«Talento Tech»

Desarrollo de Videojuegos

Unity 2D

Clase 11

«Talento Tech»

Buenos Aires Ciudad

Clase N° 11 | Interfaz

Temario:

- Diseño de interfaces (UI)

¿Qué es una interfaz de usuario?

La interfaz de usuario (UI por sus siglas en inglés, "User Interface") es el espacio donde se producen las interacciones entre nosotros/as y el juego. Su objetivo es permitir el funcionamiento y control de las funciones del juego, además de posibilitar la extracción de información acerca del estado y de variables que transcurren durante la experiencia lúdica.

Una interfaz puede adoptar diversas formas, como un menú, una barra de vida, un inventario, un mapa, un diálogo escrito, entre otras. Existen infinitos tipos de interfaces, y cada juego proporcionará distintas formas de comunicar lo que sucede. En algunos casos, los juegos pueden incluso consistir principalmente en elementos de interfaz.

Es importante comprender que, en un videojuego, la UI es, en un 95% de los casos, una capa de elementos que se superpone en la cámara. Es decir, son elementos 2D que siguen la cámara y que no se encuentran dentro del mundo del juego; son **no diegéticos**.

Las UI **diegéticas** son menos comunes, y son interfaces que se encuentran dentro del mundo donde se desarrolla el gameplay. Un ejemplo muy popular puede ser la interfaz del Dead Space, donde la barra de vida del protagonista se encuentra incrustada dentro de la lógica que nos propone el juego.

UI Diegética	IU No Diegética
 **Dead Space**	 **Valorant**
 **MechWarrior**	 **World of Warcraft**

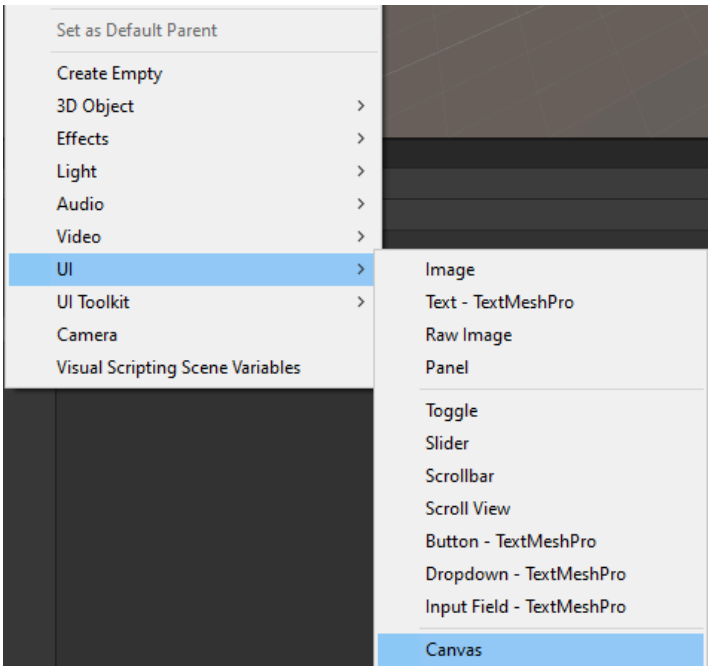
Siguiendo el ejemplo de arriba, las UI no diegéticas son las más comunes y son las que vamos a trabajar en esta clase.

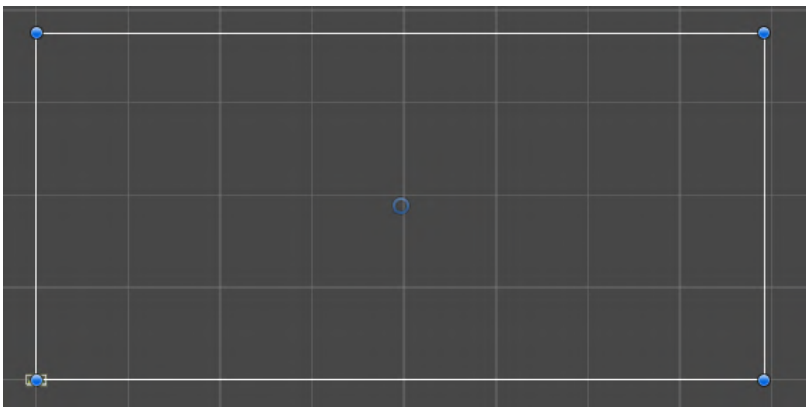
Creando el canvas.

