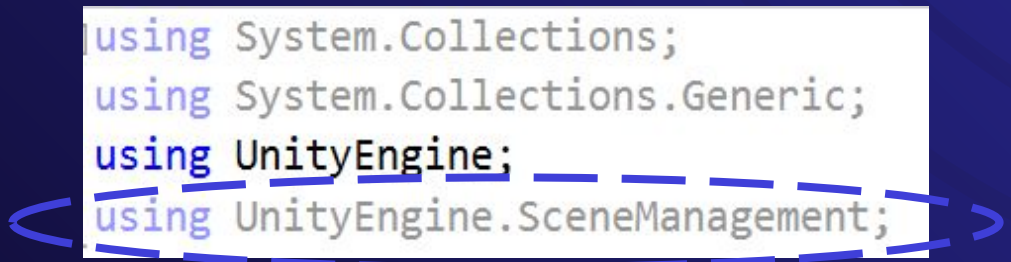
El botón **Add Open Scenes** agrega la escena que se encuentra abierta en el editor de Unity.

Scene Manager

UnityEngine.SceneManagement

Primero, se debe asegurar que **todo Script** que vaya a utilizar las funciones del Scene Manager, **debe agregar** apropiadamente el namespace o librería correspondiente.

Agregando solo una línea de código a cualquier Script de C# que vaya a utilizar **SceneManager**, **antes de la declaración public class del Script**, se agrega la librería **SceneManagemet**.

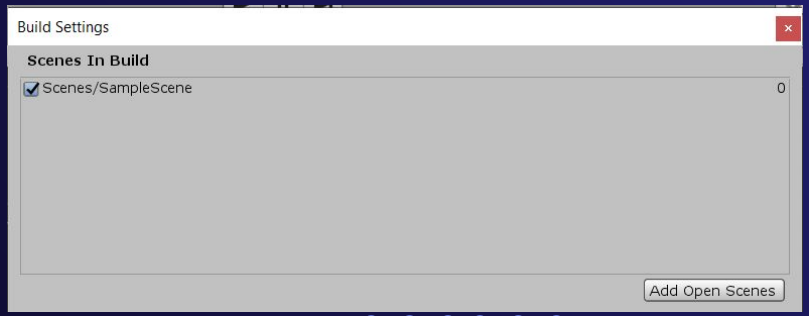


Scene Manager

SceneManager.LoadScene( )

Para cargar una nueva escena, se utiliza la función **SceneManager.LoadScene( )**, la cual puede recibir como parámetro de entrada una variable del tipo **string** con el **nombre de la escena**, o un número entero **int** con el **identificador de la escena**.

```
public class CambioScene : MonoBehaviour
{
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            SceneManager.LoadScene(0);
        }
    }
}
```



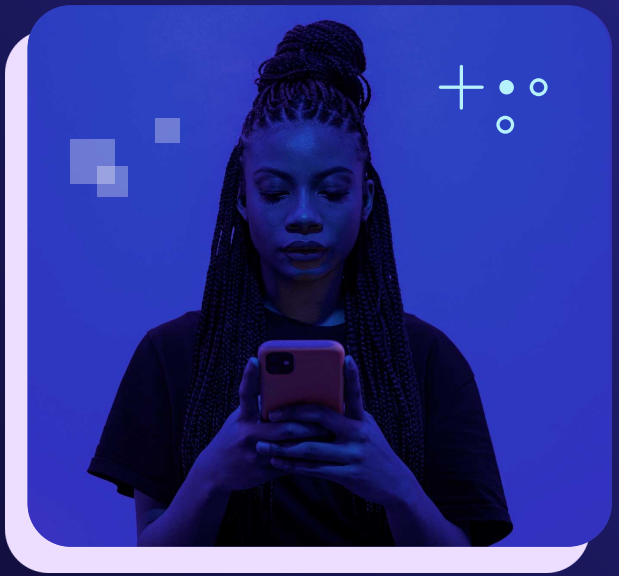

```
public class CambioScene : MonoBehaviour
{
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            SceneManager.LoadScene("Clase9Scene");
        }
    }
}
```

Scene Manager

SceneManager.GetActiveScene( )

- Puedes obtener información sobre la escena actualmente activa utilizando **SceneManager.GetActiveScene( )**, una función que permite acceder a detalles como el nombre y el índice de la escena activa.

```
public class ScoreInfo : MonoBehaviour
{
    Unity Message | 0 references
    private void Start()
    {
        Scene activeScene = SceneManager.GetActiveScene();
        string sceneName = activeScene.name;
        Debug.Log("Esena activa: " + sceneName);
    }
}
```



Ejercicio 1

Cree un nuevo Script llamado CambioScene. El script consiste en activar el final del nivel. Declare una variable booleana pública que comience en false. Cuando la variable cambie su valor, debe contar hasta 3 y luego cambiar a la escena 0. Arrastrar el script a un GameObject vacío y poner play para ver el comportamiento al cambiar el valor desde el inspector.

Ejercicio 1

```
public class CambioScene : MonoBehaviour
{
    public bool isEnd = false;
    public int timeToLoadScene = 3;
    public float timer = 0;

    void Update()
    {
        if (isEnd)
        {
            timer += Time.deltaTime;
            if (timer >= timeToLoadScene)
            {
                isEnd = false;
                timer = 0;
                SceneManager.LoadScene(0);
            }
        }
    }
}
```

02.

