**INTERFAZ DE USUARIO**

**1 - Rect Transform**

Para empezar a trabajar en interfaz de usuario necesitamos como primera cosa crear un Canvas. Un Canvas es un componente. Se puede crear uno haciendo click secundario en la jerarquía y seleccionando UI -> Canvas. Todos los elementos de interfaz de usuario (o UI, por sus siglas en inglés) existirán como hijos de algún canvas.

Cuando creamos un objeto cualquiera como hijo de un objeto Canvas, este tendrá un componente RectTransform, que **reemplazará** al componente Transform típico.

El RectTransform es necesario para tratar con interfaces de usuario, porque, a diferencia de un game object común, un objeto de UI no ocupa solo un punto en el espacio, sino que un área en la pantalla. El área que utilizará este objeto, además, es variable. Esto quiere decir, por ejemplo, que un panel de texto debe ser más o menos amplio dependiendo del tamaño de la pantalla donde estemos jugando.

**1.1 - Anclajes**

Los objetos de UI tienen anclajes. Cada RectTransform está anclado en uno o más puntos con su padre RectTransform.

Un RectTransform puede visualizarse como un rectángulo. Cada vértice de este rectángulo está asociado a uno de 4 anclajes.

Las siguientes dos afirmaciones contienen todo lo que hay que saber acerca de los anclajes. Se describen primero brevemente y luego se explican:

1 - Los anclajes se ubican **en posiciones relativas** al padre del RectTransform.

2 - La distancia que hay entre cada vértice y su punto de anclaje **se mantiene siempre constante**.

Partamos explicando el punto 1. Cuando hablamos de posiciones **relativas**, a diferencia de posiciones absolutas, es que estamos hablando de porcentajes, no de distancias concretas. Imaginemos que tenemos una línea de 100 unidades de largo. Si queremos ubicar un punto en la mitad de la línea tengo dos maneras de hacerlo. La manera absoluta de hacerlo es ubicando el punto a 50 unidades de distancia de un extremo. La otra manera es diciendo que el punto se ubica a un 50% del tamaño de la línea. Puede parecer que ambas aproximaciones tienen el mismo resultado, pero si por alguna razón el largo de la línea cambia a 200 unidades de largo, el método absoluto nos deja ubicado el punto a ¼ del largo de la línea, mientras que el método relativo nos asegura que el punto seguirá estando al medio.

Lo descrito anteriormente es muy importante en UI, porque todo se trata de que nuestra interfaz de usuario se adapte a distintas resoluciones y aspectos de pantalla. No podemos usar posiciones absolutas.

Entonces, para terminar el punto 1: Cada anclaje estará ubicado en una posición relativa dentro del rectángulo padre. Si el rectángulo padre cambia de forma, el anclaje se moverá apropiadamente.

Una vez el rectángulo padre haya tomado una forma concreta de acuerdo a la pantalla donde está corriendo nuestro juego, tenemos que poder definir cuáles serán las dimensiones concretas de nuestro rectángulo. Llegamos al punto 2.

A partir de cada uno de los anclajes dibujamos uno de nuestros vértices a una distancia definida de este. Esta **no es** una distancia relativa, sino absoluta. Si un vértice está definido para estar 10 pixeles bajo su anclaje y 10 pixeles a la derecha de este, esto siempre será así.

**1.2 - Left, Top, Bottom, Right**

Según como hayamos ubicado nuestros anclajes, en nuestro componente RectTransform aparecerán distintas propiedades. Estas propiedades varían. No siempre serán las mismas.

Lo que describen estas propiedades es el tamaño y posición de nuestro rectángulo. Dado que la forma de este puede variar según los anclajes y las dimensiones del rectángulo padre, no siempre se describe nuestro rectángulo utilizando las mismas propiedades. Por ejemplo, no puedo describir un rectángulo como que tiene 100 pixeles de ancho si esto depende del ancho de mi rectángulo padre.

Sin embargo, una vez haya decidido mis anclajes, hay ciertas propiedades que puedo definir que **siempre se mantendrán constantes**. Estas son las que aparecen en el inspector.

Si, por ejemplo, tengo anclado mi objeto a las cuatro esquinas de su rectángulo padre, no puedo describir mi rectángulo con un ancho y alto fijo, pero sí puedo describirlo como un rectángulo que mantiene **márgenes fijos**. Si ese es el caso, los valores que aparecerán en el inspector serán Top, Bottom, Right y Left, y representan los márgenes que mi rectángulo siempre mantendrá con su padre.

Si, por el contrario, mi objeto tiene todos sus anclajes en la esquina superior izquierda de su padre, ya no puedo hablar de márgenes. Lo que se mantiene constante en este caso es un ancho y alto fijo, y una distancia con respecto al punto único de anclaje. En este caso las propiedades PosX, PosY, Width y Height aparecerán en el inspector.

Entonces, según los anclajes que tengamos veremos en el componente RectTransform alguna permutación de las propiedades descritas, y las propiedades que veremos serán siempre las cantidades que se mantienen constantes según la configuración de anclajes que tengamos.