El *Canvas* en Unity es un componente fundamental utilizado en el desarrollo de interfaces de usuario, todo lo que hagamos que tenga que ver con UI, tiene que ir dentro de este componente, va a ser nuestro lienzo. El *Canvas* actúa como un contenedor que alberga diferentes elementos de la interfaz, como botones, textos, imágenes y paneles, que son visibles en la pantalla del juego. Es esencial para crear y controlar la presentación y la interacción de los elementos de la interfaz dentro del mundo del

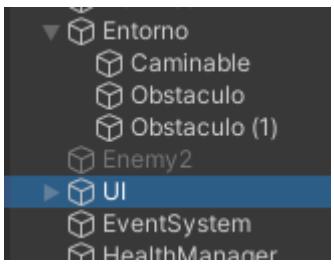
juego.

Lo que tenemos que hacer para **crear nuestro Canvas,** es dar click derecho sobre el *Hierarchy* y buscar el apartado de UI y dentro del mismo damos click donde dice Canvas.



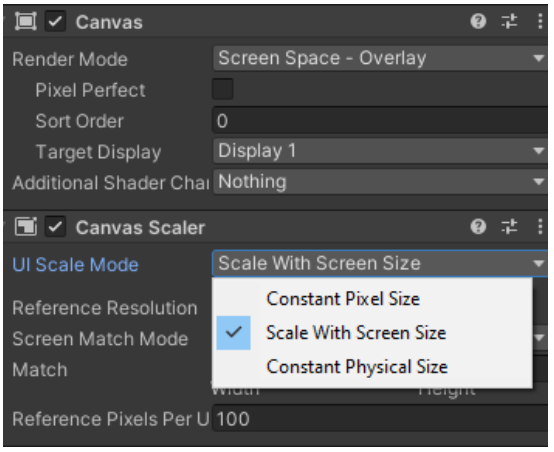


Automáticamente se nos generará una pantalla en 2D a la cual podremos agregarle varios gameobject como ser, Textos, botones, Toggle, entre otros. Cuando creamos un *Canvas* por primera vez, también se genera un objeto llamado *EventSystem*, que hace que los elementos de la interfaz sean interactuables y cumple otras funciones más avanzadas.



Por ahora vamos a arrastrar dentro de nuestro *Canvas* por una cuestión de orden, y vamos a renombrar al *Canvas* como "UI" por la misma razón.

Antes de continuar, tenemos que cambiar una variable del *Canvas*. Vamos a seleccionarlo, e iremos a ***UI Scale Mode*** en el componente ***Canvas Scaler***. Pasaremos de ***Constant Pixel Size*** a ***Scale With Screen Size***. Con esto lograremos que la interfaz se adapte al tamaño de la pantalla.