User Interface

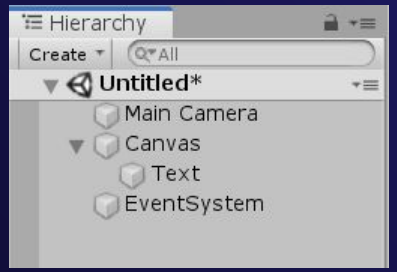
User Interface hace referencia a la interfaz con la cual el usuario podrá comunicarse con un videojuego: botones, barra de vida, texto con el puntaje o el tiempo de la partida, menú, etc.

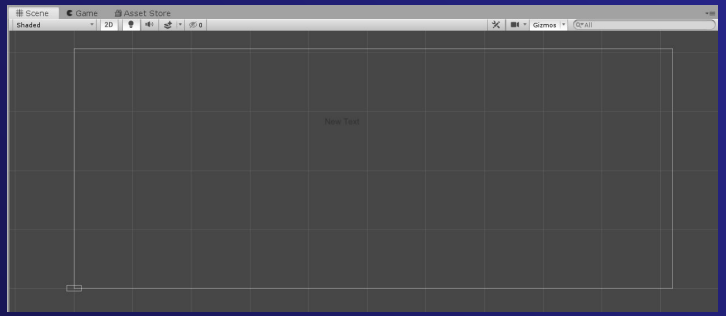
User Interface

El Canvas es el **área donde se encuentran todos los elementos UI**. Es un GameObject con un **componente Canvas** en él, y todos los elementos UI deben ser hijos de dicho Canvas. Los Elementos UI en el Canvas son dibujados en el mismo orden que aparecen en la jerarquía.

Creando un nuevo elemento UI, **automáticamente crea** un Canvas si ya **no existe** otro en la escena. El elemento UI es creado **como hijo** de este Canvas.

El área Canvas **es mostrado** como un rectángulo en la **vista de Scene** pero mucho más grande que el rectángulo de la **vista de la cámara**. Son dimensiones distintas.





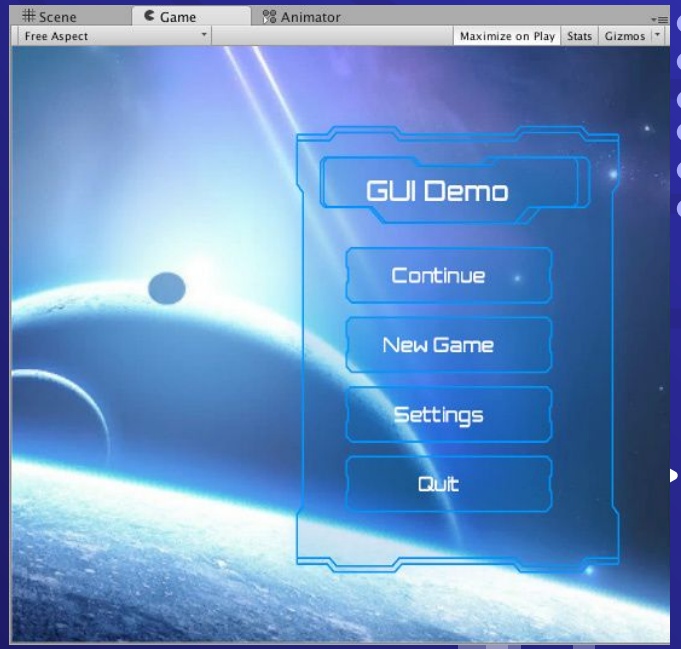
User Interface

El Canvas tiene un ajuste de Render Mode el cual puede ser utilizado para renderizar en el espacio de la pantalla o el espacio del mundo.

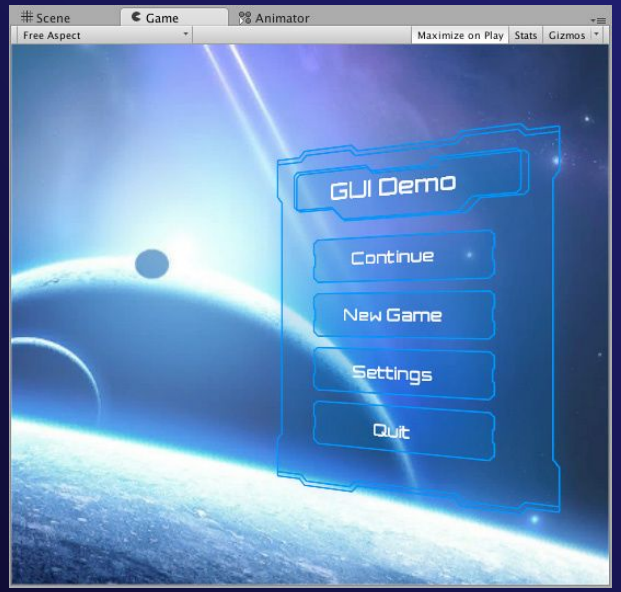
Screen Space - Overlay

El Canvas se superpone a la escena y se muestra en la parte superior de todos los demás elementos. No está afectado por la perspectiva de la cámara y siempre se muestra en la misma posición relativa a la pantalla, **sin importar la posición o rotación de la cámara**.

