

UT.2 Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Parte E

En esta Unidad de Trabajo vamos a tratar los siguientes puntos:

- Introducción. *Diapositivas 3-5*
- Puntero a Componentes. *Diapositivas 6-18*
- Acceso a los componentes y a sus parámetros. *Diapositivas 19-25*
- Trabajando con nuestras clases. *Diapositivas 26-30*
- Trabajando con teclado y creando atajos de teclado. *Diapositivas 31-40*

Material Adicional a esta presentación:

- **Software:**
 - UT2_Ejemplo.unitypackage

Prácticas:

- Práctica_UT2_IntroduccionInterfaceGrafica

Hasta este punto hemos visto como acceder a las propiedades de los componentes Panels, Text y Button desde el entorno Unity.

Ahora, vamos a ver como acceder de una forma dinámica, es decir, a través de programación en C# utilizando los Scripts.

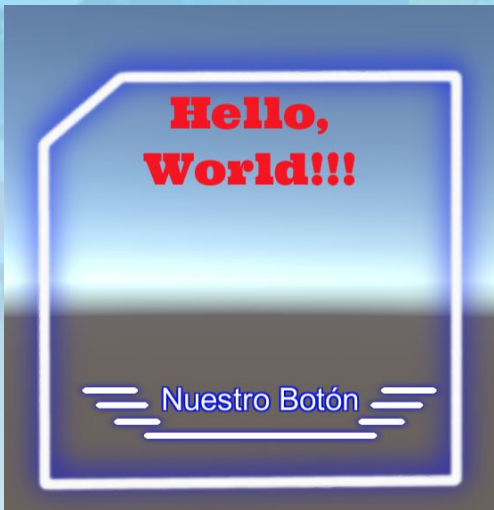
Esto nos va a proporcionar varias ventajas:

- Poder modificar el comportamiento de los componentes dinámicamente en tiempo de ejecución.
- Modificar el valor de los parámetros de los componentes.
- Estandarizar el trabajo de los componentes para distintos proyectos, utilizando clases.
- Acceso a datos externos mediante ficheros y/o BBDD.
- Comprobación de errores.

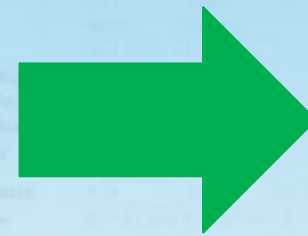
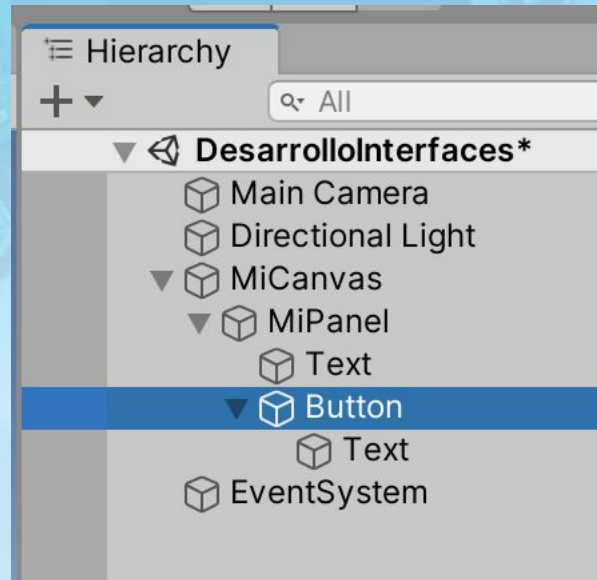


La gestión de Errores se verá en una UT posterior.

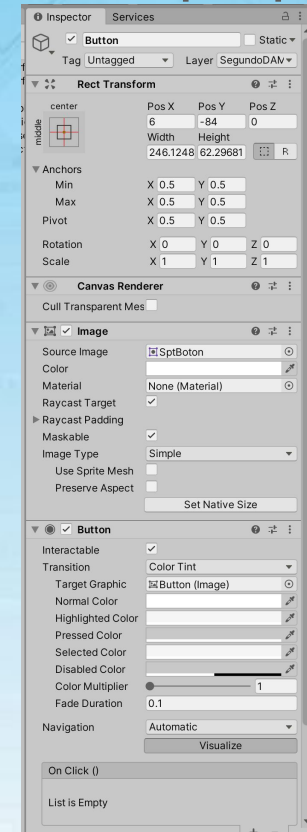
La forma de acceder a las propiedades de un componente ha sido, seleccionando el componente en la ventana Hierarchy y accediendo a los parámetros de configuración desde la ventana Inspector. Por ejemplo, si queremos acceder a las propiedades de nuestro botón.



Seleccionamos el componente

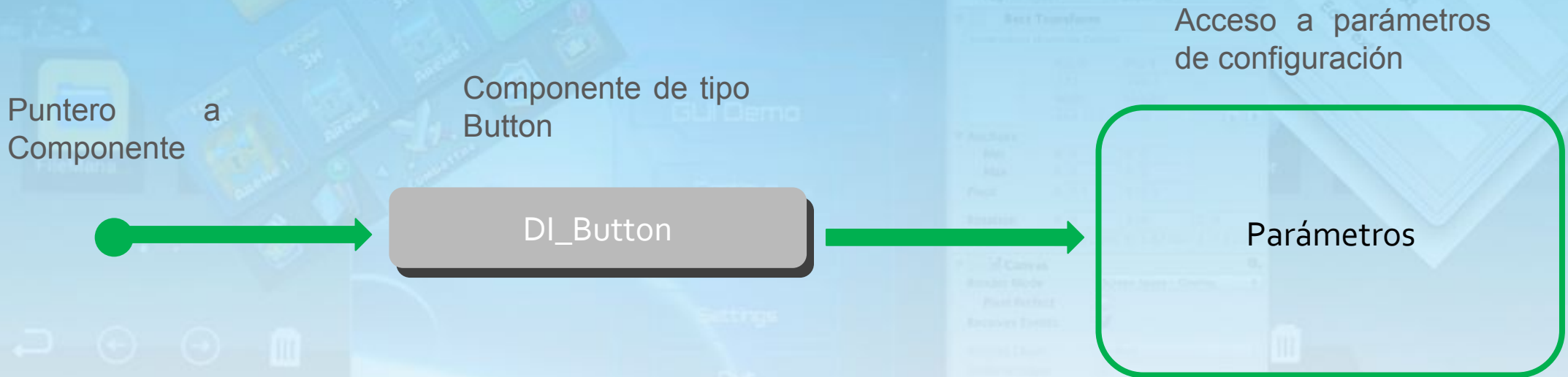


Accedemos a los parámetros de configuración



Introducción

Para acceder desde un Script a los parámetros de configuración vamos a seguir la misma filosofía. Crearemos una **variable** que nos va a servir de **puntero** al Componente. Una vez que tenemos el puntero mirando al Componente, vamos a trabajar con **Objetos**, cada Componente es un Objeto y para acceder a sus parámetros, pondremos el nombre del Objeto + “.” + Propiedad.



Puntero a Componentes

Hay varias formas de acceder a los componentes:

- Por el nombre.
- Por la etiqueta (Tag).
- Por el tipo de componente.

Lo primero que tenemos que hacer es crear una variable del **mismo tipo** que el del componente al que queremos hacer referencia desde nuestro Script.

Para hacer esto, debemos tener muy claro, que el entorno Unity, hace referencia a todos los elementos que aparecen en la ventana Hierarchy como un tipo de objeto denominado **GAMEOBJECT**.

Así que vamos a **crear una variable de tipo GameObject**, que nos servirá para tener un punto de referencia al **Objeto/Componente** que queremos **referenciar**.

Puntero a Componentes

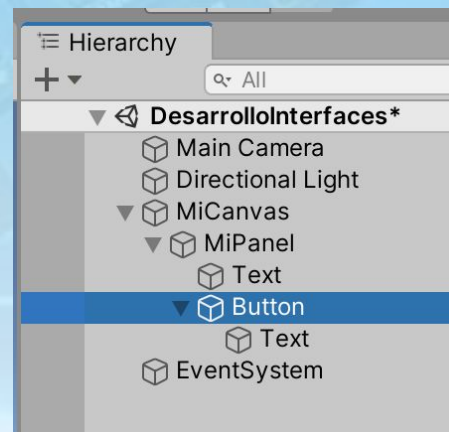
Una variable la podemos declarar de dos tipos:

- Pública
- Privada

La filosofía de este curso, es trabajar siempre con variables privadas.

