Joel Esteban Peña Pinzón – Andrés Juan Giraldo Vargas

1.

Tanto uGUI como UI Toolkit crean y mantienen la interfaz de usuario dentro de una estructura de árbol jerárquico. En uGUI, todos los elementos de esta jerarquía son visibles como GameObjects individuales. en el panel de vista de jerarquía. En UI Toolkit, elementos visuales organizar en un árbol visual. El árbol visual no es visible en el panel.

El componente Canvas en uGUI es similar al componente UIDocument en UI Toolkit. Ambos son MonoBehaviours que se adjuntan a GameObjects.

En uGUI, un componente Canvas se encuentra en la raíz del árbol UI. Funciona con el componente Canvas Scaler para determinar el orden de clasificación, la representación y el modo de escala de la "UI" que se encuentra debajo.

En UI Toolkit, el componente UIDocument contiene una referencia a un objeto PanelSettings. PanelSettings contiene la configuración de representación para la interfaz de usuario, incluido el modo de escala y el orden de clasificación. Varios componentes UIDocument pueden apuntar al mismo objeto PanelSettings, lo que optimiza el rendimiento cuando se utilizan varias pantallas de UI en la misma escena.

UI Toolkit se refiere a los elementos de la UI como controles o elementos visuales. Ejemplos de elementos de UI son: Controls, Buttons, Text labels. uGUI construye la jerarquía de la interfaz de usuario a partir de GameObjects. Agregar nuevos elementos de UI requiere agregar nuevos GameObjects a la jerarquía. Los controles individuales se implementan como componentes MonoBehaviour. En UI Toolkit, el árbol visual es virtual y no utiliza GameObjects. Ya no puedes crear ni ver la jerarquía UI en la vista de jerarquía, pero elimina la sobrecarga de usar un GameObject para cada elemento de la UI.

En uGUI, los elementos UI derivan (directa o indirectamente) de la clase base UIBehavior. De manera similar, en UI Toolkit todos los elementos de UI derivan de una clase base llamada VisualElement. La diferencia clave es que la clase VisualElement no deriva de MonoBehaviour. No puedes adjuntar elementos visuales a GameObjects. Trabajar con controles de UI Toolkit en script es similar a trabajar con controles uGUI

Así, podemos ver que el Ui toolKit nos presenta una mejor manera optimizar el trabajo para la interfaz, manejando un estilo propio de ventanas de trabajo, y controlando las capas de mejor manera, pero a costar de no tener un espacio en la escena para el UI, además, a diferencia del uGUI, el ToolKit carece de un buen sistema de animation controller, además de que aun posee de algunas limitantes en cuando al acceso al código, aun asi, es un sistema que cada dia va creciendo y mejorando.

2.

a)

El Canvas de Unity nos permite mostrar una capa de información (e interacción, puesto que puede incluir botones y otros elementos interactivos) entre la cámara y la escena de juego, de forma que dicha información siempre es visible en la escena, independientemente de la posición en la que nos encontremos. Este es una parte importante del Ui, ya que nos permite manejarla sin preocuparnos del espacio que estemos utilizando en nuestro gameplay, además de que nos permite modificar y adjustar la Ui a parte del propio espacio del juego.

Canvas Scaler: Componente encargado de controlar el escalado y la densidad total de píxel de los elementos de interfaz de usuario en el objeto Canvas. Dicho escalado afecta a todo elemento gráfico que se situé dentro del objeto Canvas, incluyendo los tamaños de fuente y bordes de la imagen.

Canvas Group: Puede ser utilizado para controlar ciertos aspectos de un grupo entero de elementos UI de un lugar sin la necesidad de manejarlos de manera individual. Las propiedades del Canvas Group afectan el GameObject al igual que sus propios hijos.

Canvas Renderer: Componente encargado de procesar los objetos gráficos de la interfaz de usuario, situados dentro del canvas.

b)

Image: El Image muestra una imagen no interactiva al usuario. Puede utilizar esto para fines como decoraciones o iconos, y puede cambiar la imagen desde un script para reflejar los cambios en otros controles. El control es similar al Raw Image Control, pero ofrece más opciones para animar la imagen y rellenar con precisión el rectángulo de control. Sin embargo, el control Imagen requiere que una Textura para que sea un Sprite, mientras que el Raw Image puede aceptar cualquier textura.

Text: El control Text muestra un fragmento de texto no interactivo al usuario. Esto se puede utilizar para proporcionar títulos o etiquetas para otros controles GUI o para mostrar instrucciones u otro texto.

Button: El control Botón responde a un clic del usuario y se utiliza para iniciar o confirmar una acción.

Slider: El Slider control permite al usuario seleccionar un valor numérico de un rango predeterminado arrastrando el mouse. Tenga en cuenta que el control similar ScrollBar se utiliza para deslizarse en lugar de seleccionar valores numéricos.

c)

Horizontal Layout: El componente Horizontal Layout Group coloca sus layout elements (Elementos de diseño) hijos al lado de cada uno, lado a lado. Sus anchuras son determinadas por su anchuras mínimas, preferidas y flexibles respectivas

Vertical Layout: El componente Vertical Layout Group coloca sus layout elements (Elementos de diseño) (Elementos de diseño) hijos encima de cada uno. Sus alturas son determinadas por sus alturas mínimas, preferidas y flexibles respectivas

d)

El componente Canvas Scaler es utilizado para controlar la escala en general y densidad de pixeles de los elementos UI en el Canvas. Esta escala afecta todo debajo los Canvas, incluyendo los tamaños de fuentes y bordes de imágenes.