Este modo de renderización coloca elementos UI en la pantalla mostrada en la parte superior de la escena. Si el tamaño de la pantalla es modificada o cambia la resolución, el Canvas va a automáticamente cambiar el tamaño para que coincida.



User Interface



Screen Space - Camera

Esto es similar a Screen Space - Overlay, pero en este modo de renderizado, el Canvas se renderiza en la pantalla pero **está ligado a una cámara** específica en la escena. El Canvas se ajusta a la vista de la cámara y se mueve, escala y rota junto con ella.

Se puede seleccionar la cámara objetivo en el **componente Canvas** para determinar cuál cámara se utiliza para el renderizado.

Los elementos de la interfaz de usuario son renderizados por esta cámara, lo que significa que **la configuración de la cámara afecta la apariencia de la interfaz de usuario**.

User Interface

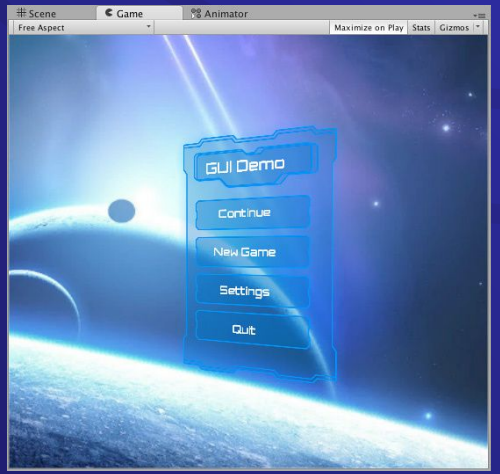
World Space

En este modo de renderizado, el Canvas se va a **comportar como cualquier otro objeto en la escena**.

El tamaño de este Canvas puede ser configurado manualmente utilizando su Rect Transform, y los elementos UI van a renderizar al frente o detrás de otros objetos en la escena basados en una colocación 3D.

Se puede posicionar y orientar en el espacio 3D, lo que permite que los elementos de la UI interactúen con la escena y los objetos del juego. El Canvas en modo World Space también **puede ser afectado por las luces y sombras de la escena**.

Esto es útil para UI que están destinadas a ser parte del mundo. Esto también es conocido como **diegetic interface**.



User Interface

Unity provee una variedad de componentes visuales que permitirán al jugador interactuar con las UI en el entorno virtual que se está desarrollando:

Text (Legacy)

El componente Text tiene un área Text para **ingresar el texto que será mostrado**. Es posible configurar un tipo de fuente, estilo de fondo, tamaño de fondo y si el texto tiene una capacidad de Rich Text.

También cuenta con opciones para configurar el **alineamiento del texto**, configuraciones para un **desbordamiento horizontal y vertical** que controla lo que sucede si el texto es más grande que el rectángulo, y una opción de encaje perfecto para que el texto cambie el tamaño para encajar con el espacio disponible.

User Interface

Para modificar un Text desde un Script, primero se debe agregar el namespace correspondiente. Luego, obtener la referencia del **Text** y modificar el valor de la **propiedad text**.

```
using UnityEngine.UI;
```

```
public Text t;
```

```
t.text = "Nuevo texto";
```

User Interface

Image

Un **Image** tiene un componente Rect Transform y un componente Image.

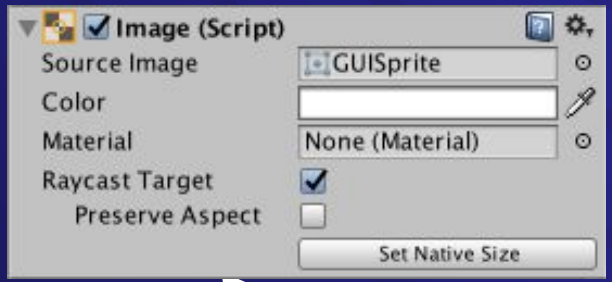
Un Sprite puede ser aplicado al componente Image en **Source Image**, y su color puede ser configurado en el campo **Color**.

El campo Image Type de un objeto **Sprite** define cómo esta imágen se comportará al escalar:

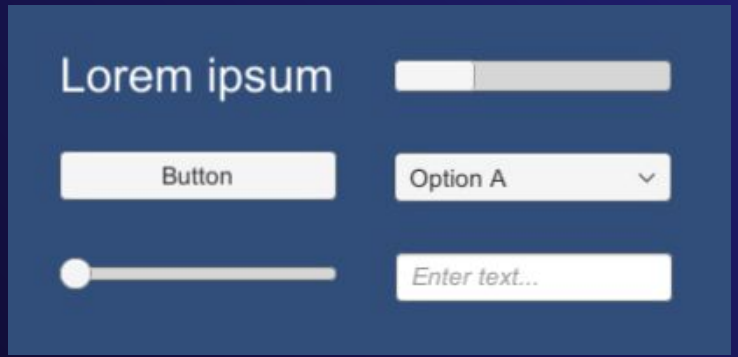
- Simple
- Sliced
- Tiled
- Filled

En el componente **Image** la opción **Set Native Size**, que es mostrada cuando Simple o Filled es seleccionada, **restablece la imagen al tamaño original del Sprite**.

Las imágenes pueden ser importadas como UI sprites al seleccionar el objeto Sprite y ajustar el Texture Type a **Sprite(2D / UI)**.



User Interface



Los componentes de interacción **no son visibles por ellos mismos**, y deben ser combinados con uno o más elementos visuales con el fin de que funcionen correctamente.