Veamos como hacer un puntero a nuestro botón

Creamos una clase "ScrBoton" y creamos una variable en "ScrNuestroBoton"



Tenemos nuestro Botón que se denomina "Button"

```
1  /*
2  * Date: 16 de Agosto de 2020.
3  * Class: ScrBoton.
4  * Description: En esta clase podremos leer
5  * y escribir los distintos parámetros de con-
6  * figuración de los botones.
7  * Author: Raquel Rojo y Mario Santos.
8  * Palomeras Vallecas - Villablanca
9  */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14 using UnityEngine.UI;
15
16 public class ScrBoton : MonoBehaviour
17 {
18     private GameObject btnPunteroBoton;
19     public ScrBoton(string nombre)
20     {
21         //Buscamos el GameObject al que queremos
22         //apuntar por el nombre.
23         btnPunteroBoton = GameObject.Find(nombre);
24     }
25     //Método para ver si el Botón está activo o no.
26     public void GetInteractivo()
27     {
28         Debug.Log(btnPunteroBoton.GetComponent<Button>().interactable);
29     }
30     //Método para activar/desactivar el botón.
31     public void SetInteractivo()
32     {
33         btnPunteroBoton.GetComponent<Button>().interactable =
34             !btnPunteroBoton.GetComponent<Button>().interactable;
35     }
36 }
37
38
39
40
41
42
43
44
```



```
1  /*
2  * Date: 16 de Agosto de 2020.
3  * Class: ScrNuestroBoton.
4  * Description: Desde este script, creamos una
5  * variable de tipo ScrBoton y llamamos a sus
6  * métodos.
7  * Author: Raquel Rojo y Mario Santos.
8  * Palomeras Vallecas - Villablanca
9  */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14
15 public class ScrNuestroBoton : MonoBehaviour
16 {
17     //declaramos una variable de tipo ScrBoton
18     private ScrBoton btnNuestroBoton;
19     public void Prueba()
20     {
21     }
22     //Llamamos al constructor de la clase ScrBoton.
23     btnNuestroBoton = new ScrBoton("Button");
24     //Llamamos a los métodos GetInteractivo y
25     //SetInteractivo.
26     btnNuestroBoton.GetInteractivo();
27     btnNuestroBoton.SetInteractivo();
28 }
29
30
```

Veamos más detalladamente estos Scripts. Empecemos con el Script en el que hemos creado la clase ScrBoton.

En lo primero que nos tenemos que fijar es en la línea que hemos añadido a nuestro Script.

Using UnityEngine.UI;

Lo que hemos hecho ha sido cargar la **librería** de UI del entorno de Unity. Por defecto, esta librería no viene cargada. Si no cargamos esta librería, Unity no reconocería clases como “Button”, “Text”, “Panel”, etc.

Para cargar una librería siempre se empieza por la palabra **Using** y a continuación el nombre de la librería.

```

1  /*
2   * Date: 16 de Agosto de 2020.
3   * Class: ScrBoton.
4   * Description: En esta clase podremos leer
5   * y escribir los distintos parámetros de con-
6   * figuración de los botones.
7   * Author: Raquel Rojo y Mario Santos.
8   * Palomeras Vallecas - Villablanca
9   */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14 using UnityEngine.UI;
15
16 public class ScrBoton : MonoBehaviour
17 {
18     private GameObject btnPunteroBoton;
19
20     public ScrBoton(string nombre)
21     {
22         //Buscamos el GameObject al que queremos
23         //apuntar por el nombre.
24
25         btnPunteroBoton = GameObject.Find(nombre);
26     }
27
28     //Método para ver si el Botón está activo o no.
29     public void GetInteractivo()
30     {
31
32         Debug.Log(btnPunteroBoton.GetComponent<Button>().interactable);
33     }
34
35     //Método para activar/desactivar el botón.
36     public void SetInteractivo()
37     {
38         btnPunteroBoton.GetComponent<Button>().interactable =
39             !btnPunteroBoton.GetComponent<Button>().interactable;
40     }
41 }
42
43
44

```


UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Puntero a Componentes

```

1  /*
2   * Date: 16 de Agosto de 2020.
3   * Class: ScrBoton.
4   * Description: En esta clase podremos leer
5   * y escribir los distintos parámetros de con-
6   * figuración de los botones.
7   * Author: Raquel Rojo y Mario Santos.
8   * Palomeras Vallecas - Villablanca
9   */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14 using UnityEngine.UI;
15
16 public class ScrBoton : MonoBehaviour
17 {
18     private GameObject btnPunteroBoton;
19
20     public ScrBoton(string nombre)
21     {
22         //Buscamos el GameObject al que queremos
23         //apuntar por el nombre.
24
25         btnPunteroBoton = GameObject.Find(nombre);
26     }
27
28     //Método para ver si el Botón está activo o no.
29     public void GetInteractivo()
30     {
31
32         Debug.Log(btnPunteroBoton.GetComponent<Button>().interactable);
33     }
34
35     //Método para activar/desactivar el botón.
36     public void SetInteractivo()
37     {
38         btnPunteroBoton.GetComponent<Button>().interactable =
39             !btnPunteroBoton.GetComponent<Button>().interactable;
40     }
41 }
42
43
44

```

La segunda línea en la que nos vamos a fijar es:

`private GameObject btnPunteroBoton;`

La primera palabra con la que nos encontramos en esta sentencia es con “**private**”. Esto quiere decir que ha esta variable, únicamente se va a tener acceso desde cualquier método de este script y desde el entorno de Unity, NO se tendrá acceso.

Si la hubiéramos declarado como “**public**”, sería lo contrario que lo dicho en el párrafo anterior y estaría accesible desde el entorno.

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Puntero a Componentes

```

1  /*
2   * Date: 16 de Agosto de 2020.
3   * Class: ScrBoton.
4   * Description: En esta clase podremos leer
5   * y escribir los distintos parámetros de con-
6   * figuración de los botones.
7   * Author: Raquel Rojo y Mario Santos.
8   * Palomeras Vallecas - Villablanca
9   */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14 using UnityEngine.UI;
15
16 public class ScrBoton : MonoBehaviour
17 {
18     private GameObject btnPunteroBoton;
19
20     public ScrBoton(string nombre)
21     {
22         //Buscamos el GameObject al que queremos
23         //apuntar por el nombre.
24
25         btnPunteroBoton = GameObject.Find(nombre);
26     }
27
28     //Método para ver si el Botón está activo o no.
29     public void GetInteractivo()
30     {
31
32         Debug.Log(btnPunteroBoton.GetComponent<Button>().interactable);
33     }
34
35     //Método para activar/desactivar el botón.
36     public void SetInteractivo()
37     {
38         btnPunteroBoton.GetComponent<Button>().interactable =
39             !btnPunteroBoton.GetComponent<Button>().interactable;
40     }
41 }
42
43
44

```

La segunda línea en la que nos vamos a fijar es:

`private GameObject btnPunteroBoton;`

La segunda palabra que nos encontramos es “GameObject”. Ya hemos mencionado que todos los componentes que aparecen en nuestras ventanas “Scene” y “Hierarchy” son de un tipo denominado **GameObject**, así que el puntero que vamos a utilizar para hacer referencia a ellos, debe de ser del mismo tipo.

Podríamos declarar punteros del tipo del componente del **GameObject** del cual queremos cambiar su configuración. Por ejemplo, del **GameObject Button**, si lo que queremos es hacer cambios en nuestro componente **Button**. La sentencia quedaría como la que se muestra a continuación.

`private Button btnPunteroBoton;`

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Puntero a Componentes

```

1  /*
2   * Date: 16 de Agosto de 2020.
3   * Class: ScrBoton.
4   * Description: En esta clase podremos leer
5   * y escribir los distintos parámetros de con-
6   * figuración de los botones.
7   * Author: Raquel Rojo y Mario Santos.
8   * Palomeras Vallecas - Villablanca
9   */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14 using UnityEngine.UI;
15
16 public class ScrBoton : MonoBehaviour
17 {
18     private GameObject btnPunteroBoton;
19
20     public ScrBoton(string nombre)
21     {
22         //Buscamos el GameObject al que queremos
23         //apuntar por el nombre.
24
25         btnPunteroBoton = GameObject.Find(nombre);
26     }
27
28     //Método para ver si el Botón está activo o no.
29     public void GetInteractivo()
30     {
31
32         Debug.Log(btnPunteroBoton.GetComponent<Button>().interactable);
33     }
34
35     //Método para activar/desactivar el botón.
36     public void SetInteractivo()
37     {
38         btnPunteroBoton.GetComponent<Button>().interactable =
39             !btnPunteroBoton.GetComponent<Button>().interactable;
40     }
41 }
42
43
44

```

La segunda línea en la que nos vamos a fijar es:

`private GameObject btnPunteroBoton;`

La tercera palabra con la que nos encontramos es el nombre que le damos a esta variable.

- El nombre debe ser significativo.
- En la UT1 si explicó como debía ser la nomenclatura de estos nombres, prefijos, mayúsculas o minúsculas, etc...
- En este caso, la variable empieza por "btn", al ser un puntero a un botón, aunque podría haber empezado por ptr, al ser un puntero de tal forma que quedaría:

`private GameObject ptrPunteroBoton;`

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Puntero a Componentes

```

1  /*
2   * Date: 16 de Agosto de 2020.
3   * Class: ScrBoton.
4   * Description: En esta clase podremos leer
5   * y escribir los distintos parámetros de con-
6   * figuración de los botones.
7   * Author: Raquel Rojo y Mario Santos.
8   * Palomeras Vallecas - Villablanca
9   */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14 using UnityEngine.UI;
15
16 public class ScrBoton : MonoBehaviour
17 {
18     private GameObject btnPunteroBoton;
19
20     public ScrBoton(string nombre)
21     {
22         //Buscamos el GameObject al que queremos
23         //apuntar por el nombre
24         btnPunteroBoton = GameObject.Find(nombre);
25     }
26
27     //Método para ver si el Botón está activo o no.
28     public void GetInteractivo()
29     {
30     }
31
32     Debug.Log(btnPunteroBoton.GetComponent<Button>().interactable);
33
34 }
35
36 //Método para activar/desactivar el botón.
37 public void SetInteractivo()
38 {
39     btnPunteroBoton.GetComponent<Button>().interactable =
40     !btnPunteroBoton.GetComponent<Button>().interactable;
41 }
42
43 }
44

```

La tercera línea en la que nos debemos fijar es:

btnPunteroBoton = GameObject.Find(nombre);

Esta es la sentencia que se asocia el puntero al GameObject. La primera palabra con la que nos encontramos es la variable en la que vamos a guardar el puntero a nuestro GameObject. En este caso, la variable es la que hemos declarado como **private** y hemos denominado **btnPunteroBoton**.