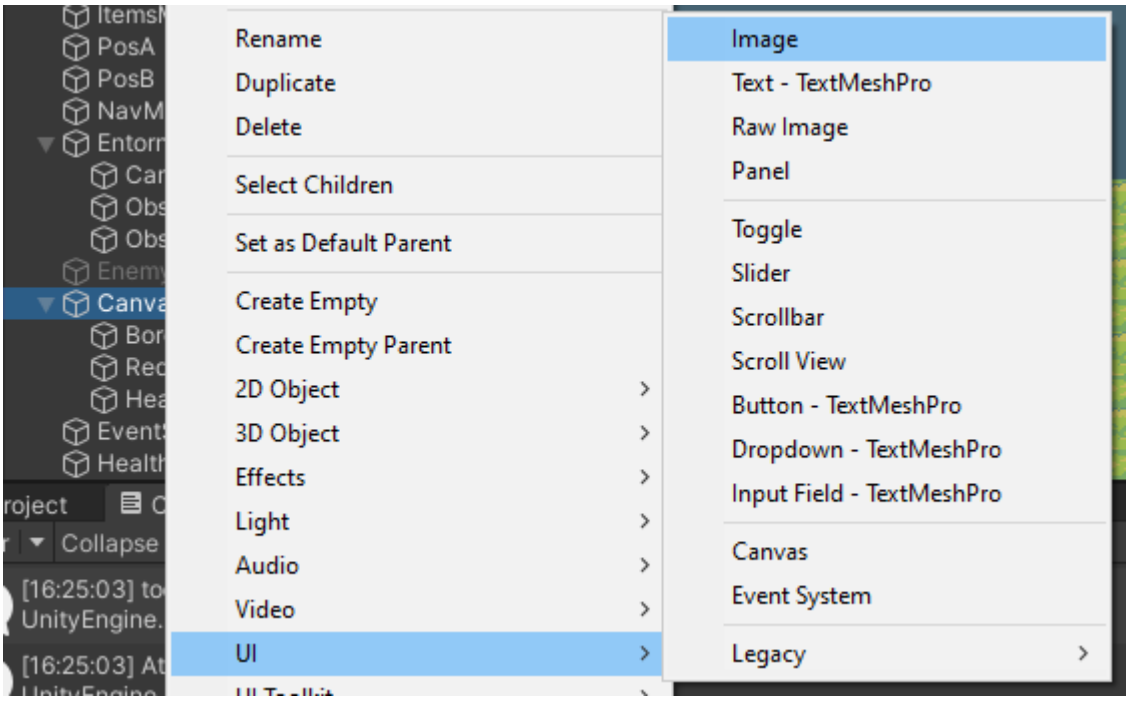


Barra de vida.

Casi todos los juegos tienen un sistema de vida, entre corazones, caras y la más común, la barra de vida. Vamos a hacer una barra sencilla para poder usar en nuestro juego.

Armado.

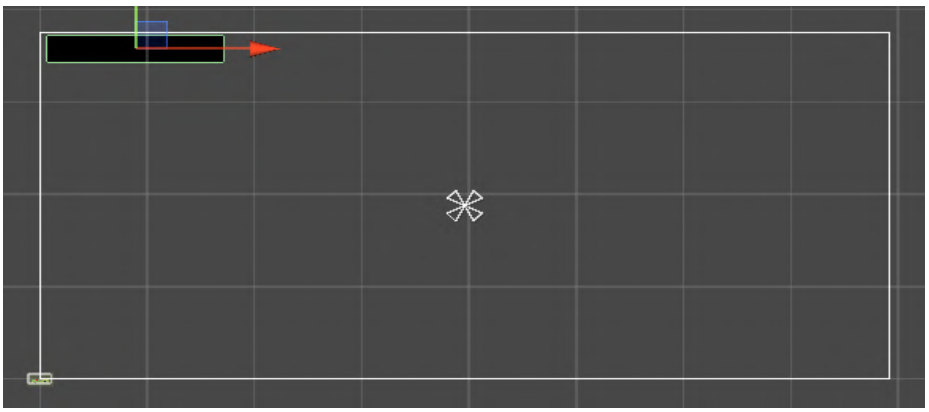
Primero crearemos 3 objetos dentro del Canva, que serán 3 Imágenes. Para eso vamos a hacer click derecho en la jerarquía, sobre el canva, iremos a UI y crearemos la primera:



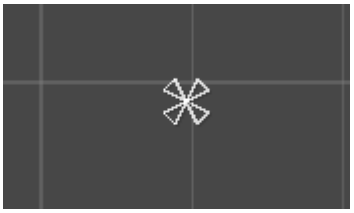
Tengan cuidado, que al ser parte del canva, se creará en el centro del mismo:

Lo que está marcado en rojo es el nivel que estábamos armando, vean que hay que retroceder bastante para tener un panorama del Canva. Arriba a la derecha, en lo que sería el centro del Canvas, está la imagen creada.

Seguiremos cambiando el color a negro, le daremos forma rectangular y la acomodaremos arriba de todo a la izquierda, del Canvas.