La mayoría de componentes de interacción tienen algunas cosas en común: Tienen una funcionalidad integrada compartida para **visualizar transiciones entre estados**: normal, highlighted, pressed, disabled, y para navegar a otros seleccionables utilizando el teclado u otro controlador.

Los componentes de interacción tienen al menos un **UnityEvent** que se invoca cuando el usuario interactúa con el componente de manera específica.

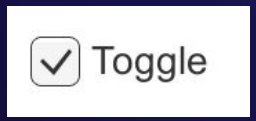
User Interface

Botón

Un botón tiene un UnityEvent OnClick para definir lo qué hará cuando sea presionado.

Toggle

Cuadros de tickets o checkbox que determina cuando el interruptor está encendido o apagado.

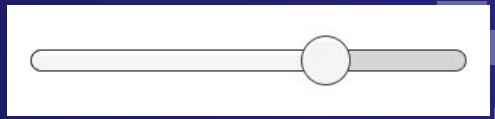


Toggle Group

Puede ser utilizado para un grupo de un conjunto de Toggles que son mutuamente excluyentes, sólo uno de ellos pueda ser seleccionados a la vez.

Slider

Posee un número Value decimal el cual el usuario puede arrastrar entre un valor mínimo y máximo.



User Interface

Button (Legacy)

Para el caso de los botones, el **Unity Event OnClic( )** es la función encargada de recibir la interacción de clic sobre el botón.

Desde el Inspector del objeto con el componente Button, Al final de las propiedades, se encuentra el **Unity Event OnCLick( )**.

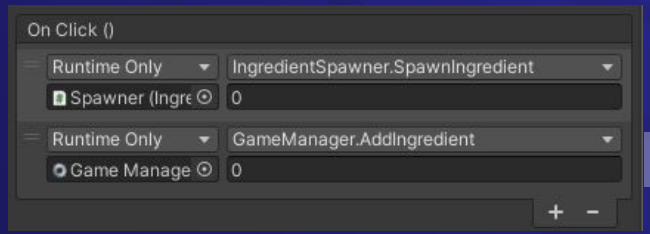
Al presionar **+** se creará una fila con campos editables.

Se debe posicionar el GameObject **que tiene** el componente **que tiene** la función que queremos **activar** cada vez que se **presione clic** sobre el botón.

En la imágen se alcanza a apreciar que bajo el dropdown **Runtime Only**, hay un objeto llamado Spawner.

A la derecha de **Runtime Only** se debe seleccionar el componente que tienen la función que se va a utilizar, en este caso **SpawnIngredient( )**.

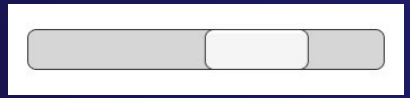
Si la función a invocar tiene UN parámetro de entrada, podra modificarlo del mismo inspector, tal como aparecen los 0 en la imagen, que es el indice del ingrediente a Spawnear.



User Interface

Scrollbar

Tiene un número decimal Value entre 0 y 1. Cuando el usuario arrastre el scrollbar, el valor cambia de acuerdo a esto.



Dropdown

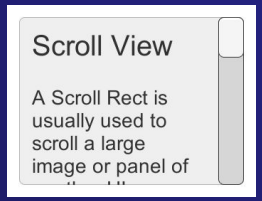
Un despegable tiene una lista de opciones para escoger, un string de texto y opcionalmente una imagen.

Input Field

Utilizado para hacer que el texto de un Text Element editable por el usuario.

Scroll Rect

Similar a un Input Field pero cuando el texto es más grande que el contenedor, proporciona la funcionalidad para desplazarse hacia el contenido.



User Interface

Input Field

Es muy común en los videojuegos tener una cuenta de acceso, poner nombres a los personajes o los archivos de guardado.

El texto utilizado en esos campos son los **Input Field**, los cuales son encargados de **gestionar todo el texto ingresado** por el jugador:

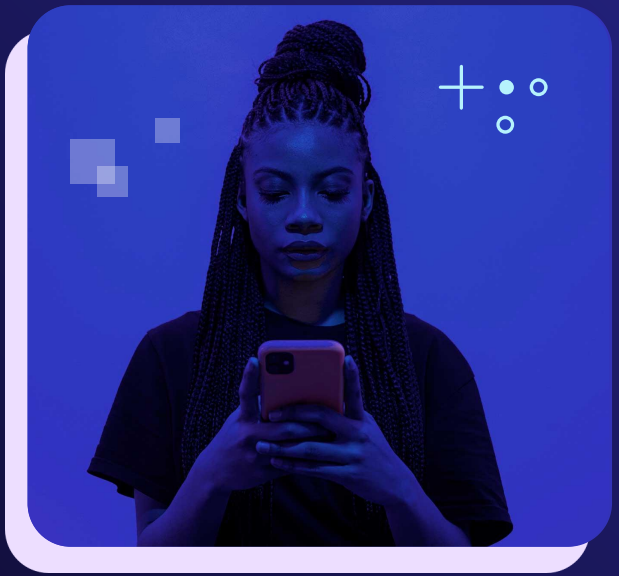
- CharacterLimit
- Interactable
- Text
- GameObject
- RectTransform
- Y muchos más!

Desde el punto de vista del código, se debe crear una referencia para el **Input Field**.