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Puntero a Componentes

```

1  /*
2   * Date: 16 de Agosto de 2020.
3   * Class: ScrBoton.
4   * Description: En esta clase podremos leer
5   * y escribir los distintos parámetros de con-
6   * figuración de los botones.
7   * Author: Raquel Rojo y Mario Santos.
8   * Palomeras Vallecas - Villablanca
9   */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14 using UnityEngine.UI;
15
16 public class ScrBoton : MonoBehaviour
17 {
18     private GameObject btnPunteroBoton;
19
20     public ScrBoton(string nombre)
21     {
22         //Buscamos el GameObject al que queremos
23         //apuntar por el nombre
24         btnPunteroBoton = GameObject.Find(nombre);
25     }
26
27     //Método para ver si el Botón está activo o no.
28     public void GetInteractivo()
29     {
30
31         Debug.Log(btnPunteroBoton.GetComponent<Button>().interactable);
32     }
33
34     //Método para activar/desactivar el botón.
35     public void SetInteractivo()
36     {
37         btnPunteroBoton.GetComponent<Button>().interactable =
38             !btnPunteroBoton.GetComponent<Button>().interactable;
39     }
40 }
41
42
43
44

```

La tercera línea en la que nos debemos fijar es:

btnPunteroBoton = GameObject.Find(nombre);

A continuación nos encontramos la asignación del puntero que nos devuelve la sentencia.

GameObject.Find();

Esta sentencia es la que vamos a utilizar para localizar un determinado GameObject en nuestra estructura Hierarchy. Fíjate que debemos de utilizar la clase **GameObject**, que viene heredada con **MonoBehaviour**.

En puntos anteriores, se ha comentado que tenemos tres formas de realizar esta búsqueda:

- **GameObject.Find()**
- **GameObject FindGameObjectWithTag()**
- **GameObject FindObjectOfType()**

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Puntero a Componentes

```

1  /*
2   * Date: 16 de Agosto de 2020.
3   * Class: ScrBoton.
4   * Description: En esta clase podremos leer
5   * y escribir los distintos parámetros de con-
6   * figuración de los botones.
7   * Author: Raquel Rojo y Mario Santos.
8   * Palomeras Vallecas - Villablanca
9   */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14 using UnityEngine.UI;
15
16 public class ScrBoton : MonoBehaviour
17 {
18     private GameObject btnPunteroBoton;
19
20     public ScrBoton(string nombre)
21     {
22         //Buscamos el GameObject al que queremos
23         //apuntar por el nombre
24
25         btnPunteroBoton = GameObject.Find(nombre);
26     }
27
28     //Método para ver si el Botón está activo o no.
29     public void GetInteractivo()
30     {
31
32         Debug.Log(btnPunteroBoton.GetComponent<Button>().interactable);
33     }
34
35
36     //Método para activar/desactivar el botón.
37     public void SetInteractivo()
38     {
39         btnPunteroBoton.GetComponent<Button>().interactable =
40             !btnPunteroBoton.GetComponent<Button>().interactable;
41     }
42
43 }
44

```

Método1: GameObject.Find()

Con esta sentencia encontramos un GameObject en nuestra Hierarchy, por su nombre. Si nos fijamos, en el código de la clase ScrBoton, entre los paréntesis de esta sentencia, hemos escrito "nombre"

GameObject.Find(nombre)

En la cabecera de nuestro constructor, hemos declarado esta variable de tipo string y se pasa por valor.

```

Public ScrBoton(string nombre){
}

```

De tal manera que con la sentencia GameObject.Find(nombre), haremos un puntero al GameObject cuyo nombre pasemos por valor.

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Puntero a Componentes

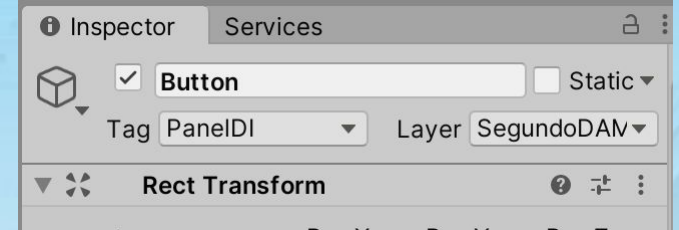
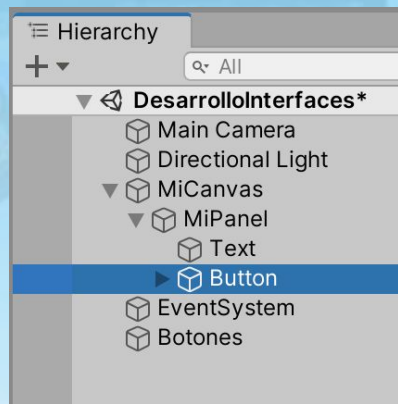
```

1  /*
2   * Date: 16 de Agosto de 2020.
3   * Class: ScrBoton.
4   * Description: En esta clase podremos leer
5   * y escribir los distintos parámetros de con-
6   * figuración de los botones.
7   * Author: Raquel Rojo y Mario Santos.
8   * Palomeras Vallecas - Villablanca
9   */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14 using UnityEngine.UI;
15
16 public class ScrBoton : MonoBehaviour
17 {
18     private GameObject btnPunteroBoton;
19
20     public ScrBoton(string nombre)
21     {
22         //Buscamos el GameObject al que queremos
23         //apuntar por el nombre
24         btnPunteroBoton = GameObject.Find(nombre);
25     }
26
27     //Método para ver si el Botón está activo o no.
28     public void GetInteractivo()
29     {
30     }
31
32     Debug.Log(btnPunteroBoton.GetComponent<Button>().interactable);
33
34 }
35
36 //Método para activar/desactivar el botón.
37 public void SetInteractivo()
38 {
39     btnPunteroBoton.GetComponent<Button>().interactable =
40     !btnPunteroBoton.GetComponent<Button>().interactable;
41 }
42
43 }
44

```

Método 2: GameObject.FindGameObjectWithTag()

Con esta sentencia encontramos un GameObject en nuestra Hierarchy, por su Tag (etiqueta). En este caso, le pasamos el parámetro por valor, del nombre de la etiqueta.



`btnPunteroBoton = GameObject.FindGameObjectWithTag("PanelDI");`

```

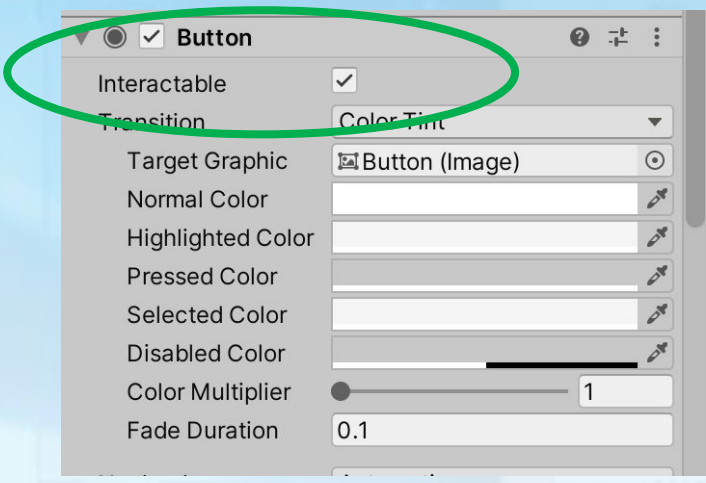
1  /*
2   * Date: 16 de Agosto de 2020.
3   * Class: ScrBoton.
4   * Description: En esta clase podremos leer
5   * y escribir los distintos parámetros de con-
6   * figuración de los botones.
7   * Author: Raquel Rojo y Mario Santos.
8   * Palomeras Vallecas - Villablanca
9   */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14 using UnityEngine.UI;
15
16 public class ScrBoton : MonoBehaviour
17 {
18     private GameObject btnPunteroBoton;
19
20     public ScrBoton(string nombre)
21     {
22         //Buscamos el GameObject al que queremos
23         //apuntar por el nombre
24
25         btnPunteroBoton = GameObject.Find(nombre);
26     }
27
28     //Método para ver si el Botón está activo o no.
29     public void GetInteractivo()
30     {
31
32         Debug.Log(btnPunteroBoton.GetComponent<Button>().interactable);
33     }
34
35
36     //Método para activar/desactivar el botón.
37     public void SetInteractivo()
38     {
39         btnPunteroBoton.GetComponent<Button>().interactable =
40         !btnPunteroBoton.GetComponent<Button>().interactable;
41     }
42
43 }
44

```

Método 3: GameObject.FindObjectOfType()

Con esta sentencia encontramos un GameObject en nuestra Hierarchy, por su Type. En este caso, le pasamos el parámetro por valor, del tipo de GameObject, button.

