UD 4

DESARROLLO DE APLICACIONES INTERACTIVAS DE ENTRETENIMIENTO

Álvaro Holguera – CIFP José Luis Garci (Madrid)

Este documento está bajo una licencia de Creative Commons

www.laultimapregunta.com

ÍNDICE

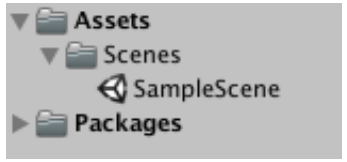
ESCENAS

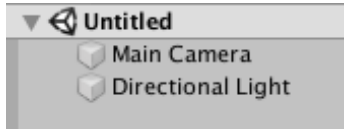
MANEJAR Y CAMBIAR ESCENAS

Álvaro Holguera (CIFP José Luis Garci)

GESTIONANDO LAS ESCENAS

- Una escena es lo que contiene todos los elementos de nuestro juego (personajes, terrenos, menús…). Es como si cada escena fuese un nivel
- Podemos crear tantas escenas como queramos ("Ctrl+n") y guardarlas en la carpeta del proyecto que queramos (es importante guardar la escena una vez creada y cada vez que realicemos cambios)
  - Unity por defecto crear una escena al crear un proyecto nuevo, que aparecer como "Untitled" en nuestra ventana de jerarquíia hasta que la guardemos
  - Es posible incluso editar varias escenas a la vez, mediante la herramienta "Multi-Scene editing"





- Todas las escenas que creemos deberán ser añadidas en la compilación del proyecto, si no, no aparecerán en el juego final:

  File>Build Settings

- Se puede añadir la escena abierta en ese momento, o arrastrarlas directamente desde la ventana del proyecto
- Veremos que se les asigna automáticamente un número de índice a la derecha, comenzando por 0.
  - Este nº sirve para identificar a cada escena en el código, además de por su nombre, muy útil si queremos cambiarle el nombre a la escena en algún momento

SCENEMANAGER

- Mediante este método podemos cargar otras escenas desde nuestro código
- Para usarlo, es imprescindible usar la librería "UnityEngine.SceneManagement"
- Las funciones básicas son:
  - LoadScene: carga la escena indicada, poniendo el nombre de la escena o su nº índice que se la ha dado en el Build Settings
  - LoadSceneAsync: a diferencia de LoadScene, este método carga la escena en segundo plano y avisar cuándo se ha terminado de cargar. Esto es muy útil para crear escenas de carga
  - MoveGameObjectToScene: permite mover el GameObject indicado a la escena de destino. Solo funciona con GameObjects del raíz (no se puede mover hijos), y la escena tiene que estar cargada en el background, de forma asíncrona, antes de moverlo.

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

public class SalidaEscena1 : MonoBehaviour
{

    private void OnTriggerEnter(Collider other)
    {
        //Cargamos la escena con el índice 1
        SceneManager.LoadScene(1);
    }