Luego usando esa variable podremos modificar el límite de caracteres, si lo dejamos interactuable o estará bloqueado para el jugador, y modificar el texto de ejemplo que aparece en la zona de escritura.

```
[SerializeField] public TMP_InputField inf;
```

```
inf.characterLimit = 10;
inf.interactable = true;
inf.text = "Escribeme los versos más tristes esta noche.";
```

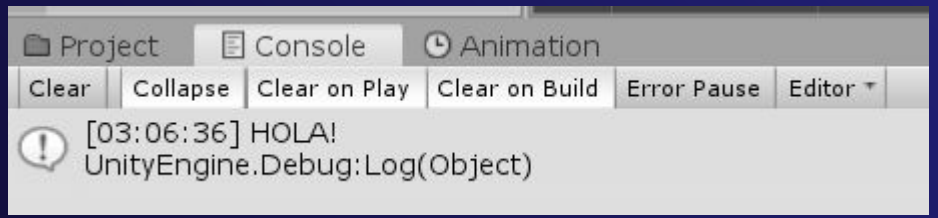


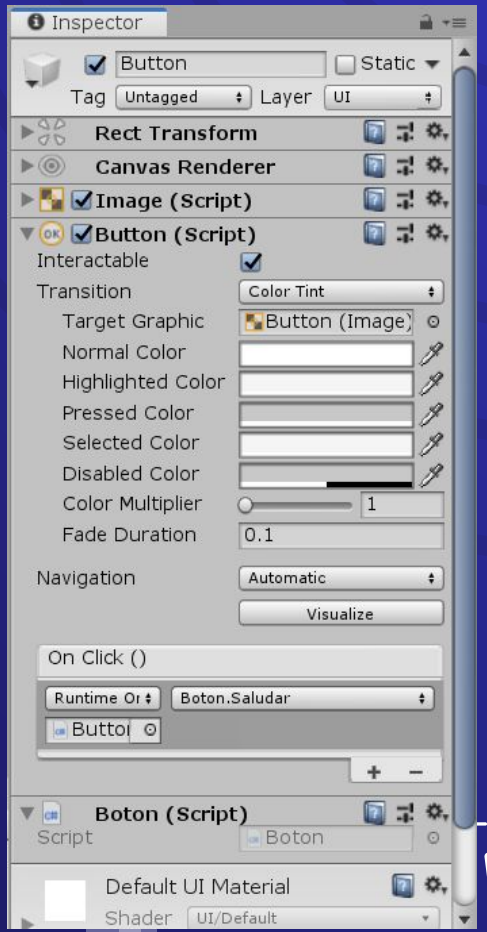
Ejercicio 2

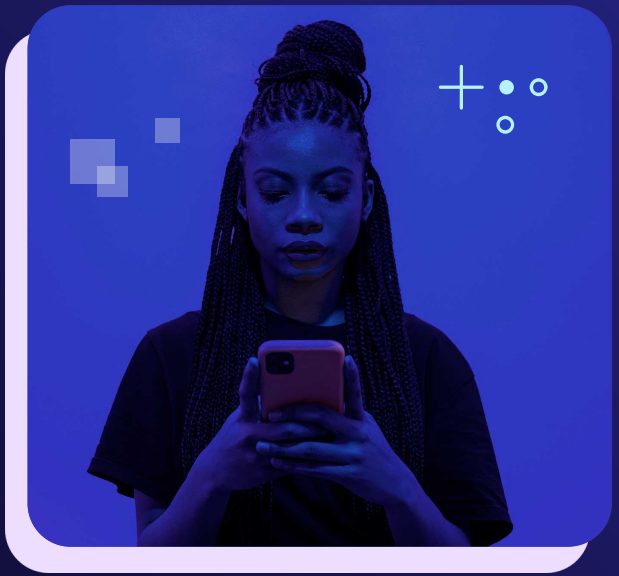
Crear una nueva escena y añadir un botón. Luego debe crear un nuevo Script llamado “Boton”, en él, desarrolle una función que envíe un mensaje por consola “Hola!”. Arrastre el script al objeto botón creado y configurar su componente Button. Dar play para visualizar los resultados.

Ejercicio 2

```
public class Boton : MonoBehaviour
{
    public void Saludar()
    {
        Debug.Log("HOLA!");
    }
}
```







Ejercicio 3

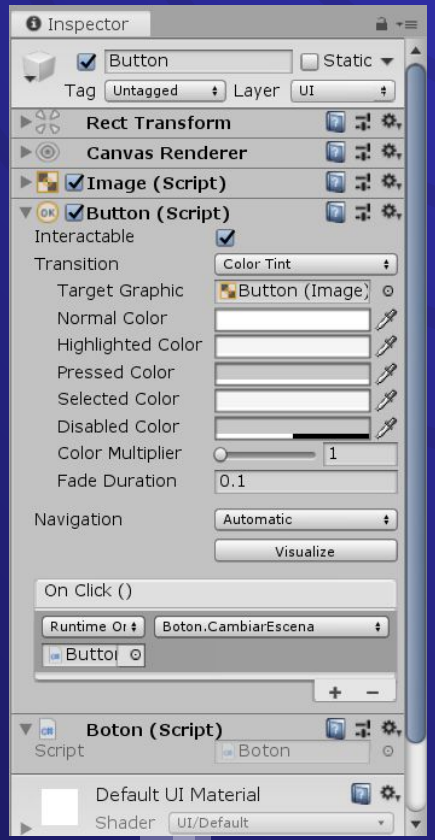
Usando el ejercicio anterior, agregue una nueva función que cambie de escena al presionar el botón. Dar play para visualizar los resultados.