**2 - Componentes de uso común**

Hay muchos elementos que tenemos que ser capaces de crear con UI. Los ejemplos más básicos (y útiles) son las imágenes y el texto, pero también podemos querer crear botones y sliders, por ejemplo. Cada uno de estos elementos está representado con un componente en Unity. A continuación se describen algunos de los más importantes:

**2.1 - Image**

Las propiedades más relevantes de este componente son las siguientes:

**Source Image:** Este es el sprite que aparecerá como imágen.

**Image Type:** En UI tenemos varias maneras de mostrar imágenes. Podemos querer mostrarla como un simple sprite o podemos querer agrandarla y deformarla, como un panel. esta propiedad define cómo se verá de acuerdo con las dimensiones de mi RectTransfom.

**2.2 - Text**

Las propiedades más relevantes de este componente son las siguientes:

**Text:** Este es el texto que se mostrará dentro de las dimensiones de mi rect transform.

**Character:** Este conjunto de propiedades definen la tipografía, el espaciado y el tamaño del texto. Son similares a la mayoría de los software de edición de texto.

**Horizontal y Vertical Overflow:** Estas propiedades definen qué es lo que ocurrirá si mi texto no cabe horizontal o verticalmente dentro de las dimensiones de mi RectTransform. Las opciones son básicamente o bien cortar o bien dejar sangrar el texto más allá de los límites de mi rectángulo.

**Best Fit:** Este booleano define si el texto intentará acomodarse a mi rectángulo de manera dinámica o no. Si esta propiedad es seleccionada el tamaño del texto cambiará para intentar cubrir por completo el espacio que tengo.

**2.3 - Button**

Este componente, al ser añadido a un objeto con un RectTransform, lo transforma en un botón con el que mi usuario puede interactuar. Hay dos cosas principales que debemos configurar cuando creamos un botón: las transiciones y los eventos. Las transiciones definen cómo cambiará el botón al estar el mouse fuera, sobre y apretando el botón. Los eventos definen lo que ocurrirá cuando el botón sea apretado. Los eventos se ven en la siguiente sección de este documento.

Las transiciones de un botón pueden ser de varios tipos. El botón puede simplemente cambiar de color al ser utilizado o puede cambiar su imágen por completo.

**2.3 - Slider**

Un slider es una manilla que se mueve sobre un riel, y que sirve para ajustar alguna propiedad numérica, como el volumen y la luminosidad del juego.

Si creamos un objeto slider (Create -> UI -> Slider) veremos que se crea toda una jerarquía de objetos. esto es porque un slider es un objeto compuesto. Está la manilla, el riel y el color de relleno.

En el componente Slider tenemos, al igual que en el componente Button, parámetros de transición y una sección de eventos. Podemos fijarnos también que tiene propiedades llamadas FillRect y HandleRect. Los objetos que le asignemos a estas variables representarán el relleno de la barra y la manilla de esta.

Muchos componentes de interfaz de usuario tienen propiedades como las recién descritas, que guardan referencias a las partes que lo componen y son necesarias para que funcionen correctamente.

**3 - Eventos**

Todos los elementos interactivos de UI, como los botones y los sliders, tienen una sección de eventos dentro de su componente.

Esta sección es una lista con todas las cosas que van a ocurrir cuando el jugador interactúe con este elemento. Un botón puede equipar un ítem o pausar el juego, un slider puede controlar la fuerza de un golpe en un juego de golf.

Para añadir un evento a la lista primero hay que apretar el botón de ‘+’. Cada elemento que será añadido a la lista representa **una función** que será ejecutada cuando se cumpla el evento.

Para añadir una función a un evento primero hay que arrastrar al campo de object **el objeto que contiene el script que contiene la función**. Suena enredado, recomiendo releer la frase en negrita hasta que les haga sentido.

Una vez asignamos un objeto en este campo podemos seleccionar en un dropdown la función que queremos que se ejecute. Esta función **debe ser pública y retornar void**.

Vamos a ver un ejemplo con el evento de un botón y uno con el slider. Eso debería dar una buena idea de cómo funcionan todos los eventos de UI.

Imaginemos que queremos tener un botón que pause el juego. Primero creamos en un script una función como la que sigue:

public void PauseGame ()

{

Time.timeScale = 0;

}

Ahora le añadimos este script a un GameObject en nuestra jerarquía.

Luego, en la sección eventos de nuestro componente botón apretamos ‘+’ y arrastramos el GameObject con el script al campo “object”. Luego en el dropdown podemos buscar la función PauseGame. Al apretar el botón el juego quedará en pausa.

El caso de un slider es similar, con la diferencia de que la función debe aceptar un número como parámetro. Dado que dependiendo de la posición de la manilla del slider podemos querer hacer cosas distintas, necesitamos un número que represente el valor (posición) actual del slider.

Los pasos a seguir son los mismos que en el botón pero nuestra función sería la siguiente si, por ejemplo, quisiéramos controlar el volumen del juego:

public void SetVolume (float newVolume)

{

AudioListener.volume = newVolume;

}