}
```

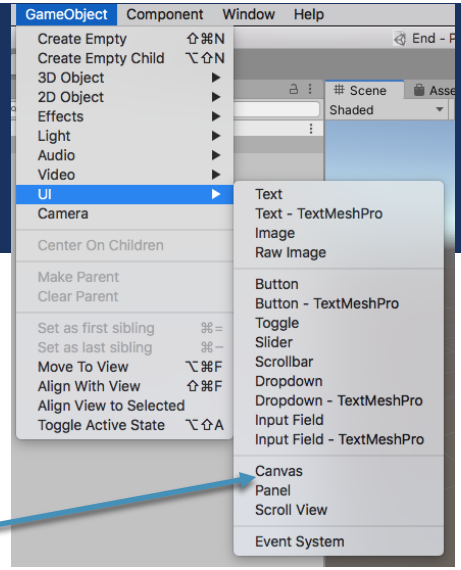
NOTA: Si queremos que unas variables estén disponibles en varias escenas, podemos crearlas de tipo "static" en nuestras clases, y de esa forma su valor no cambiará al cambiar de escena

UI

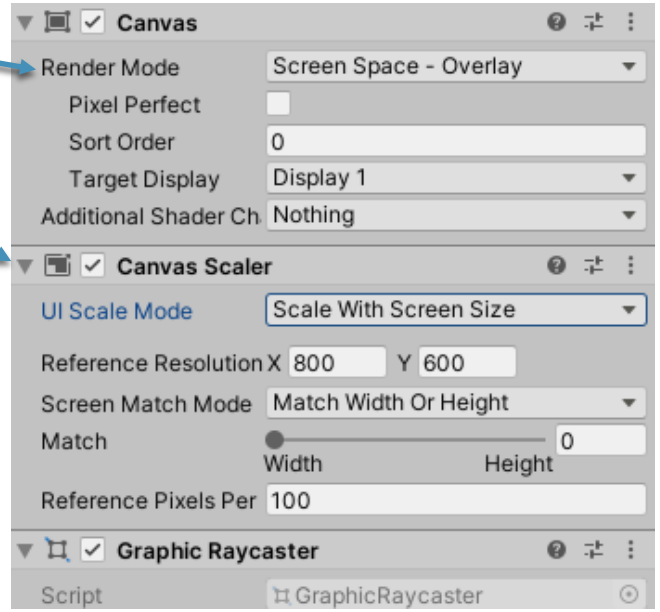
USER INTERFACE

La UI es la forma que tiene el juego de comunicarse con el jugador, darle información (como estado, vida, mensajes, etc.), y también permite recabar información a partir de botones, campos de texto, etc.

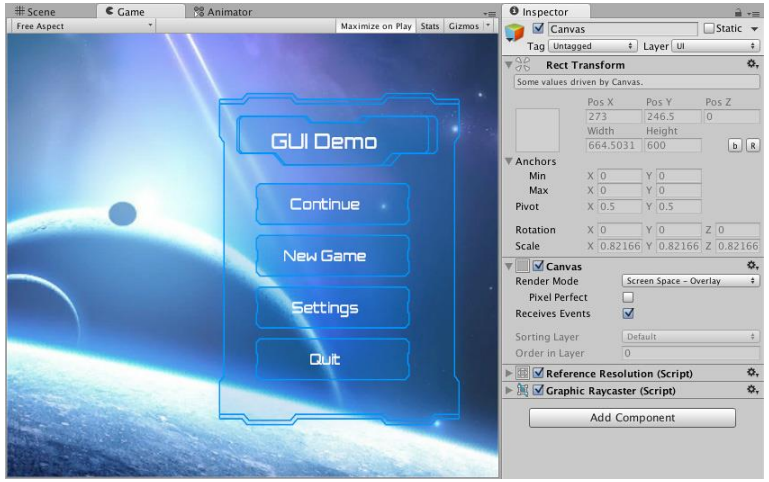
USER INTERFACE - CANVAS



- Todos los elementos de un UI tienen que ser hijos de un elemento padre de tipo canvas, que dibujará un cuadrado en el escenario
- Al insertar un elemento de UI, ya sea un texto, una imagen o un botón, se crea automáticamente. Este puede contener varios elementos, y a su vez tiene varias opciones:
  - Render mode: cómo se adapta a la vista del juego. En la siguiente diapositiva se muestran las opciones
  - Canvas scaler: determina en caso de que el canvas se ajuste a la ventana del juego o la cámara, cómo debe escalarse en caso de cambiar el tamaño y si debe deformarse o no si cambia la proporción de aspecto
- En el menú de GameObject, entre las opciones tenemos las de UI: textos, imágenes, botones, campos de textos, etc
  - Estos objetos son 2D, por ello, para modificarlos muchas veces es útil pulsar la vista 2D de la escena.
- Podemos hacer un canvas invisible (desactivándolo en el inspector) y haciéndolo visible, por ejemplo para mostrar una pantalla de Game Over.

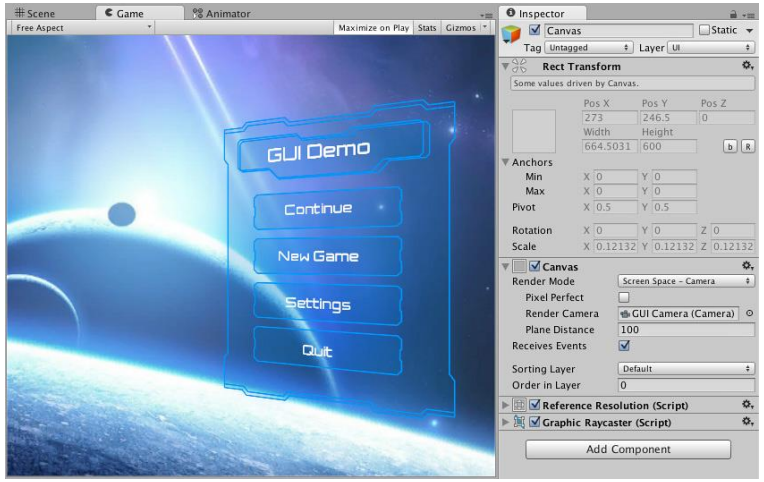