En este caso, cambiamos la forma de trabajar ya que no vamos a hacer referencia al GameObject, vamos a hacer referencia al componente del GameObject en el que se encuentra el parámetro que queremos modificar. En nuestro caso es:



Button y el parámetro a modificar "Interactable"

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Puntero a Componentes

```

1  /*
2   * Date: 16 de Agosto de 2020.
3   * Class: ScrBoton.
4   * Description: En esta clase podremos leer
5   * y escribir los distintos parámetros de con-
6   * figuración de los botones.
7   * Author: Raquel Rojo y Mario Santos.
8   * Palomeras Vallecas - Villablanca
9   */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14 using UnityEngine.UI;
15
16 public class ScrBoton : MonoBehaviour
17 {
18     private GameObject btnPunteroBoton;
19
20     public ScrBoton(string nombre)
21     {
22         //Buscamos el GameObject al que queremos
23         //apuntar por el nombre
24         btnPunteroBoton = GameObject.Find(nombre);
25     }
26
27     //Método para ver si el Botón está activo o no.
28     public void GetInteractivo()
29     {
30         Debug.Log(btnPunteroBoton.GetComponent<Button>().interactable);
31     }
32
33     //Método para activar/desactivar el botón.
34     public void SetInteractivo()
35     {
36         btnPunteroBoton.GetComponent<Button>().interactable =
37             !btnPunteroBoton.GetComponent<Button>().interactable;
38     }
39 }
40
41
42
43
44

```

Método 3: GameObject.FindObjectOfType()

Teniendo esto claro, la sentencia quedaría de la siguiente manera:

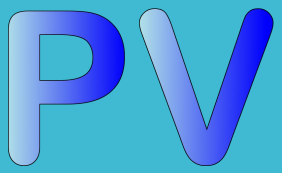
```
private Button btnPunteroBoton;
```

```

Public ScrBoton(){
    btnPunteroBoton = (Button)FindObjectOfType(typeof(Button));
}

```

Fijarse en cómo ha cambiado la declaración de la variable btnPunteroBoton, que ha pasado de ser de tipo GameObject a tipo Button.



I.E.S PALOMERAS
VALLECAS

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Puntero a Componentes



I.E.S. VILLABLANCA

```
1  /*
2   * Date: 16 de Agosto de 2020.
3   * Class: ScrBoton.
4   * Description: En esta clase podremos leer
5   * y escribir los distintos parámetros de con-
6   * figuración de los botones.
7   * Author: Raquel Rojo y Mario Santos.
8   * Palomeras Vallecas - Villablanca
9   */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14 using UnityEngine.UI;
15
16 public class ScrBoton : MonoBehaviour
17 {
18     private GameObject btnPunteroBoton;
19
20     public ScrBoton(string nombre)
21     {
22         //Buscamos el GameObject al que queremos
23         //apuntar por el nombre
24         btnPunteroBoton = GameObject.Find(nombre);
25     }
26
27     //Método para ver si el Botón está activo o no.
28     public void GetInteractivo()
29     {
30
31         Debug.Log(btnPunteroBoton.GetComponent<Button>().interactable);
32     }
33
34     //Método para activar/desactivar el botón.
35     public void SetInteractivo()
36     {
37         btnPunteroBoton.GetComponent<Button>().interactable =
38             !btnPunteroBoton.GetComponent<Button>().interactable;
39     }
40 }
41
42
43
44
```

Existen otras sentencias para hacer otros tipos de búsquedas como todos los componentes que tengan una determinada Tag

`GameObject.FindGameObjectsWithTag();`

O todos los que son de un determinado tipo

`GameObject.FindObjectsOfType();`

De los que hemos visto, podéis consultar la ayuda de Unity en:

- `GameObject.Find()`
<https://docs.unity3d.com/2020.2/Documentation/ScriptReference/GameObject.Find.html>
- `GameObject.FindGameObjectsWithTag()`
<https://docs.unity3d.com/2020.2/Documentation/ScriptReference/GameObject.FindWithTag.html>
- `GameObject.FindObjectsOfType`
<https://docs.unity3d.com/2020.2/Documentation/ScriptReference/GameObject.FindObjectsOfType.html>

Profesores: Raquel Rojo y Mario Santos.

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Acceso a los componentes y a sus parámetros.

```

1  /*
2   * Date: 16 de Agosto de 2020.
3   * Class: ScrBoton.
4   * Description: En esta clase podremos leer
5   * y escribir los distintos parámetros de con-
6   * figuración de los botones.
7   * Author: Raquel Rojo y Mario Santos.
8   * Palomeras Vallecas - Villablanca
9   */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14 using UnityEngine.UI;
15
16 public class ScrBoton : MonoBehaviour
17 {
18     private GameObject btnPunteroBoton;
19
20     public ScrBoton(string nombre)
21     {
22         //Buscamos el GameObject al que queremos
23         //apuntar por el nombre.
24
25         btnPunteroBoton = GameObject.Find(nombre);
26     }
27
28     //Método para ver si el Botón está activo o no.
29     public void GetInteractivo()
30     {
31
32         Debug.Log(btnPunteroBoton.GetComponent<Button>().interactable);
33     }
34
35     //Método para activar/desactivar el botón.
36     public void SetInteractivo()
37     {
38         btnPunteroBoton.GetComponent<Button>().interactable =
39             !btnPunteroBoton.GetComponent<Button>().interactable;
40     }
41 }
42
43
44

```

La última sentencia que vamos a analizar es

```
Debug.Log(btnPunteroBoton.GetComponent<Button>().interactable);
```

Esta es una unión o anidamiento de dos sentencias:

Debug.Log()

<https://docs.unity3d.com/2020.2/Documentation/ScriptReference/Debug.Log.html>

GetComponent<>()

<https://docs.unity3d.com/2020.2/Documentation/ScriptReference/GameObject.GetComponent.html>

Debug.Log() es una sentencia que ya hemos utilizado a lo largo de este curso y que nos permite sacar información por la ventana “Console” de nuestro entorno Unity.

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Acceso a los componentes y a sus parámetros.

```

1  /*
2   * Date: 16 de Agosto de 2020.
3   * Class: ScrBoton.
4   * Description: En esta clase podremos leer
5   * y escribir los distintos parámetros de con-
6   * figuración de los botones.
7   * Author: Raquel Rojo y Mario Santos.
8   * Palomeras Vallecas - Villablanca
9   */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14 using UnityEngine.UI;
15
16 public class ScrBoton : MonoBehaviour
17 {
18     private GameObject btnPunteroBoton;
19
20     public ScrBoton(string nombre)
21     {
22         //Buscamos el GameObject al que queremos
23         //apuntar por el nombre.
24
25         btnPunteroBoton = GameObject.Find(nombre);
26     }
27
28     //Método para ver si el Botón está activo o no.
29     public void GetInteractivo()
30     {
31
32         Debug.Log(btnPunteroBoton.GetComponent<Button>().interactable);
33     }
34
35     //Método para activar/desactivar el botón.
36     public void SetInteractivo()
37     {
38         btnPunteroBoton.GetComponent<Button>().interactable =
39             !btnPunteroBoton.GetComponent<Button>().interactable;
40     }
41 }
42
43
44

```

`GetComponent<>()` es la instrucción que vamos a utilizar para entrar en un componente del `GameObject` al que hemos referenciado. Veamos su uso más despacio.

- `GetComponent<>()` es un método de la clase `GameObject`. Cuando hemos creado nuestra variable puntero la hemos declarado como `GameObject`, por eso, esta sentencia tiene este formato:

`GameObject.GetComponent<>()`

- Que en nuestro caso sería...

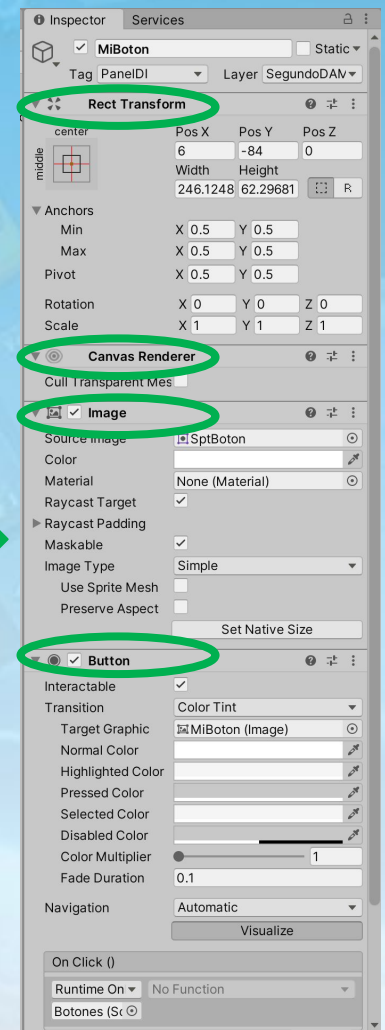
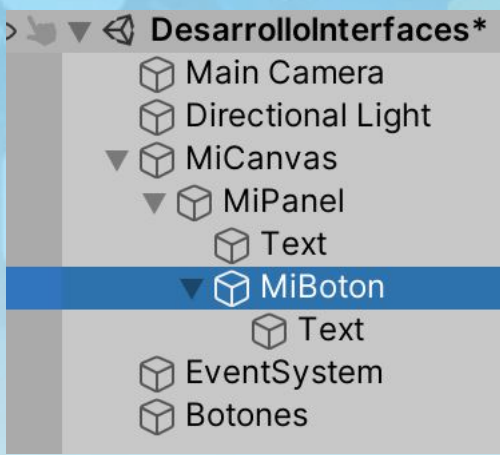
`btnPunteroBoton.GetComponent<>()`

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Acceso a los componentes y a sus parámetros.