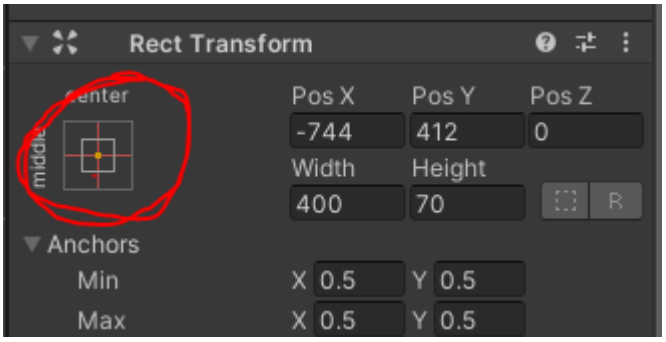


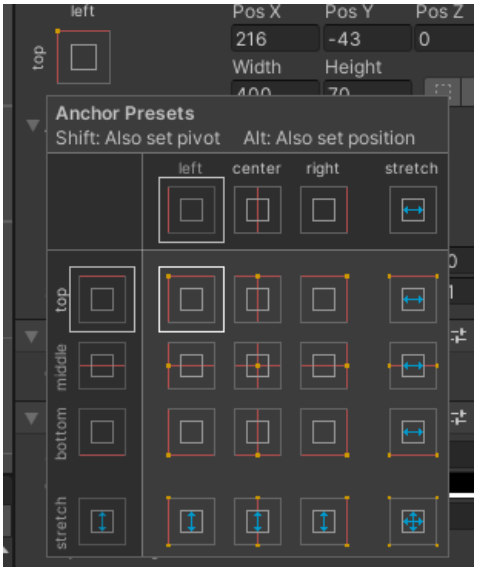
Si prestan atención al centro, veran una “cruz”, ese es el Anchor:



Este “Ancla” ayuda a posicionar a nuestro objeto, haciendo que, si la pantalla cambia mucho su tamaño, el objeto se aproxime cerca del mismo Anchor.

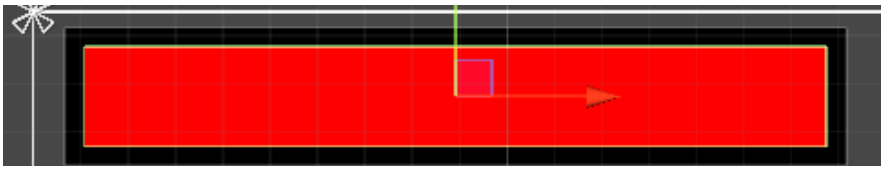
En este caso está en el medio, pero si queremos posicionar nuestra Barra de vida, (arriba de todo a la izquierda) debemos modificar su ubicación. Para eso, teniendo seleccionado nuestro rectangulo negro, iremos al inspector y haremos click aqui:



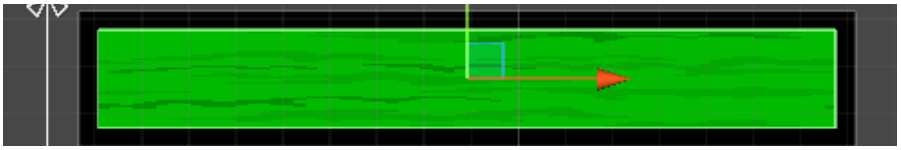


Nos aparecerán opciones para colocar el Anchor y elegiremos la que más nos sirva.

Ahora que sabemos lo del Anchor y ya tenemos una imagen. Duplicaremos la que ya hicimos, le daremos el color rojo, y la haremos un poco más chica, dejando a la parte negra como “contorno”.

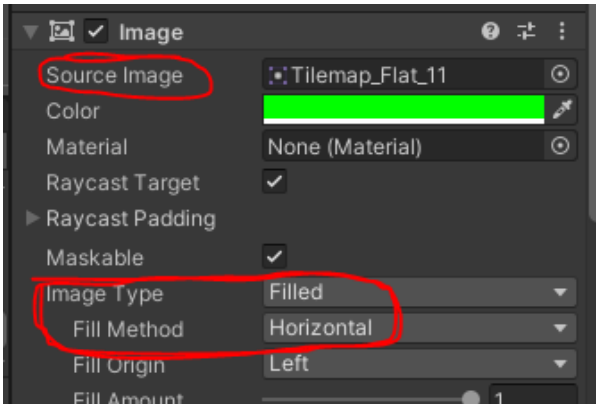


La volveremos a duplicar y le pondremos un color verde.



Antes de ir con el código, tendremos que cambiar el “**Type**” de nuestra imagen para que funcione con “**Fill**”, tipo “relleno”. Esto será para poder programarla más tarde.