RENDER MODE



Screen Space - Overlay
(Superposición)

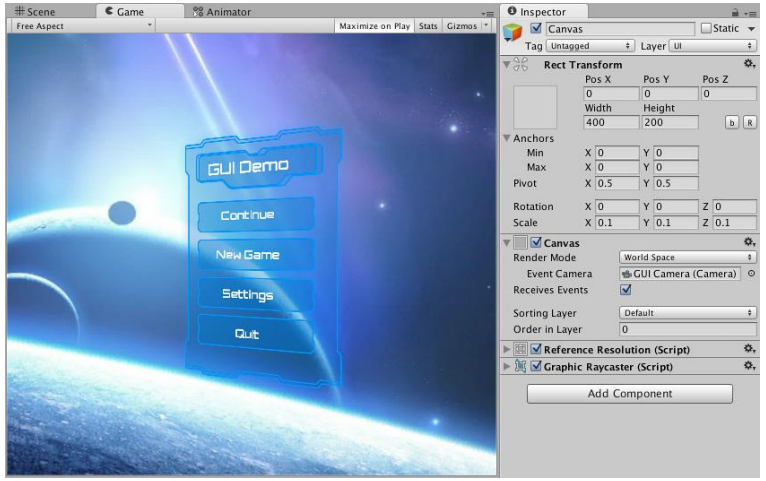
Este modo de renderización coloca elementos UI en la pantalla mostrada en la parte superior de la escena. Si el tamaño de la pantalla es modificada o cambia la resolución, el Canvas va a automáticamente cambiar el tamaño para que coincida.



Screen Space - Camera

Esto es similar a Screen Space - Overlay, pero en este modo de renderizado, el Canvas se coloca a una distancia dada (Plane Distance) delante de una Camera especificada.

La configuración de la cámara afecta la apariencia de la interfaz de usuario. Si la cámara está configurada en Perspectiva, los elementos de la interfaz de usuario se mostrarán con perspectiva, y si la pantalla cambia de tamaño, cambia la resolución, el canvas cambiará automáticamente el tamaño para que coincida también.

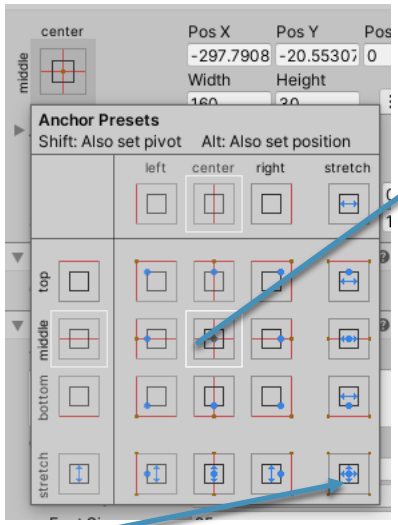


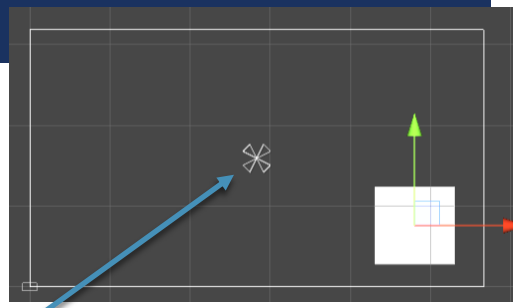
World SPace

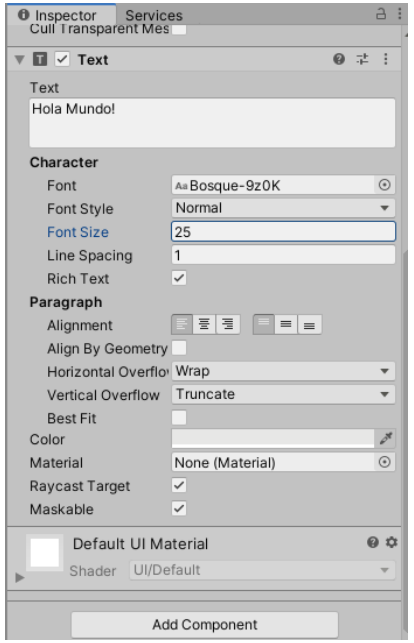
En este modo de renderizado, el Canvas se va a comportar como cualquier otro objeto en la escena. El tamaño de este Canvas puede ser configurado manualmente utilizando su Rect Transform, y los elementos UI van a renderizar al frente o detrás de otros objetos en la escena basados en una colocación 3D. Esto es útil para UIs que están destinados a ser parte del mundo. Esto también es conocido como “diegetic interface”.

UBICAR ELEMENTOS EN EL CANVAS

- Al colocar una imagen, un texto o cualquier otro elemento en el canvas tendremos que ubicar su posición.