Ejercicio 3

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Boton : MonoBehaviour
{
    public void Saludar()
    {
        Debug.Log("HOLA!");
    }

    public void CambiarEscena()
    {
        SceneManager.LoadScene(1);
    }
}
```



User Interface

TextMesh Pro

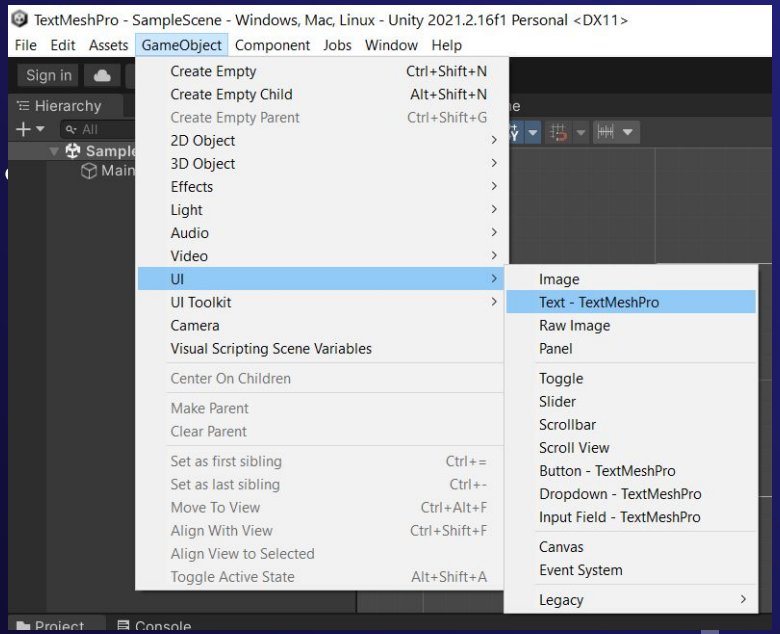
Es la **versión ultra mejorada** del componente Text (Legacy) de Unity, potente y fácil de usar.

TextMesh Pro utiliza técnicas avanzadas de representación de texto junto con un conjunto de shaders personalizados; ofrece mejoras sustanciales en la calidad visual al tiempo que brinda a los usuarios una flexibilidad increíble en lo que respecta al estilo y la textura del texto.

TextMesh Pro proporciona un **control mejorado sobre el formato y el diseño del texto** con características como: espaciado entre caracteres, palabras, líneas y párrafos, kerning, texto justificado, enlaces, más de 30 etiquetas de texto enriquecido disponibles, soporte para fuentes múltiples y sprites, estilos personalizados y mucho más.

Gran rendimiento, debido a que la geometría creada por TextMesh Pro usa **dos triángulos por carácter** al igual que los componentes de texto de Unity. Esta calidad visual y flexibilidad mejoradas **no tienen costo de rendimiento adicional**.

Scene Manager



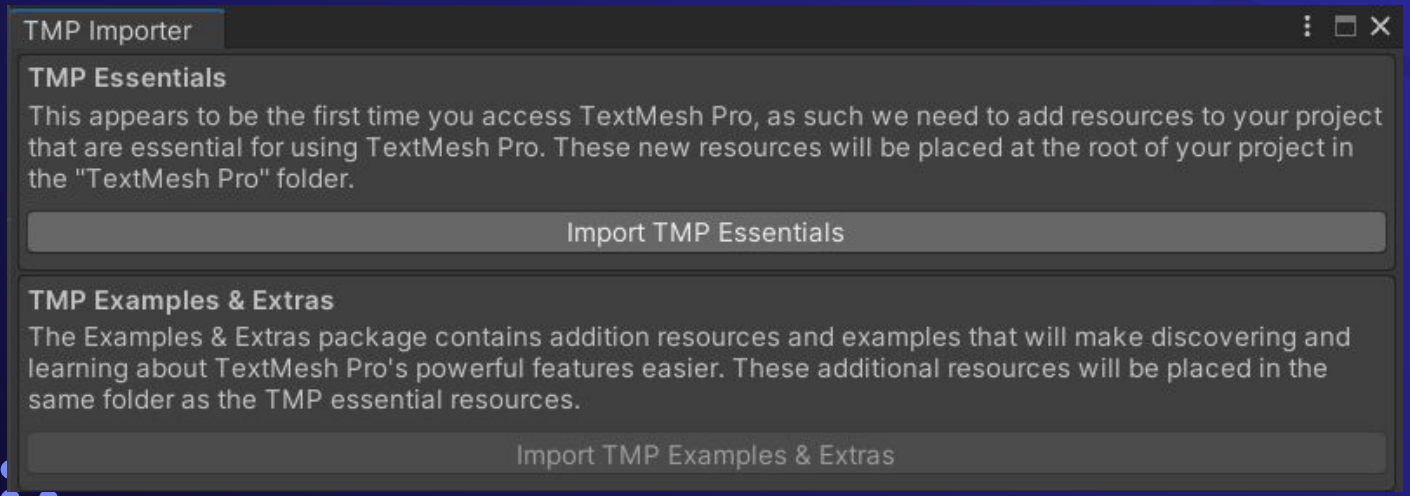
Para poder crear un objeto del tipo **Text de TextMesh Pro**, dirigirse al menú superior y seleccionar **GameObject**.

Navegar al sub menú **UI** y luego seleccionar **Text - TextMeshPro**.

Si no hay un Canvas en la escena, Unity creará uno y posicionará el elemento **Text** como hijo en la jerarquía.

User Interface

Si es la **primera vez** que creas un objeto TextMesh Pro en el proyecto, aparecerá una ventana emergente llamada **TMP Importer**. Se debe seleccionar la primera opción **Import TMP Essentials** y en la consola aparecerá un mensaje confirmando la importación. Con esto ya se podrán utilizar elementos TextMesh Pro **en ese proyecto**.