Vamos a fijarnos un momento en la ventana Inspector y vamos a ver los componentes de nuestro GameObject MiBotón



Dentro del GameObject MiBoton tenemos diferentes componentes:

- Rect Transform
- Canvas Renderer
- Image
- Button

Y dentro de estos componentes, tenemos diferentes parámetros:

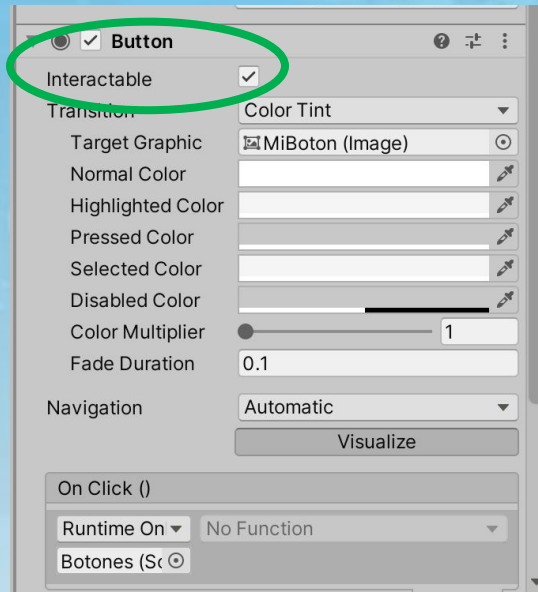
- Rect Transform
 - Anchors
 - Pivot
 - Rotation
 - Scale
- Canvas Renderer
 - Cull Transparent Mesh
- Image
 - Source Image
 - Color
 - Etc.

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Acceso a los componentes y a sus parámetros.

En nuestro caso, primero accederemos al componente **Button** y luego a su parámetro «interactable».



Para acceder al componente Button

```
btnPunteroBoton.GetComponent<Button>()
```

Con esta sentencia estamos diciendo a Unity que vamos a trabajar sobre el componente Button del GameObject que tenemos referenciado con nuestra variable btnPunteroBoton.

Por último, para acceder a la propiedad, en este caso interactable, ponemos un punto y el nombre de la propiedad.

```
btnPunteroBoton.GetComponent<Button>().interactable
```

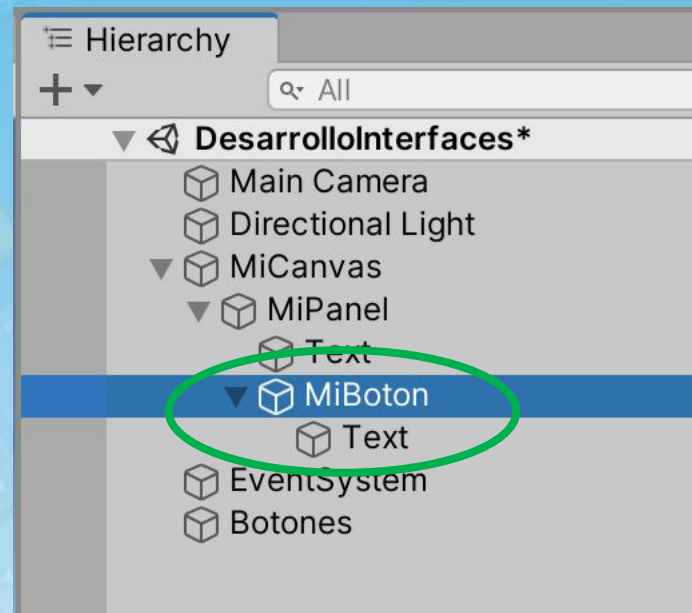
De esta manera, podremos acceder y modificar desde nuestros Scripts, cualquier parámetro público de los GameObjects.

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Acceso a los componentes y a sus parámetros.

Ahora bien, visto el método GetComponent<>() y sabiendo que el GameObject “Button” tiene como hijo otro GameObject de tipo “Text”.



¿Cómo podemos acceder a un parámetro que se encuentra en el GameObject hijo “Text”?

Para este tipo de casos, la clase GameObject dispone del método

GetComponentInChildren<>(); <https://docs.unity3d.com/2020.2/Documentation/ScriptReference/GameObject.GetComponentInChildren.html>

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Acceso a los componentes y a sus parámetros.

La forma de uso de este método es igual que la anterior, pero ahora en pondremos entre <> el nombre del GameObject. Si por ejemplo quisiera cambiar el texto del botón a “Pulsar”, escribiría la siguiente sentencia.



`btnPunteroBoton.GetComponentInChildren<Text>().text = “Pulsar”;`



Haremos notar que cuando el GameObject se denomina igual que el parámetro a configurar, esto puede llevar a la confusión si no se tiene muy claro el concepto de GameObject, Componente y Parámetro. Es importante y nos facilitará el trabajo (workflow), renombrar los GameObjects.

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Acceso a los componentes y a sus parámetros.

Esta última instrucción es lo que se denomina en programación, un interruptor o switch.

```
btnPunteroBoton.GetComponent<Button>().interactable =
!btnPunteroBoton.GetComponent<Button>().interactable
```

La clave es el signo de exclamación (significa negación), situado a continuación del igual. Cada vez que se ejecuta este método provoca que se guarde en el parámetro interactable siempre el valor contrario al que tiene.

Este es el Script ScrBoton, que como hemos dicho con anterioridad es nuestra Clase para trabajar con el botón. Veamos ahora como llamar a esta clase desde un script asociado a un GameObject.

```
1  /*
2   * Date: 16 de Agosto de 2020.
3   * Class: ScrBoton.
4   * Description: En esta clase podremos leer
5   * y escribir los distintos parámetros de con-
6   * figuración de los botones.
7   * Author: Raquel Rojo y Mario Santos.
8   * Palomeras Vallecas - Villablanca
9   */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14 using UnityEngine.UI;
15
16 public class ScrBoton : MonoBehaviour
17 {
18     private GameObject btnPunteroBoton;
19
20     public ScrBoton(string nombre)
21     {
22         //Buscamos el GameObject al que queremos
23         //apuntar por el nombre.
24
25         btnPunteroBoton = GameObject.Find(nombre);
26     }
27
28     //Método para ver si el Botón está activo o no.
29     public void GetInteractivo()
30     {
31
32         Debug.Log(btnPunteroBoton.GetComponent<Button>().interactable);
33     }
34
35     //Método para activar/desactivar el botón.
36     public void SetInteractivo()
37     {
38         btnPunteroBoton.GetComponent<Button>().interactable =
39         !btnPunteroBoton.GetComponent<Button>().interactable;
40     }
41 }
42
43
44
```