  - Podemos usar los parámetros del componente transform ("Rect Transform"), y moverlo o escalarlo
- Si pinchamos en la posición del elemento, podemos indicar el punto de anclaje respecto al canvas en el que está incluido
  - La mayoría de las opciones no cambia la posición del elemento, sino su punto de anclaje (representado por un puntero blanco), por lo que tendremos que modificar la posición del elemento a 0,0,0
  - Si pulsamos Shift+Alt al hacer click en la posición, se moverá automáticamente
  - Algunas opciones rellenan el canvas, muy útiles para imágenes de fondo
- En el inspector podremos modificar sus parámetros básicos: color, tamaño, fuente, alineación, etc.
  - Podemos imprtar a nuestro proyecto fuentes externas







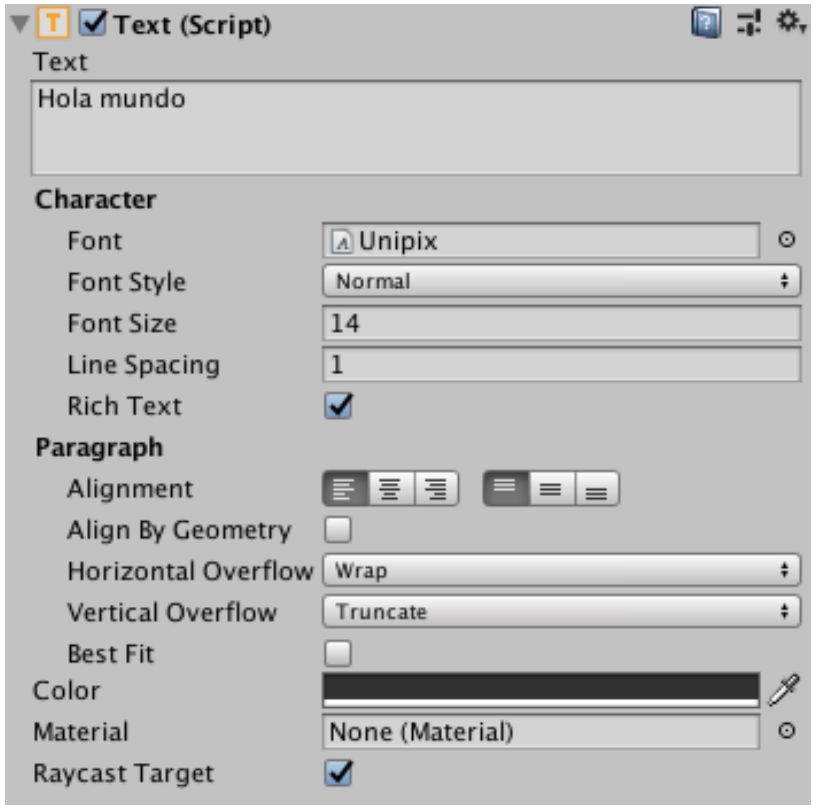
TEXTOS

- La opción más básica es añadir textos a nuestro Canvas
  - IMPORTANTE: si en lugar de texto, ponemos una imagen con texto, no podremos cambiarlo a través del código
- Podemos modificar sus parámetros básicos
  - Fuente y estilos: podemos cargar en nuestro proyectos tipografías, siempre respetando las licencias de uso
  - Tamaño de fuente y espaciado:
  - La opción de "Rich Texto" nos permitirá añadir estilos medianate código
  - Parámetros del párrafo: alineación, o cómo se comporta con la caja que lo contiene
- En scripting, podemos crear en un script asociado al canvas la variable pública de tipo Text, asignarle el texto en Unity, y mediante código modificar sus parámetros, como el texto que contiene
- Para poder acceder a los elementos del UI mediante scripts debemos cargar la librería:

