Ejercicios Scripts: Movimiento

1. Crea un script asociado a un objeto en la escena que inicialice un vector de 3 posiciones con valores entre 0.0 y 1.0, para tomarlo como un vector de color (Color). Cada 120 frames se debe cambiar el valor de una posición aleatoria y asignar el nuevo color al objeto. Parametrizar la cantidad de frames de espera para poderlo cambiar desde el inspector.

Ayuda: Usa la clase Random de Unity. Busca información en la referencia de la API. Recuerda, para cambiar valores desde el inspector deben estar declarados públicos en la clase del script.

Ayuda: Crea el script y asócialo a uno de los objetos agregándolo como Componet o símplemente arrastrando el script sobre el objeto.

2. Crea un script asociado a la esfera con dos variables Vector3 públicas. Dale valor a cada componente de los vectores desde el inspector. Muestra en la consola:
    a. La magnitud de cada uno de ellos.
    b. El ángulo que forman
    c. La distancia entre ambos.
    d. Un mensaje indicando qué vector está a una altura mayor.

    Muestra en el inspector cada uno de esos valores.

Ayuda: para crear los vectores tienes que usar el constructor, para verlos en el Inspector deben ser públicos:

```
Vector3 myVector;
------------------------
myVector = new Vector3(0.0f, 1.0f, 0.0f);
```

Ayuda: consulta en la referencia de la API de Unity las utilidades para el tratamiento de vectores con la clase Vector3.

3. Muestra en pantalla el vector con la posición de la esfera.

Ayuda: La posición es una propiedad del component Transform del GameObject, en este caso la esfera. Será necesario recuperar una referencia al objeto Transform (Component).

    a. Las referencias a los Components se recuperan con `GetComponent<TipodeComponent>()` → `GetComponent<Transform>()`
    b. En el caso del Transform la propiedad transform del GameObject nos da su referencia. `transform.position`

Por tanto, puedes resolver el problema con alguna de las alternativas:

```
transform.position
GetComponent<Transform>()
```

4. Crea un script para la esfera que muestre en consola la distancia a la que están el cubo y el cilindro.

Ayuda: Los objetos cubo y cilindro pueden buscar el objeto esfera por su etiqueta. De esta forma se tendría una referencia al GameObject, a partir de él necesitamos la referencia al Component Transform y calcular la distancia entre los vectores position.

```
laesfera = GameObject.FindWithTag(“blue_sphere”);
```

5. Selecciona tres posiciones en tu escena a través de un objeto invisible (marcador) que incluya 3 vectores numéricos para configurar posiciones en las que quieres ubicar los objetos en respuesta a pulsar la barra espaciadora. Estos vectores representan un desplazamiento respecto a la posición original de cada objeto. Crea un script que ubique en las posiciones configuradas cuando el usuario pulse la barra espaciadora.

Ayuda:

- Configura la escena con 3 objetos, cada uno de ellos debe tener una variable pública `desplazamiento` para que puedas configurarla en el inspector. Debe ser de tipo `Vector3`, cada componente `(deltax, deltay, deltaz)` representa la variación que sufrirá la `(x, y, z)` de la posición respectivamente.
- Utiliza `Input.GetAxis()` para determinar si se ha pulsado la barra espaciadora.

6. Agrega un campo velocidad a un cubo y asígnale un valor que se pueda cambiar en el inspector de objetos. Muestra la consola el resultado de multiplicar la velocidad por el valor del eje vertical y por el valor del eje horizontal cada vez que se pulsan las teclas flecha arriba-abajo ó flecha izquierda-derecha. El mensaje debe comenzar por el nombre de la flecha pulsada.

Ayuda: Usa las clase Input y KeyCode de Unity:

```
Input.GetAxis("Horizontal"), Input.GetAxis("Vertical")

Input.GetKey(), KeyCode.
```

Ayuda: Recuerda configurar el proyecto para usar el Input System (Old), en lugar de Input System Manager (New). Para ello: Edit → Project Settings → Other → Player

7. Mapea la tecla H a la función disparo.

Ayuda: Utiliza el Input Manager para redefinir el mapeo por defecto en los controladores.

8. Crea un script asociado al cubo que en cada iteración traslade al cubo una cantidad proporcional un vector que indica la dirección del movimiento: **`moveDirection`** que debe poder modificarse en el inspector. La velocidad a la que se produce el movimiento también se especifica en el inspector, con la propiedad `speed`. Inicialmente la velocidad debe ser mayor que 1 y el cubo estar en una posición **`y=0`**. En el informe de la práctica comenta los resultados que obtienes en cada una de las siguientes situaciones:
    a. duplicas las coordenadas de la dirección del movimiento.
    b. duplicas la velocidad manteniendo la dirección del movimiento.
    c. la velocidad que usas es menor que 1

d. la posición del cubo tiene **`y>0`**
e. intercambiar movimiento relativo al sistema de referencia local y el mundial.

Ayuda: Debes crear un vector que recoja la dirección del movimiento de clase **`Vector3`**.

Ayuda: Debes utilizar **`Translate(x, y, z)`**, siendo (x, y, z) las coordenadas de **moveDirection**.

9. Mueve el cubo con las teclas de flecha arriba-abajo, izquierda-derecha a la velocidad **`speed`**. Cada uno de estos ejes implican desplazamientos en el eje vertical y horizontal respectivamente. Mueve la esfera con las teclas **`w-s`** (movimiento vertical) **`a-d`** (movimiento horizontal).

Ayuda: Utiliza lo aprendido en los ejercicios anteriores sobre la clase **`Input`** y la clase **`Translate`**.

10. Adapta el movimiento en el ejercicio 9 para que sea proporcional al tiempo transcurrido durante la generación del frame.

Ayuda:Utiliza el valor **`Time.DeltaTime`** para escalar la cantidad de espacio que recorre el objeto entre un frame y otro.

11. Adapta el movimiento en el ejercicio 10 para que el cubo se mueva hacia la posición de la esfera. Debes considerar que el avance no debe estar influenciado por cuánto de lejos o cerca estén los dos objetos.

Ayuda: La dirección del movimiento debe ser el vector que une al cubo con la esfera. Se debe tener en cuenta que el cubo no debe modificar su altura.

Ayuda: El vector que marca el movimiento aporta la información de la dirección. Es recomendable utilizar vectores de magnitud 1, para que el avance no dependa de dicha magnitud. La clase Vector3 tiene la utilidad de normalizar vectores.

```
miVector.normalized
```

12. Adapta el movimiento en el ejercicio 11 de forma que el cubo avance mirando siempre hacia la esfera, independientemente de la orientación de su sistema de referencia. Para ello, el cubo debe girar de forma que el eje Z positivo apunte hacia la esfera . Realiza pruebas cambiando la posición de la esfera mediante las teclas **`awsd`**

Ayuda: La clase **`Transform`** tiene el método **`LookAt(Transform target)`**, que rota al objetivo para hacer que el vector hacia delante apunte al objetivo (target). Se debe tener en cuenta que puede ser necesario especificar que el movimiento sea relativo al espacio del mundo.

13. Utilizar el eje “Horizontal” para girar el objetivo y que avance siempre en la dirección hacia adelante.

Ayuda: La dirección hacia adelante (eje Z positivo) se puede obtener con la propiedad **`fordward`** del **`Transform`**. No confundir con **`Vector3.fordward`**.

Ayuda: Puedes usar la función **`Debug.DrawRay`** para depuración