Necesitaremos ir al component de Image y reemplazar el “Source Image” por alguna imagen/Sprite a gusto. Una vez hecho esto, cambiará la textura de la barra y aparecerán “**Image Type**” y “**Fill Method**”

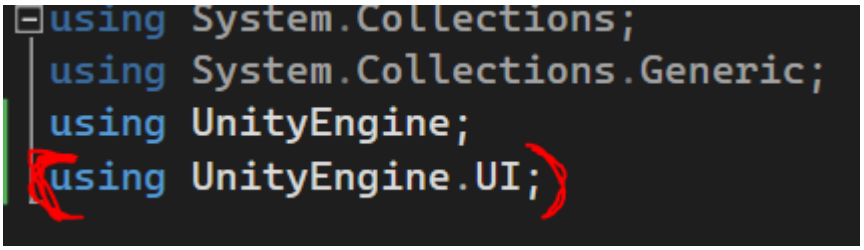


Tendremos que poner en “Image Type: Filled” y “Fill Method: Horizontal”

Ahora si, ¡podemos pasar al código!

Código.

En el Script de nuestro personaje, agregaremos la librería “UnityEngine.UI” que nos permitirá trabajar con gran parte de lo que es la interfaz.



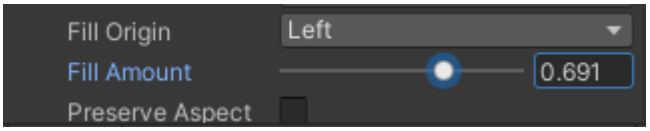
Crearemos una variable para referir a la barra de vida, y obviamente si no la tenemos todavía, otra variable para nuestra Vida en sí.

```
    [SerializeField]
    private float vida;
    public Image healthBar;
```

Asegurense que la variable de vida sea **float**, para evitar conflictos con la **HealthBar** más adelante.

```
    void Start()
    {
        healthBar.fillAmount = (vida / 100);
    }
```

¿Por qué la dividimos por 100? Porque el “relleno” de la HealthBar, llega de 0 a 1, entonces, tenemos que hablar en términos de **decimales**, para poder trabajar con ella.



Una vez hecho esto, basta con implementar la misma linea de codigo en las situaciones en las que recibo daño o me curo. Por ejemplo esta función para curarme:

```
    public void Heal(float a)
{
    vida += a;
    healthBar.fillAmount = (vida/100);
}
```

(La función Damage (de daño), seria igual, solo que con un negativo (-=))
Verán que recibe un parámetro para que cuando sea llamada, nos diga cuánto nos curamos y siempre estamos igualando a la HealthBar con el dato de (vida/100).

Contador de ítems.

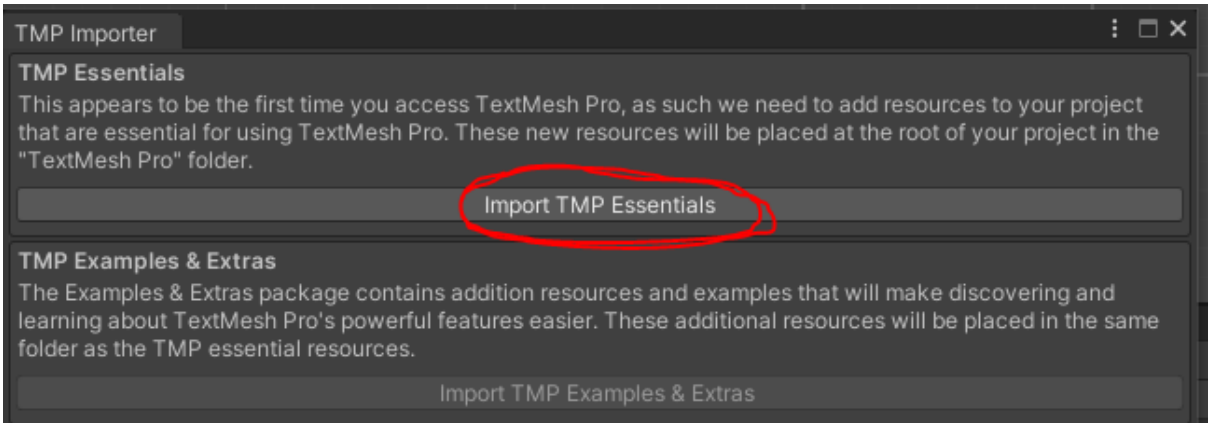
Por último en la clase de hoy, haremos una interfaz sencilla que deberá contar algún ítem que esté juntando.