```
using UnityEngine.UI;
```

- Si creamos una variable pública (o serializada) de tipo Text, podremos arrastrar al campo disponible en el Inspector el GamObject del texto disponible en el canvas, y a partir de ese momento cambiar el contenido, o el color, o cualquier parámetro, mediante:

```
myVariable.text = "hola mundo";
myVariable.color = Color.white;
```



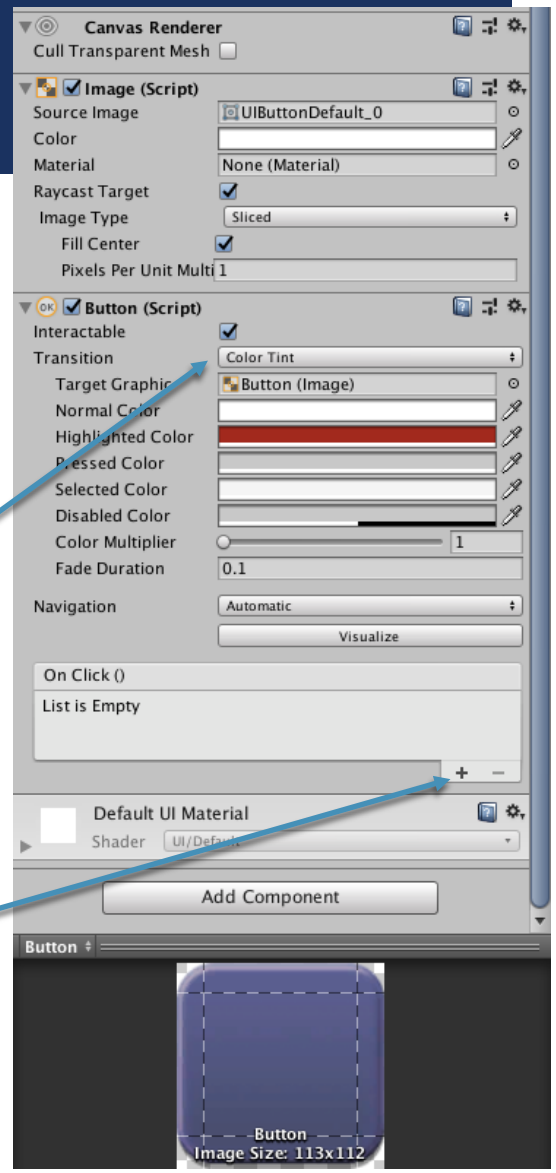
IMPORTANTE: cualquier texto que ponemos en un UI de tipo texto, debemos convertirlo en string mediante el método .ToString();

SLIDERS

- Una herramienta disponible en la UI que nos puede ofrecer muchas utilidades son los Sliders.
- Usos habituales:
  - Barra de estado de carga de una escena
  - Modificar parámetros del juego por el usuario (volumen, brillo, etc)
  - Barras de vida usando sprites de fondo
- Mediante código, podemos cambiar la posición del slider, o leer su estado, para actuar en consecuencia

BOTONES

- Unity permite añadir un gran n° de elementos interactivos al canvas. El más básico de todos es el botón
- Al añadirlo, mediante GameObject>UI>Button, veremos que añadre 2 elementos
  - El botón en sí, que permite añadir imágenes de fondo
  - El texto, que se comporta como una caja de texto normal (podemos eliminarlo si queremos)
  - Podemos sustituirlos por una imagen de sprite, y borrar ese hijo
- Podemos modificar cómo se comporta al interaccionar con él, en el componente "Button", con varias opciones:
  - Cambio de color: dependiendo del estado del botón lo tintamos de un color u otro (blanco lo deja como está)
  - Sprite Swap: permite cambiar la imagen de fondo dependiendo del estado del botón
  - Animation. Si pulsamos el botón "Auto Generate Animation", guardará un Animator Controller con tantas animaciones como estados tiene el botón. Podremos ir a cada animación y modificar sus parámetros en la línea de tiempo.
- En la opción de "Navigation" determinaremos cómo se comportan los cursores del teclado cuando tenemos varios botones en pantalla.
