UD X04

MANEJAR EL TIEMPO

Álvaro Holguera – IES Puerta Bonita (Madrid)

Este documento está bajo una licencia de Creative Commons

www.laultimapregunta.com

ÍNDICE

MÉTODO TIME

El control del tiempo es una de las herramientas básicas en cualquier programa. Desde saber cuánto tiempo llega el juego funcionando hasta para crear ciclos que nos permitan generar contadores.

TIME.DELTATIME

- El control del tiempo en Unity se realiza mediante el método `Time`, y todos sus submétodos
  - Un ejemplo, el submétodo `Time.time` nos devuelve el nº de segundos que han pasado desde que se inició el juego
- Una herramienta de gran utilidad es el método `Time.deltaTime`, que mide el tiempo pasado desde el anterior fotograma
  - Siempre hay que tener en cuenta que la función Update no funciona igual en todos los ordenadores, algunos lo ejecutarán más rápido y otros más lentos, dependiendo de su potencia, sin embargo, esta variable nos devuelve el tiempo pasado en segundos desde el anterior fotograma
  - Siempre que queramos mover/rotar un objeto, debemos multiplicarlo por este parámetro para asegurarnos que se moverá a la misma velocidad en todos los ordenadores
- Podemos usarlo para movimientos suaves:

```
transform.Translate(Vector3.forward * Time.deltaTime)
```

- También para medir distancias transcurridas:

```
distanceTraveled = distanceTraveled + Time.deltaTime
```

CORRUTINAS

- Uno de los problemas es que, al ser programación sincrónica, las funciones impiden que se siga ejecutando el código hasta que se han ejecutado por completo
  - También es un problema si queremos realizar una acción permanente, pero hacerlo en cada frame consumiría demasiados recursos
- Esto es un problema, especialmente en bucles en los que necesitamos que cada iteración del bucle se ejecute en un fotograma distinto (ejemplo, un fundido)
  - Si creamos una función con un bucle en el que en cada iteración se baja la luminosidad, descubriremos que pasará a negro de golpe, debemos usar una corrutina
- Para evitar esto, existen las corrutinas mediante el tipo de funciones IEnumerator, que permiten ejecutar código de forma asíncrona
  - Son llamadas mediante `StartCoroutine("nombreDeLaFuncion")` y detenidas mediante `StopCoroutine("nombreDeLaFuncion")`;
- Este método debe incluir un valor retornado mediante la sintaxis "yield return", que si devuelve un valor null se ejecutará cada fotograma

```
yield return null;
```

- También puede especificarse que espere un n° de segundos o décimas de segundo mediante la clase "WaitForSeconds()":

```
yield return new WaitForSeconds(1f);
```

Esta función permite que al pulsar la tecla f, se ejecute la corutina "Fade" que produce un desvanecimiento

```
void Update()
{
	if (Input.GetKeyDown("f"))
	{
		StartCoroutine("Fade");
	}
}

IEnumerator Fade()
{
	for (float f = 1f; f >= 0; f -= 0.1f)
	{
		Color c = renderer.material.color;
		c.a = f;
		renderer.material.color = c;
		yield return null;
		//yield return new WaitForSeconds(.1f);
	}
}
```

IMPORTANTE: las corrutinas no ejecutan un bucle por defecto, eso lo tenemos que crear mediante un for, o un while.

Este script sustituye la función de Update mediante una corrutina que se ejecuta cada segundo.

Incluso, cuando llega a 10 segundos se detiene

```
public class coroutine : MonoBehaviour
{
    private int cont = 0;

    void Start()
    {
        //Ejecutamos la corrutina
        StartCoroutine("Contador");
    }

    //Esta corrutina sustituye al update y se ejecuta cada segundo
    IEnumerator Contador()
    {
        for (; ; )
        {
            Debug.Log("Ha pasado un segundo");
            cont++;
            if(cont == 10)
            {
                StopCoroutine("Contador");
            }

            yield return new WaitForSeconds(1f);
        }
    }
}
```

Otra forma de repetir el proceso sin necesidad de bucle for, es iniciar la coroutina justo después de esperar el tiempo indicado

CONTADORES

EJEMPLO DE USO DE CORRUTINAS PARA CREAR CONTADORES Y RELOJES

Álvaro Holguera (CIFP José Luis Garci)