User Interface

Una vez creados cualquiera de estos elementos de UI, el componente **Transform** de ese objeto es reemplazado por otro llamado **Rect Transform**.

El **Rect Transform** es un componente muy importante en Unity, es utilizado para **controlar la posición, el tamaño y la escala de un elemento en la interfaz de usuario**, como botones, paneles, imágenes y texto.

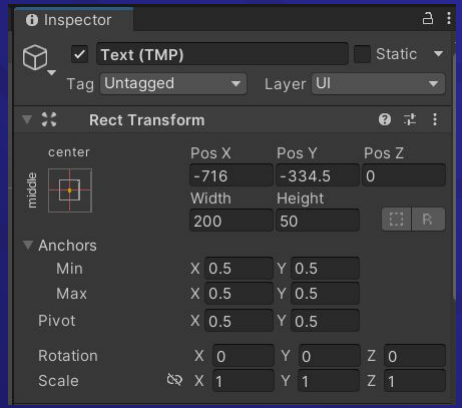
El RectTransform define la posición del objeto en relación con su objeto padre y la pantalla.

Se utiliza un sistema de coordenadas en el que el punto (0, 0) representa la **esquina inferior izquierda de la pantalla** o del rectángulo padre.

El RectTransform permite establecer el tamaño del objeto en términos de **ancho y alto**.

También puedes ajustar la escala del objeto, lo que cambia su tamaño sin afectar su posición o anclaje.

El tamaño y la escala se pueden ajustar tanto de forma absoluta como relativa utilizando porcentajes o valores proporcionales.



User Interface

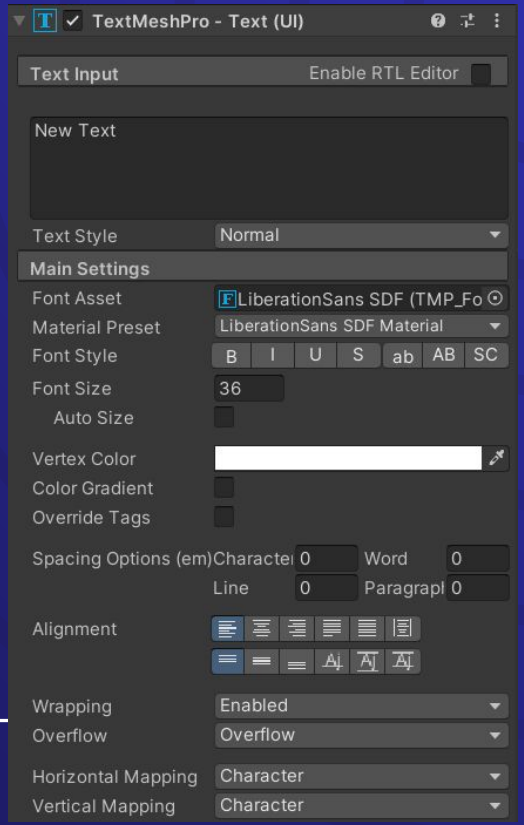
Este componente posee los siguientes valores modificables en el inspector:

- **Text Style**: Determina el estilo del texto (título, subtítulo, etc.).

- **Font Style**: Selección del estilo de la fuente (negrita, cursiva, subrayado, etc.).

- **Font Size**: Corresponde al tamaño del texto.

- **Alignment**: Opciones para alinear el texto a su contenedor (alineado vertical y/u horizontal).



User Interface

Para utilizar un TextMesh Pro desde scripts, primero se debe **agregar la librería TMPro**. Luego, se declara una variable del tipo de dato **TextMeshProUGUI**.

Para cambiar el texto de un TextMesh Pro, se utiliza su **propiedad text** y se le asigna un valor de texto.

Si desea usar una variable de otro tipo que no sea string, puede utilizar la función **ToString()** para intentar convertir su valor a texto.

Por ejemplo, un número entero en texto alfanumérico.

