**Coroutines & IEnumerator**

Una corrutina es realmente solo un método con un tipo de devolución de “IEnumerator”. Una diferencia clave es que no simplemente “devuelve” el tipo de datos como lo haría en un método normal. En cambio, “cede” un valor que hace que la ejecución de la lógica se detenga en su lugar; puede reanudarse más tarde. Echemos un vistazo a una muestra rápida:

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;

}

}

Es esencialmente una función declarada con un tipo de retorno IEnumerator y con una instrucción de retorno yield incluida en algún lugar de su cuerpo. La linea con la instrucción de retorno yield es el punto en el cual la ejecución se pausará y reanudará en el siguiente frame. Para establecer una corrutina en ejecución, necesitas usar la función StartCoroutine

void Update()

{

if (Input.GetKeyDown("f"))

{

StartCoroutine("Fade");

}

}

Notará que el contador en el bucle de la función Fade mantiene su valor correcto durante el tiempo de vida de la corrutina. De hecho, cualquier variable o parametro será preservado correctamente entre los yields.

Por defecto, una corrutina es reanudada en el siguiente frame después de haberse interrumpido (yield), pero también es posible introducir un tiempo de retardo usando WaitForSeconds:-

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 new WaitForSeconds(.1f);

}

}