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

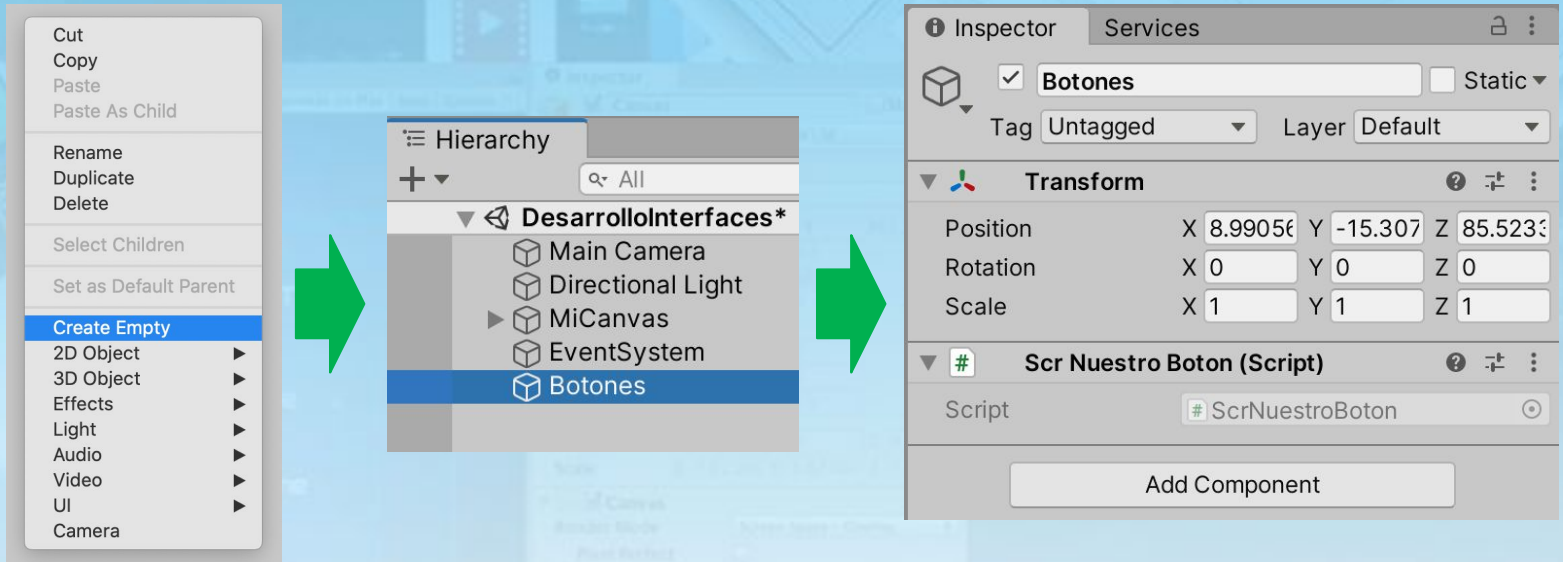
Trabajando con nuestras clases.

```

1  /*
2   * Date: 16 de Agosto de 2020.
3   * Class: ScrNuestroBoton.
4   * Description: Desde este script, creamos una
5   * variable de tipo ScrBoton y llamamos a sus
6   * métodos.
7   * Author: Raquel Rojo y Mario Santos.
8   * Palomeras Vallecas - Villablanca
9   */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14
15 public class ScrNuestroBoton : MonoBehaviour
16 {
17     //declaramos una variable de tipo ScrBoton
18     private ScrBoton btnNuestroBoton;
19
20     public void Prueba()
21     {
22         //Llamamos al constructor de la clase ScrBoton.
23         btnNuestroBoton = new ScrBoton("MiBoton");
24         //Llamamos a los métodos GetInteractivo y
25         //SetInteractivo.
26         btnNuestroBoton.GetInteractivo();
27         btnNuestroBoton.SetInteractivo();
28     }
29
30 }

```

Lo primero que hemos hecho con este Script es asociarlo con un GameObject te tipo Empty, que hemos denominado “Botones”. Como se ha dicho con anterioridad en este curso, para que se ejecute un Script, debe estar asociado a un GameObject.



UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Trabajando con nuestras clases.

Lo primero que hacemos en nuestro Script ScrNuestroBoton, es crear una variable, **privada**, del tipo de la clase que hemos creado con anterioridad.

```
Private ScrBoton btnNuestroBoton;
```

Lo segundo que vamos a hacer es instanciar esa clase y para ello utilizamos la instrucción:

```
btnNuestroBoton = new ScrBoton("MiBoton");
```

En esta instrucción, llamamos al constructor de nuestra clase ScrBoton y le pasamos como parámetro el nombre del botón que queremos referenciar. Esta forma de trabajar es clásica y, aún siendo perfectamente válida, Unity está cambiando la forma de crear una variable de una clase. La sentencia que quiere implantar es:

```
btnNuestroBoton = gameObject.AddComponent(typeof(ScrBoton)) as ScrBoton;
```

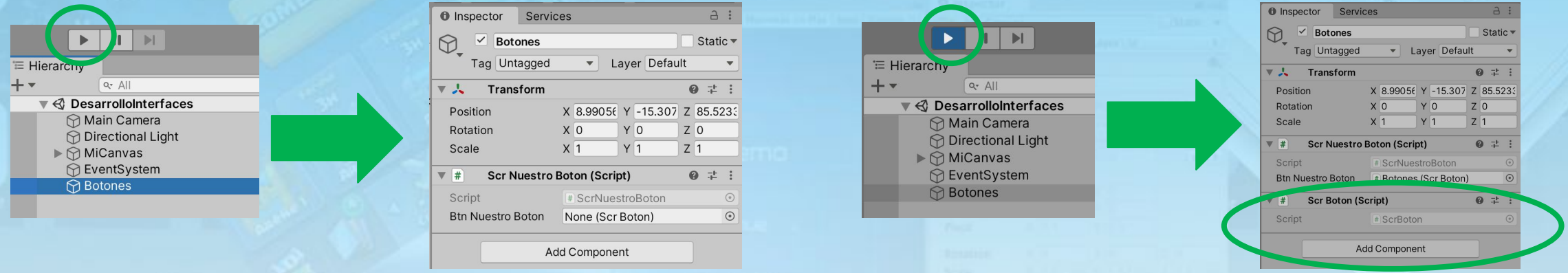
```
1  /*
2   * Date: 16 de Agosto de 2020.
3   * Class: ScrNuestroBoton.
4   * Description: Desde este script, creamos una
5   * variable de tipo ScrBoton y llamamos a sus
6   * métodos.
7   * Author: Raquel Rojo y Mario Santos.
8   * Palomeras Vallecas - Villablanca
9   */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14
15 public class ScrNuestroBoton : MonoBehaviour
16 {
17     //declaramos una variable de tipo ScrBoton
18     private ScrBoton btnNuestroBoton;
19
20     public void Prueba()
21     {
22         //Llamamos al constructor de la clase ScrBoton.
23         btnNuestroBoton = new ScrBoton("MiBoton");
24         //Llamamos a los métodos GetInteractivo y
25         //SetInteractivo.
26         btnNuestroBoton.GetInteractivo();
27         btnNuestroBoton.SetInteractivo();
28     }
29 }
30
```

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales: Componentes contenedores básicos.


Trabajando con nuestras clases.

```
btnNuestroBoton = gameObject.AddComponent(typeof(ScrBoton)) as ScrBoton;
```

Veamos como funciona esta sentencia. Lo que hace `gameObject.AddComponent()` es **añadir** un **componente** especificado entre los paréntesis, en este caso la clase `ScrBoton`, al `GameObject` como componente, únicamente en modo **ejecución**.



En nuestro caso, cada vez que pulsamos el botón de nuestro panel es cuando inicializamos nuestra variable del tipo de nuestra clase `ScrBoton`. Esto significa que, cada vez que pulsemos el botón, se nos cargará un nuevo componente de esta variable. **Esto es un error de programación que debemos de corregir.**

 **Estudiar el paquete `UT2_Ejemplo.unitypackage`**

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Trabajando con nuestras clases.

```
btnNuestroBoton = gameObject.AddComponent(typeof(ScrBoton)) as ScrBoton;
```

Para solucionar este problema, tenemos la sentencia `Destroy()`, que lo que hace es eliminar, en proceso de ejecución, cualquier `GameObject` con sus componentes de memoria.

En este caso, nuestra variable `btnNuestroBoton`, está apuntando directamente al componente que hemos creado, así que si ponemos la siguiente sentencia:

```
Destroy(btnNuestroBoton);
```

Se eliminará el componente que hemos añadido.

El código quedaría como muestra la imagen.



Estudiar el paquete `UT2_Ejemplo.unitypackage`

Profesores: Raquel Rojo y Mario Santos.

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Trabajando con nuestras clases.

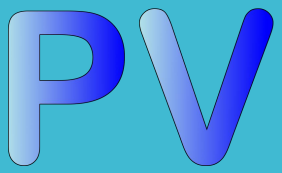
```
btnNuestroBoton = gameObject.AddComponent(typeof(ScrBoton)) as ScrBoton;
```

```
1  /*
2   * Date: 16 de Agosto de 2020.
3   * Class: ScrNuestroBoton.
4   * Description: Desde este script, creamos una
5   * variable de tipo ScrBoton y llamamos a sus
6   * métodos.
7   * Author: Raquel Rojo y Mario Santos.
8   * Palomeras Vallecas - Villablanca
9   */
10
11 using System.Collections;
12 using System.Collections.Generic;
13 using UnityEngine;
14
15 public class ScrNuestroBoton : MonoBehaviour
16 {
17     //declaramos una variable de tipo ScrBoton
18     public ScrBoton btnNuestroBoton;
19
20     public void Prueba()
21     {
22         //Llamamos al constructor de la clase ScrBoton.
23
24         btnNuestroBoton = gameObject.AddComponent(typeof(ScrBoton)) as ScrBoton;
25
26         //Llamamos a los métodos GetInteractivo y
27         //SetInteractivo.
28
29         btnNuestroBoton.GetInteractivo();
30         btnNuestroBoton.SetInteractivo();
31
32         //Destruimos el componente añadido.
33
34         Destroy(btnNuestroBoton);
35     }
36 }
37
```

Al trabajar con esta instrucción **no** podemos pasar ningún valor al constructor de nuestra clase, así que habría que estudiar la mejor forma de crear un puntero al GameObject con el que queremos trabajar, por ejemplo con un GameObject de tipo Input Field que será estudiado en la próxima UT.

 Estudiar el paquete UT2_Ejemplo.unitypackage

Profesores: Raquel Rojo y Mario Santos.



I.E.S. PALOMERAS
VALLECAS

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.