Utilizando el modo Constant Pixel Size, las posiciones y tamaños de los elementos UI son especificados en pixeles en la pantalla. Esto también es la funcionalidad por defecto del Canvas cuando no hay un Canvas Scaler adjunto. Sin embargo, con los ajustes de With the Scale Factor en el Canvas Scaler, una constante que escala puede ser aplicada a todos los elementos UI en el Canvas.

Al utilizar el modo Scale With Screen Size, las posiciones y tamaños pueden ser especificados de acuerdo a los pixeles de una resolución de referencia específica. Si la resolución de pantalla actual es mayor que la resolución de referencia, el Canvas va a seguir teniendo la resolución de la resolución de referencia, pero se va a escalar con el fin de encajar en la pantalla. Si la resolución actual de la pantalla es más pequeña que la resolución referencia, el Canvas de manera similar será escalada para abajo para encajar.

e)

Generalmente cuando se posicione un elemento UI con su Rect Transform, su posición y tamaño es especificado manualmente. Sin embargo, cuando se quiere que el tamaño del rectángulo sea configurado de manera automática para que encaje con el contenido del elemento UI. Esto se puede hacer al agregar un componente llamado Content Size Fitter.

Lo que sucede aquí es que el componente Text funciona como un Layout Element (Elemento de diseño) que puede proporcionar información acerca de qué tan grande es su tamaño mínimo y preferido. En un diseño manual esta información no es utilizada. Un Content Size Fitter es un tipo del Layout Controller, el cual escucha a la información del diseño proporcionado por Layout Elements y controla el tamaño del Rect Transform de acuerdo a esto.

3.

a)

i) **VisualElement**:

* + La base de todos los elementos de UI en el Toolkit.
  + Puede tener otros elementos y tiene múltiples propiedades y estilos para personalizar su apariencia y comportamiento.

1. **Label**:
   * Muestra texto en la interfaz de usuario.
   * Puede contener texto estático o ser actualizado directamente desde el código.
2. **Button**:
   * Elemento que permite a los usuarios hacer click en él.
   * Puede tener eventos de click asociados para realizar acciones específicas.
3. **TextField**:
   * Un campo de entrada de texto que permite a los usuarios ingresar y editar texto.
   * Puede utilizarse para formularios y entradas de texto.
4. **Slider**:
   * Un control deslizante que permite a los usuarios seleccionar un valor dentro de un rango.
   * Útil para ajustar valores numéricos, como volumen o intensidad.
5. **Toggle**:
   * Un interruptor de encendido/apagado que permite a los usuarios cambiar entre dos estados.
6. **Image**:
   * Muestra una imagen en la interfaz de usuario.
   * Puede utilizarse para iconos, imágenes de fondo y otros elementos visuales.
7. **VisualTreeAsset**:
   * Una representación serializable de un conjunto de elementos de UI.
   * Permite la creación eficiente de múltiples instancias de elementos similares.
8. **UXML**:
   * Un formato XML utilizado para describir la jerarquía de elementos de UI.
   * Puede ser utilizado para crear interfaces de usuario de manera visual o programáticamente.
9. **StyleSheets**:
   * Permite definir estilos de manera declarativa para aplicar apariencias coherentes a los elementos de UI.
   * Usa un lenguaje similar a CSS para definir estilos.
10. **Layouts**:
    * Componentes que facilitan la organización y el posicionamiento de elementos en la interfaz de usuario.
    * Incluye cajas (Box), rejillas (Grid), pilas (Stack), entre otros.

b)

Visual Tree es un concepto del Unity UI Toolkit y hace referencia a la estructura de jerárquica de los elementos de la interfaz del usuario.

Este es importante ya que permite organizar los elementos visuales de la interfaz en tiempo de ejecución. El visual tree permite organizar elementos y sus subelementos de manera eficiente para el rendimiento del proyecto, gestionar las interacciones de eventos de entradas y modificarlos directamente desde el código.

c)

UXML es un formato de archivo de Unity UI Toolkit usado para establecer la estructura de la interfaz de usuario de un proyecto.

El formato UXML es similar al XML y HTML pero en un diferente contexto, puesto que el primero es usado en Unity, con configuraciones, características y especificaciones diseñadas para el uso del visual tree del UI toolkit, permitiendo al usuario personalizar la interfaz de su proyecto de Unity, mientras que los otros 2 formatos son usados también para la estructura de interfaces pero en el diseño web y no tienen relación con Unity o su estructura de interfaces.

d)

USS es un formato de archivo utilizado en el Unity UI Toolkit, similar al formato CSS, pero desarrollado específicamente para definir estilos visuales y comportamientos de elementos de una interfaz en proyectos de Unity.

USS esta especializado para el diseño específico de elementos dentro de la interfaz de Unity (como botones, etiquetas, paneles…), uso de la sintaxis del mismo y la integración con los comportamientos y eventos relacionados con la interfaz dentro del proyecto, mientras que CSS no está desarrollado con este contexto en mente, en cambio, su creación fue para la utilización en el diseño de páginas web.