Armado.

Para eso, empezaremos a colocar una imagen del “objeto a reunir” en alguna parte de nuestra UI.

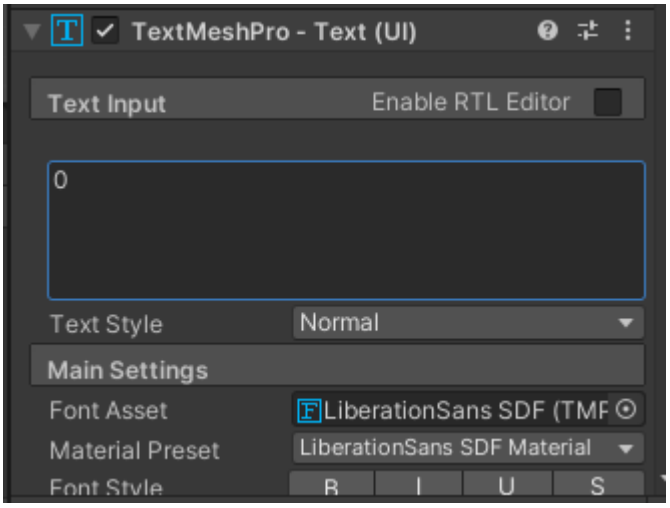
Recuerden que debe hacerlo yendo a **UI -> Image**

Seguiremos en la jerarquía, dentro del Canvas y haremos **Click Derecho -> UI -> TextMeshPro**
Nos aparecerá una ventana para importar algunas cosas y le daremos al boton de “**Import TMP Essentials**”



Cuando terminemos de exportar, nos habrá aparecido el objeto creado.

Lo acomodaremos al lado de la imagen, tal cual hicimos con el resto. y Si vamos al inspector, notaremos el component “Text Mesh Pro - Text (UI)”, que nos dejará cambiar el texto inicial del GameObject, junto con las propiedades típicas de un Editor de Texto:



Código.

Dentro del código en el que hayamos creado la obtención del Objeto, sea el Script ItemManager, Inventario u otro. Pasaremos a añadir la librería del TMPro y crear la variable de referencia.

```
using TMPro;
 public class ItemManager : MonoBehaviour {
    public TextMeshProUGUI contadorHuesos;
```

Noten el tipo de variable “TextMeshProUGUI”

Iremos al sector donde aumenta/cuenta los huesos que obtengo y agregaremos el siguiente código:

```
contadorHuesos.text = huesos;
```

O, Si tienen el “Inventario” de la clase 8, tendrán que ir al Script “**Item**”, en la función llamada en el **OntriggerEnter2D** que permite el reconocimiento del “**objID**” y obtención del objeto, y tendrán que poner esto:

```
if (col.CompareTag("Player")) {
        ItemManager inv = col.GetComponent<ItemManager>();
   foreach (KeyValuePair<GameObject, int> items in inv.inventory) {
         Item invID = items.Key.GetComponent<Item>();
        if (invID.objID == objID) {
            Debug.Log("Obtuve un: " + gameObject.tag);
            inv.inventory[items.Key] += myCount;
   if (CompareTag("Bone")) {
      inv.contadorHuesos.text = inv.inventory[items.Key].ToString();
               }
            Destroy(gameObject, 0.2f);
            break;
         }
```

Diciendo que “Si soy un hueso, entonces agarra la propiedad “contadorHuesos.text” de mi variable “inv”(Referencia a ItemManager) y pasale el **Value** actual de esta **Key.**

Finalmente, esto nos permitirá mostrar la cantidad de elementos adquiridos.

Ejercicios prácticos:

Realizar la interfaz de cualquier elemento deseado. Algunas sugerencias:

- **Poner la cara de nuestro personaje**. Le podemos agregar varias expresiones de dolor o alegría dependiendo la vida y la cantidad de ítems.

- **Crear diferentes estilos de la HealthBar.** Nosotros hicimos una sencilla, pero como verán pueden usar cualquier imagen de base. ¡Sean creativos!
- **Poner Imágenes con las habilidades a usar.**
- ¡**Hasta podrías agregar las Misiones de tu personaje**!

Buenos Aires aprende

Agencia de Habilidades para el Futuro

Buenos Aires Ciudad