I.E.S. VILLABLANCA

Trabajando con Teclado y creando Atajos de Teclado.

Podemos utilizar combinaciones de teclas, botones del ratón, Joystick o combinaciones entre todas para trabajar con nuestras interfaz de usuario.

Para poder trabajar desde el teclado, ratón o Joystick, tenemos seis sentencias:

```
Input.GetKey(KeyCode.)  
Input.GetKeyDown(KeyCode.)  
Input.GetKeyUp(KeyCode.)  
Input.GetButton("")  
Input.GetButtonDown("")  
Input.GetButtonUp("")
```

Las tres primeras se traducen en:

```
Input.GetKey() □ Pulsa una tecla y se puede dejar pulsada.  
Input.GetKeyDown() □ Mantienes pulsada la tecla.  
Input.GetKeyUp() □ Se suelta la tecla.
```

Profesores: Raquel Rojo y Mario Santos.



I.E.S. PALOMERAS
VALLECAS

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.



I.E.S. VILLABLANCA

Trabajando con Teclado y creando Atajos de Teclado.

`Input.GetKey()` □ Pulsa una tecla y se puede dejar pulsada.

`Input.GetKeyDown()` □ Mantiene pulsada la tecla.

`Input.GetKeyUp()` □ Se suelta la tecla.

Lo importante de estas tres sentencias es el comando que escribimos entre los paréntesis:

`KeyCode`

Acompañando a este comando va un “.”, a la derecha. Observar que cuando escribís el “.”, aparecen las diferentes combinaciones o teclas que podéis utilizar.

Como ejemplo para verlo mejor, vamos a suponer que queremos que cuando pulsemos la “A”, se active o desactive nuestro panel. Para hacer esto, tenemos que conocer el parámetro

`GameObject.SetActive(bool)`

Con este parámetro podemos visualizar/ocultar cualquier `GameObject` pasándole entre los paréntesis `true/false`.

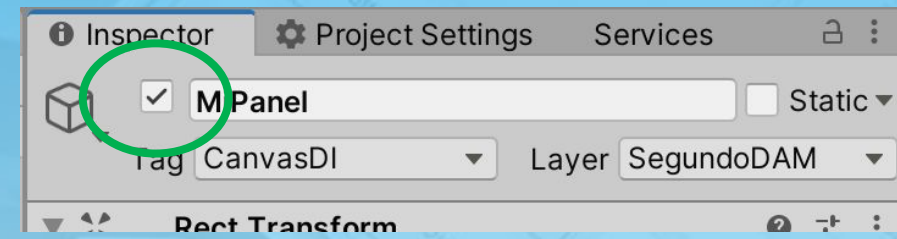
Profesores: Raquel Rojo y Mario Santos.

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Trabajando con Teclado y creando Atajos de Teclado.

GameObject.SetActive(false/true);



El código resultante es:

```

/*
 * Date: 16 de Agosto de 2020.
 * Class: AtajosDeTeclado.
 * Description: Utilización de teclado con
 * GetKey y con GetButton
 * Author: Raquel Rojo y Mario Santos.
 * Palomeras Vallecas - Villablanca
 */

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AtajosDeTeclado : MonoBehaviour
{
    public GameObject ptrPanel;

    // Start es llamada antes de que se actualice el primer frame.
    void Start()
    {
        ptrPanel = GameObject.Find("MiPanel");
    }

    // Update es llamada por frame.
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.A))
        {
            if (ptrPanel.activeInHierarchy == true)
            {
                ptrPanel.SetActive(false);
            }
            else
            {
                ptrPanel.SetActive(true);
            }
        }
    }
}

```

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Trabajando con Teclado y creando Atajos de Teclado.

```

/*
 * Date: 16 de Agosto de 2020.
 * Class: AtajosDeTeclado.
 * Description: Utilización de teclado con
 * GetKey y con GetButton
 * Author: Raquel Rojo y Mario Santos.
 * Palomerias Vallecas - Villablanca
 */

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AtajosDeTeclado : MonoBehaviour
{
    private GameObject ptrPanel;

    // Start es llamada antes de que se actualice el primer frame.
    void Start()
    {
        ptrPanel = GameObject.Find("MiPanel");
    }

    // Update es llamada por frame.
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.A))
        {
            if (ptrPanel.activeInHierarchy == true)
            {
                ptrPanel.SetActive(false);
            }
            else
            {
                ptrPanel.SetActive(true);
            }
        }
    }
}

```

Veamos el código...

Como hemos visto con anterioridad, hemos declarado una variable **privada** de tipo **GameObject** que vamos a utilizar como puntero al Panel "MiPanel". Esta variable se denomina "ptrPanel".

Es interesante observar que la declaración del puntero al **GameObject**, lo estamos haciendo dentro del método **void Start()**. Este método es **proporcionado** por Unity de forma **predeterminada** al crear el Script y se va a ejecutar siempre, **una única vez**, en el momento en el que es cargado en la escena el **GameObject** al que está asociado.

Es un método que suele ser muy utilizado para la **inicialización** de variables, clases, etc. Veremos en una UT posterior, que aún existe un método que se ejecuta, incluso, antes que este.



Utilizar la nomenclatura para la declaración de variables, Clases y métodos explicada en UT1

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Trabajando con Teclado y creando Atajos de Teclado.

```
/*
 * Date: 16 de Agosto de 2020.
 * Class: AtajosDeTeclado.
 * Description: Utilización de teclado con
 * GetKey y con GetButton
 * Author: Raquel Rojo y Mario Santos.
 * Palomerias Vallecas - Villablanca
 */

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AtajosDeTeclado : MonoBehaviour
{
    private GameObject ptrPanel;

    // Start es llamada antes de que se actualice el primer frame.
    void Start()
    {
        ptrPanel = GameObject.Find("MiPanel");
    }

    // Update es llamada por frame.
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.A))
        {
            if (ptrPanel.activeInHierarchy == true)
            {
                ptrPanel.SetActive(false);
            }
            else
            {
                ptrPanel.SetActive(true);
            }
        }
    }
}
```

Veamos el código...

Lo siguiente con lo que nos encontramos es con una sentencia de control if(), anidada.

El primer if() que nos encontramos es:

```
if (Input.GetKeyDown(KeyCode.A))
{
}
```

La sentencia Input.GetKeyDown significa que vamos a controlar la pulsación de una tecla y la sentencia KeyCode.A, significa que es la pulsación de la tecla "A".

GetKeyDown: <https://docs.unity3d.com/ScriptReference/Input.GetKeyDown.html>

KeyCode: <https://docs.unity3d.com/ScriptReference/KeyCode.html>

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Trabajando con Teclado y creando Atajos de Teclado.

```

/*
 * Date: 16 de Agosto de 2020.
 * Class: AtajosDeTeclado.
 * Description: Utilización de teclado con
 * GetKey y con GetButton
 * Author: Raquel Rojo y Mario Santos.
 * Palomeras Vallecas - Villablanca
 */

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AtajosDeTeclado : MonoBehaviour
{
    private GameObject ptrPanel;

    // Start es llamada antes de que se actualice el primer frame.
    void Start()
    {
        ptrPanel = GameObject.Find("MiPanel");
    }

    // Update es llamada por frame.
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.A))
        {
            if (ptrPanel.activeInHierarchy == true)
            {
                ptrPanel.SetActive(false);
            }
            else
            {
                ptrPanel.SetActive(true);
            }
        }
    }
}

```

Veamos el código...

El segundo if que nos encontramos es:

```

if (ptrPanel.activeInHierarchy == true)
{
}

```

El parámetro **activeInHierarchy**, nos devuelve si el GameObject, está visible en la ventana Hierarchy y en la escena. En nuestro caso, es justo lo que estamos haciendo, visibilizar/ocultar el panel.

Para visibilizar/ocultar un GameObject, utilizamos el parámetro **SetActive**.

```
ptrPanel.SetActive(true||false)
```

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Trabajando con Teclado y creando Atajos de Teclado.

```

/*
 * Date: 16 de Agosto de 2020.
 * Class: AtajosDeTeclado.
 * Description: Utilización de teclado con
 * GetKey y con GetButton
 * Author: Raquel Rojo y Mario Santos.
 * Palomeras Vallecas - Villablanca
 */

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AtajosDeTeclado : MonoBehaviour
{
    private GameObject ptrPanel;

    // Start es llamada antes de que se actualice el primer frame.
    void Start()
    {
        ptrPanel = GameObject.Find("MiPanel");
    }

    // Update es llamada por frame.
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.A))
        {
            if (ptrPanel.activeInHierarchy == true)
            {
                ptrPanel.SetActive(false);
            }
            else
            {
                ptrPanel.SetActive(true);
            }
        }
    }
}

```

Veamos el código...

Este if anidado, lo hemos escrito dentro del método **void Update()**. Al igual que el método void Start(), viene por defecto cuando creamos un Script en nuestro proyecto.

En el caso del método Update(), se va a ejecutar una vez por cada frame. Es decir, es como un bucle que se va a estar repitiendo todo el tiempo de ejecución. Esto nos va a permitir trabajar de una forma muy cómoda situaciones como la que estamos viendo con nuestro ejemplo, en el que tenemos que estar preguntando, constantemente, si la tecla "A", ha sido pulsada.