  - Podemos dejarlo que Unity lo configure de forma automática, o podemos indicar a qué botón se accede cuando usamos los cursores. Si pulsamos el botón para visualizar, veremos cómo está configurado mediante flechas amarillas
- También podemos llamar a métodos de código al hacer click en los botones
  - El añadir un evento, nos permite adjuntar GameObjects, que si tienen scripts asociados nos permitirá llamar a funciones que incluyan esos scripts
  - En el desplegable de funciones, ahora aparecerá el script asociado a ese GameObject y dentro las funciones disponibles, habrá una con la clase creada
- Desde código podemos añadir *listeners* a los botones o a cualquier elemento del UI, lo que nos da mayor control sobre la interactividad, aunque de forma más compleja

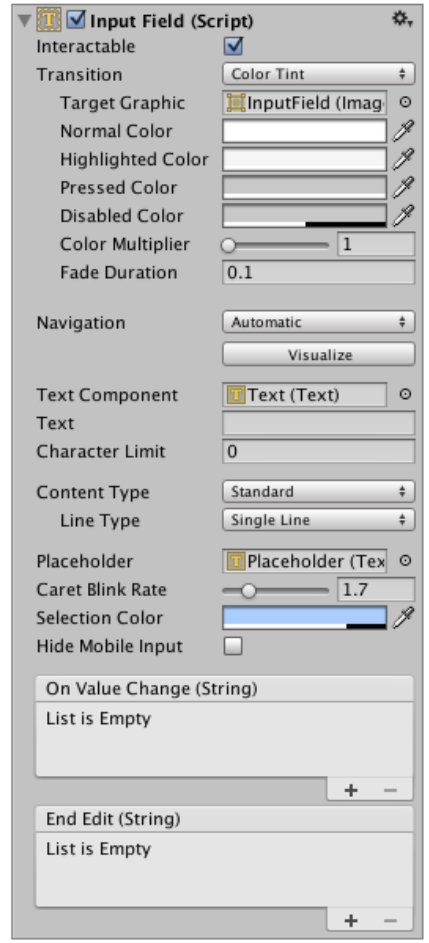


CAMPOS DE ENTRADA DE DATOS (INPUT FIELDS)

- Otro tipo de elemento que podemos añadir a nuestro canvas son los campos de entrada de datos (o *input fields*), diseñados para reoger información directamente del usuario
- Al igual que los botones normales, permite añadir transiciones para sus diferentes estados, o incluso añadir imágenes de fondo
- Otra opción es vincularlo al componente de texto que Unity crea automáticamente, y que recogerá los datos que escriba el usuario.
- Más opciones:
  - Limitar el n° de caracteres
  - Definir el tipo de texto que se va a introducir (Standard, email, password, etc.)
  - Permitir introducir textos de varias líneas
  - Placeholder: texto semitransparente que aparece por defecto en la caja, pero que desaparece cuando comenzamos a escribir en ella
  - Etc.
- Como los botones, podemos vincularlo con scripts en la escena y lanzar métodos que se ejecutan cuando cambia algo el contenido de la caja, o al salir el usuario de la caja indicando que ha terminado
- Desde el código, podemos obtener en cualquier momento lo que está escrito en la caja de texto, creando una variable de tipo Text, vinculada al objeto, y obteniendo su propiedad "text" (es recomendable convertirla en variable de tipo String"

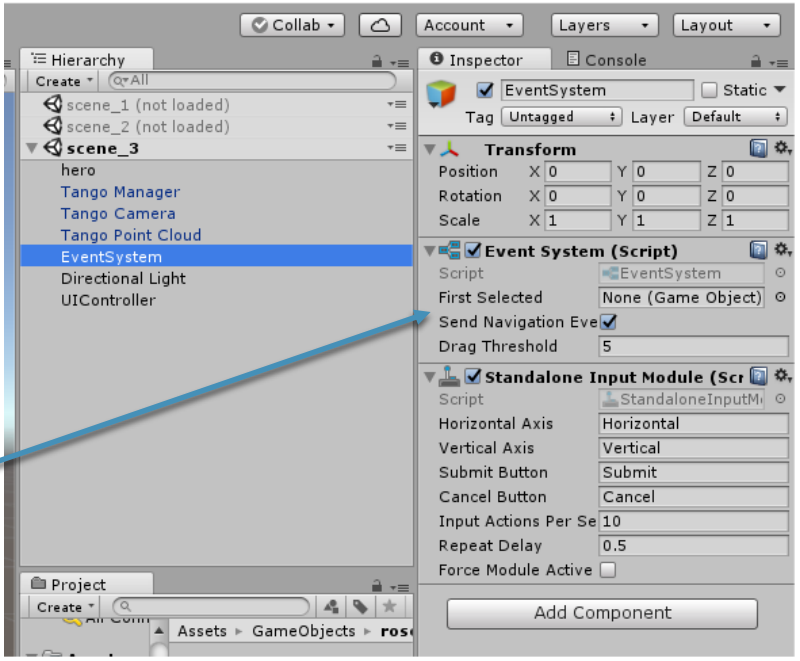
```
public InputField Username_field; //lo vinculamos en Unity