CONTADORES

- Se puede crear un contador de intervalos sin necesidad de usar corrutinas mediante Time.deltaTime y la función Update
- Crearemos una variable privada de tipo float que determinará el intervalo (1 segundo por ejemplo)

```
private float Intervalo = 1.0f;
```

- Ahora en la función de Update restamos a ese intervalo una unidad de deltaTime, que es el intervalo entre el frame actual y el anterior:

```
Intervalo -= Time.deltaTime;
```

- Creamos una comprobación, que cuando llega a cero lo ponemos de nuevo en el tiempo estipulado y hacemos lo que corresponda

```
if(Intervalo < 0) { Intervalo = 1; }
```

```
private float Intervalo = 1.0f;

// Start is called before the first frame update
void Start()
{

}

// Update is called once per frame
void Update()
{
    Intervalo -= Time.deltaTime;
    if(Intervalo < 0)
    {
        Debug.Log(”Se ha cumplido el intervalo");
        Intervalo = 1;
    }
}
```

CUENTA ATRÁS Y RELOJES

Cuenta hacia atrás

- Crearemos un texto que lo colocaremos en el lugar de la pantalla que deseemos
- Hay que añadir la librería de UI a nuestro scrip

```
using UnityEngine.UI;
```

- Creamos las variables: tiempo desde el que se inicia, variable pública a la que asignaremos el texto en Unity, y una variable que convertirá valores hexadecimales en unidades de tiempo:

```
private float timer = 100f;
public Text counter;
private int TimerInt;
```

- En la función de Update

```
timer -= Time.deltaTime;
TimerInt = (int)timer;
counter.text = TimerInt.ToString();
```

Reloj

- Con estos parámetros, podemos crear un reloj que va aumentando
- Hay que depurar el código para que los números se muestren con un 0 delante si son menos de 10
- Otra posibilidad para reducir el consumo de memoria es ejecutar el script cada segundo, en lugar de cada frame utilizando una corutina. Lo veremos en las siguientes dos diapositivas.

CREAR UN RELOJ

Añadir un reloj digital optimizado

Ejemplo de código para un reloj con ceros delantes:

```
// Update is called once per frame
    void Update()
    {
        timer += Time.deltaTime;
        seconds = (int)timer;
        if(seconds < 10)
        {
            secondsSt = "0" + seconds.ToString();
        }
        else
        {

            //Si superamos el minuto
            if (seconds > 59)
            {
                minutes++;
                timer = 0f;
            }
            secondsSt = seconds.ToString();
        }
        if(minutes < 10)
        {
            minutesSt = "0" + minutes.ToString();
        }
        else
        {

            //Si superamos la hora
            if (minutes > 59)
            {
                hours++;
                minutes = 0;
            }
            minutesSt = minutes.ToString();
        }
        if (hours < 10)
        {
            hoursSt = "0" + hours.ToString();
        }
        else
        {
            hoursSt = hours.ToString();
        }

        //Añadimos el texto
        clockCounter.text = hoursSt + ":" + minutesSt + ":" + secondsSt;
    }

}
```

CREAR RELOJ MEDIANTE CORRUTINAS

Ejemplo de código que hace el mismo reloj, pero con un bucle que se repite cada segundo:

```
// Start is called before the first frame update
    void Start()
    {
        //Llamamos a la corrutina que se ejecutará cada segundo
        StartCoroutine("clockStart");
    }

    //No tenemos función update
    IEnumerator clockStart()
    {
        for (; ; )
        {

            seconds += 1;
            if (seconds < 10)
            {
                secondsSt = "0" + seconds.ToString();
            }
            else
            {

                //Si superamos el minuto
                if (seconds > 59)
                {
                    minutes++;
                    seconds = 0;
                }
                secondsSt = seconds.ToString();
            }
            if (minutes < 10)
            {
                minutesSt = "0" + minutes.ToString();
            }
            else
            {

                //Si superamos la hora
                if (minutes > 59)
                {
                    hours++;
                    minutes = 0;
                }
                minutesSt = minutes.ToString();
            }
            if (hours < 10)
            {
                hoursSt = "0" + hours.ToString();
            }
            else
            {
                hoursSt = hours.ToString();
            }

            //Añadimos el texto
            clockCounter.text = hoursSt + ":" + minutesSt + ":" + secondsSt;

            yield return new WaitForSeconds(1f);
        }

    }
```