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Trabajando con Teclado y creando Atajos de Teclado.

```

/*
 * Date: 16 de Agosto de 2020.
 * Class: AtajosDeTeclado.
 * Description: Utilización de teclado con
 * GetKey y con GetButton
 * Author: Raquel Rojo y Mario Santos.
 * Palomeras Vallecas - Villablanca
 */

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AtajosDeTeclado : MonoBehaviour
{
    private GameObject ptrPanel;

    // Start es llamada antes de que se actualice el primer frame.
    void Start()
    {
        ptrPanel = GameObject.Find("MiPanel");
    }

    // Update es llamada por frame.
    void Update()
    {
        if (Input.GetButtonDown("DI"))
        {
            if (ptrPanel.activeInHierarchy == true)
            {
                ptrPanel.SetActive(false);
            }
            else
            {
                ptrPanel.SetActive(true);
            }
        }
    }
}

```

En este caso, es el mismo código pero hemos cambiado la sentencia del primer if().

```

{
    if (Input.GetButtonDown("DI"))
    {

```

Utilizamos el método **GetButtonDown()** y le pasamos como parámetro de entrada "DI".

GetButtonDown() se utiliza para trabajar con **teclas virtuales** que nos proporciona Unity. Estas teclas virtuales vienen por defecto y, además, podemos crear las que nos hagan falta.

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Trabajando con Teclado y creando Atajos de Teclado.

```

/*
 * Date: 16 de Agosto de 2020.
 * Class: AtajosDeTeclado.
 * Description: Utilización de teclado con
 * GetKey y con GetButton
 * Author: Raquel Rojo y Mario Santos.
 * Palomeras Vallecas - Villablanca
 */

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

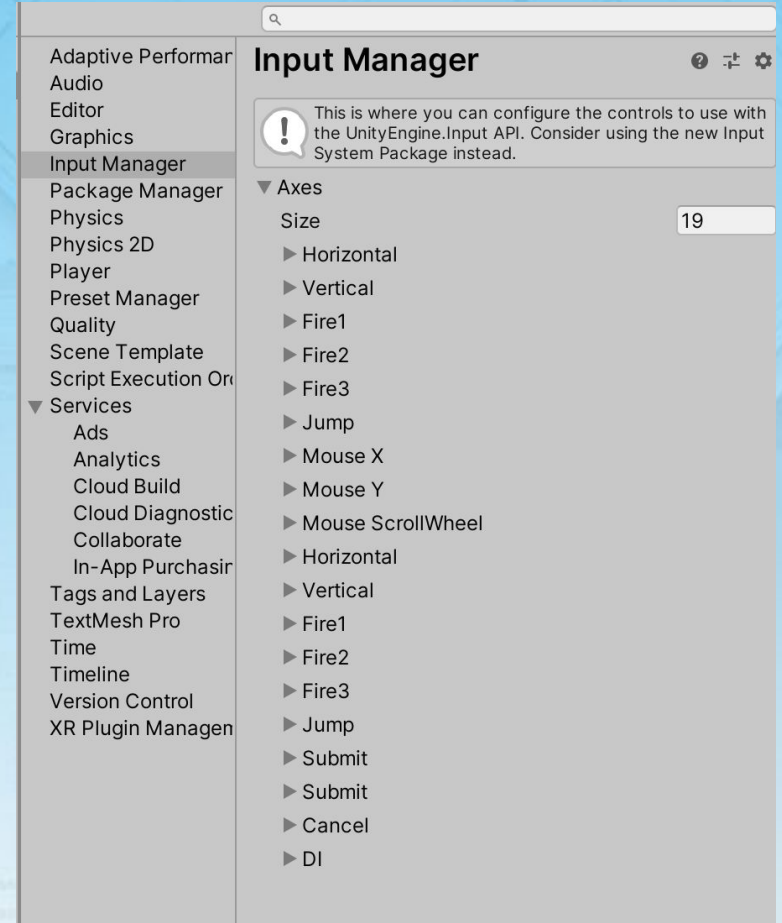
public class AtajosDeTeclado : MonoBehaviour
{
    private GameObject ptrPanel;

    // Start es llamada antes de que se actualice el primer frame.
    void Start()
    {
        ptrPanel = GameObject.Find("MiPanel");
    }

    // Update es llamada por frame.
    void Update()
    {
        if (Input.GetButtonDown("DI"))
        {
            if (ptrPanel.activeInHierarchy == true)
            {
                ptrPanel.SetActive(false);
            }
            else
            {
                ptrPanel.SetActive(true);
            }
        }
    }
}

```

Para ver y crear teclas virtuales ir al menú Edit->Project Settings->Input

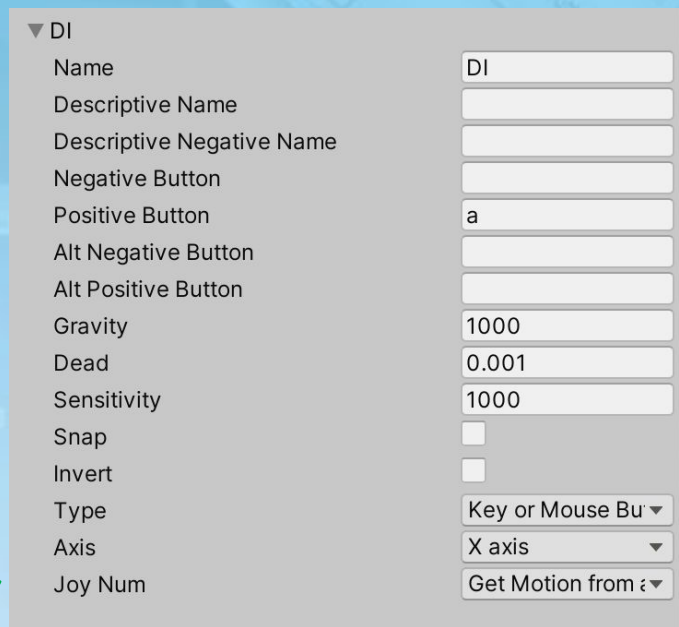
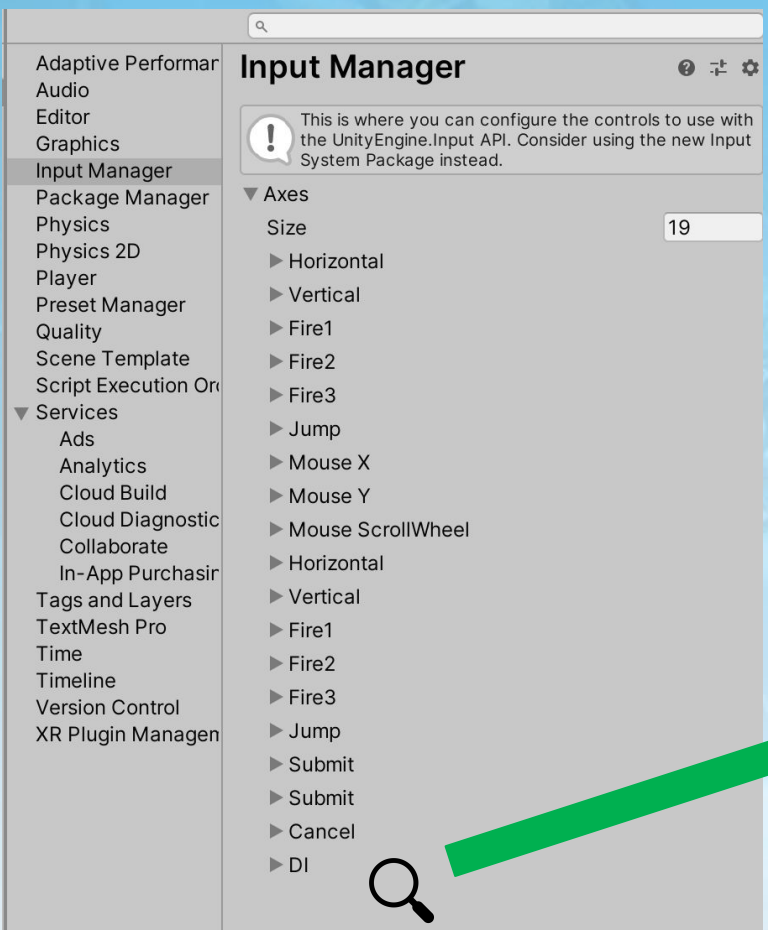


UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales:

Componentes contenedores básicos.

Trabajando con Teclado y creando Atajos de Teclado.

Si os fijáis en la última tecla virtual es “DI”, que es la que hemos creado y la que hemos utilizado en el código.



Más información:

<https://docs.unity3d.com/2020.2/Documentation/Manual/class-InputManager.html>



I.E.S. PALOMERAS
VALLECAS

UT2.- Introducción a la generación de interfaces gráficas de usuario con editores visuales: Componentes contenedores básicos.

Trabajando con Teclado y creando Atajos de Teclado.



I.E.S. VILLABLANCA



Y llegamos al final de la UT2. Una vez trabajado todas la documentación, trabajar sobre la práctica:
Práctica_UT2_IntroduccionInterface Grafica

Profesores: Raquel Rojo y Mario Santos.