    string userID = Username_field.text.ToString();
```



EVENT SYSTEM

- Toda la interactividad de un canvas se gestiona a través del Event System:
- El EventSystem es una manera de enviar eventos a objetos en la aplicación basado en input, sea el teclado, mouse, tacto, o un input personalizado.
  - Está diseñado como un administrador y facilitador de comunicaciones entre módulos del EventSystem.
- Las funciones principales que gestiona son:
  - Primer botón seleccionado (podemos hacerlo por código)
  - Desplazamiento entre los botones
  - Manejar qué InputModule está en uso



IMPORTANTE: un mismo Event System gestiona todos los canvas, por ello, es buena idea desactivarlos cuando no estén visibles

LISTENERS

- Podemos gestionar la interacción con los botones (y con otros elementos del UI) directamente con el código.
- OnClick:
  - Creamos una variable púbica de tipo Button (es necesario añadir la librería UnityEngine.UI)

```
public Button m_YourButton;
```

  - Vinculamos en Unity el botón a esa variable, y en el método Startañadimos un Listener que ejecutará una función (TaskOnClick()) cuando se haga click:

```
m_YourFirstButton.onClick.AddListener(TaskOnClick);
```

```
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class InitMenu : MonoBehaviour
{
    public Button m_YourFirstButton

    void Start()
    {
        m_YourFirstButton.onClick.AddListener(TaskOnClick);
    }

    void TaskOnClick()
    {
        Debug.Log("You have clicked the button!");
    }
}
```

```
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class TestButton : MonoBehaviour, ISelectHandler, IPointerEnterHandler
{
    //Do this when the selectable UI object is selected.
    public void OnSelect(BaseEventData eventData)
    {
        Debug.Log(this.gameObject.name + " was selected");
    }

    //Do this when the cursor enters the rect area of this selectable UI
object.
    public void OnPointerEnter(PointerEventData eventData)
    {
        Debug.Log("The cursor entered the selectable UI element.");
    }

}
```

- Selectable
  - Cualquier elemento de la UI que se pueda seleccionar (como un botón) puede ser detectado mediante código cuando es seleccionado, o cuando pasa el ratón por encima, etc.
  - Es necesario añadir la librería UnityEngine.EventSystems
  - También debemos añadir a continuación de la MonoBehaviour la clase extendida que vayamos a usar, dependiendo del evento (ej.- "ISelectHandler", ó "IPointerEnterHandler")
  - Este script lo tendremos que añadir al botón que queremos controlar