```
int numerito = 3;
tmp_saludo.text = numerito;
tmp_saludo.text = numerito.ToString();
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

Unity Script | 0 references
public class TMPScript : MonoBehaviour
{
    public TextMeshProUGUI tmp_saludo;

    Unity Message | 0 references
    void Start()
    {
        tmp_saludo.text = "HOLA";
    }
}
```

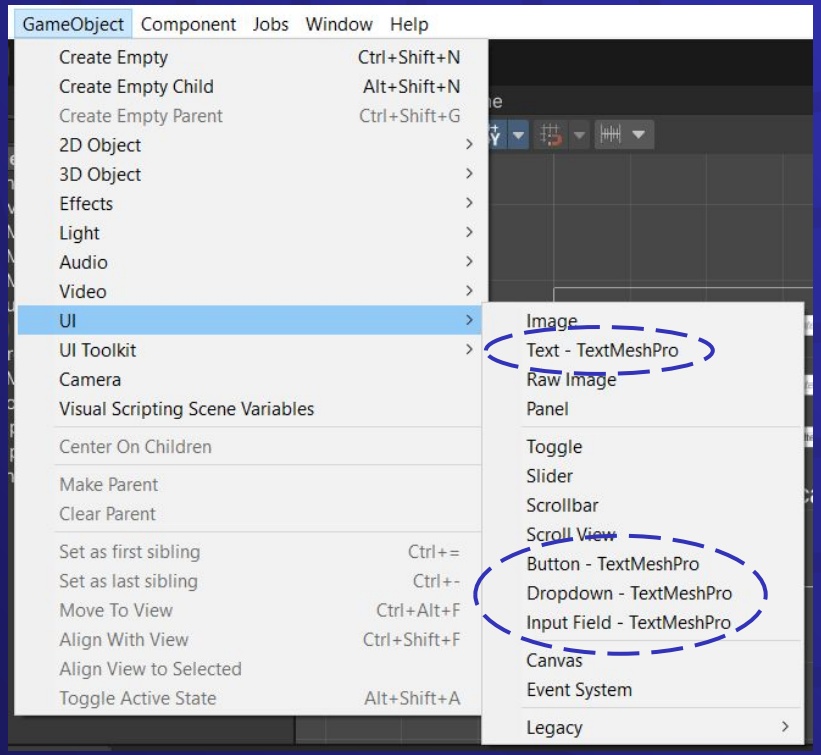
User Interface

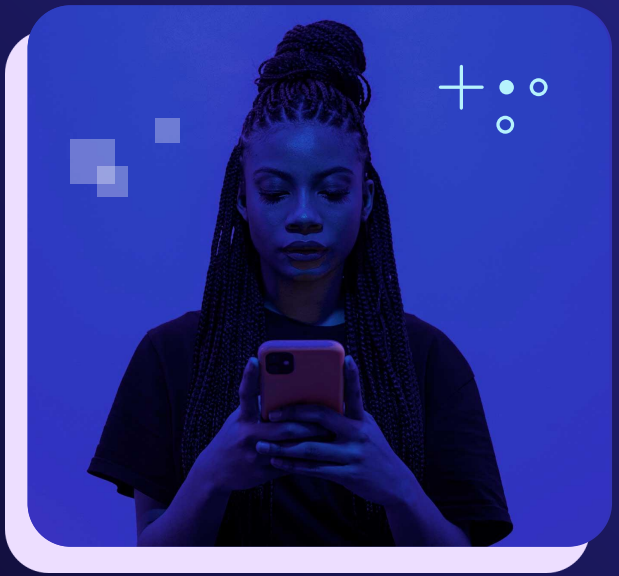
TextMesh Pro también cuenta con componentes de interacción:

- **Button**
- **Dropdown**
- **Input Field**

Para trabajar tipos de datos de **TextMesh Pro** para el Text, Input Field y Dropdown List tenemos que trabajar las variables de la siguiente forma:

```
TextMeshProUGUI text;
TMP_InputField field;
TMP_Dropdown drop;
```

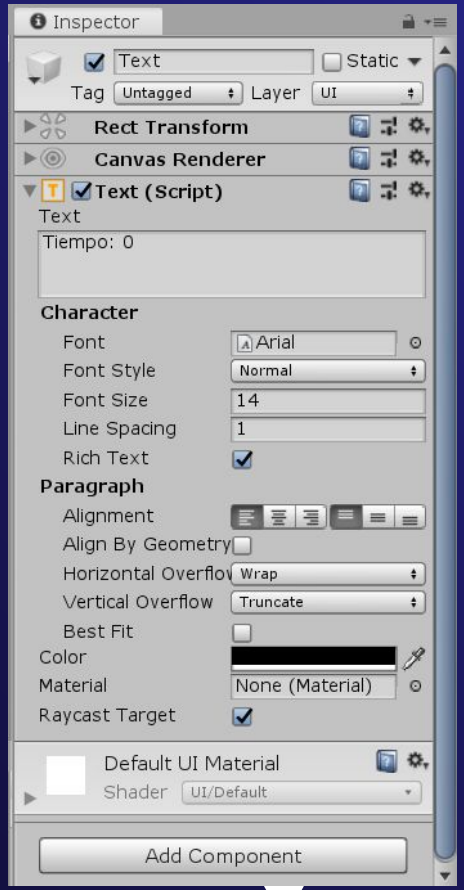




Ejercicio 4

Agregar un TextMesh Pro en la UI. Desarrolle un script que vaya contando el tiempo de juego. Este valor debe ser mostrado por el texto en la UI. Dar play para ver el comportamiento.

Ejercicio 4




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

public class Texto : MonoBehaviour
{
    public float timer = 0;
    Text timerTxt;

    void Start()
    {
        timerTxt = GetComponent<Text>();
    }

    void Update()
    {
        timer += Time.deltaTime;
        timerTxt.text = "Tiempo: " + timer.ToString("000");
    }
}
```

Introducción a la programación de videojuegos

Sebastián Díaz Arancibia

Universidad Técnica Federico Santa María - 2023