DESARROLLO DE VIDEOJUEGOS Y REALIDAD VIRTUAL CON UNITY 3D

Tema 7: Scripting

unity

TEMA 7. SCRIPTING

- Conceptos esenciales de la programación en C#
- Scripts y la clase MonoBehaviour
- Variables de un script
- Eventos del bucle de juego
- Input mediante teclado, ratón y botones
- Transformaciones, coordenadas y vectores
- Creación y destrucción de objetos
- Comunicación entre objetos

TEMA 7. SCRIPTING

- Conceptos esenciales de la programación
  - Sentencia
  - Instrucciones y palabra reservada
  - Variable
  - Arrays y Listas
  - Constante
  - Estructuras de control
    - De selección:
      - If / else if / else
      - Switch
    - Iterativas
      - For
      - Foreach
      - While
      - Dowhile

TEMA 7. SCRIPTING

- Conceptos esenciales de la programación orientada a objetos
  - Clases
  - Objetos
  - Atributos
  - Métodos
  - Constructores
  - Herencia
  - Clases abstractas
  - Modificadores de acceso
    - Public
    - Private
    - Protected
  - El modificador static

TEMA 7. SCRIPTING

- Conceptos esenciales de la programación en C#
  - Comentarios
  - Definición de clase
    - Public class MiClase : MonoBehaviour
  - Importación de librerías
    - using UnityEngine;
  - Variables, atributos y métodos.
  - Visibilidad: public y private.
    - [SerializeField]
  - Métodos virtuales / override
  - El tipo "var"
  - Arrays y Listas
  - Constantes
  - Estructuras (struct)
  - Enumeraciones (enum)
  - Herencia
  - Sobrecarga
  - Clases abstractas
  - Interfaces
  - Corrutinas

TEMA 7. SCRIPTING

- Conceptos esenciales de la programación en C#
  - Tipos
    - Enteros: int, long
    - Punto flotante: float, double
    - Decimal de alta precisión: decimal
    - Booleano: bool
    - Enumeraciones: enum
    - Estructuras: struct
    - Carácter: char
    - Cadenas: string

TEMA 7. SCRIPTING

- Conceptos esenciales de la programación en C#

  - Operadores

    https://docs.microsoft.com/es-es/dotnet/csharp/language-reference/operators/

TEMA 7. SCRIPTING

- Scripts y la clase MonoBehaviour
  - Crear un script
  - Nombrar un script
  - Métodos y propiedades:
    - enabled
    - name
    - tag
    - transform
    - gameObject
    - gameObject.SetActive
    - GetComponent

TEMA 7. SCRIPTING

- Variables de un script
  - int
  - bool
  - float
  - double
  - string
  - char
  - Vector3. Almacena tres flotantes.
  - GameObject
  - Transform
  - Texture, Material, AudioClip, etc.
  - Array
  - List

TEMA 7. SCRIPTING

- Variables de un script

  - Variables editables desde el inspector
    - public
    - [SerializeField]
    - [RequireComponent(typeof(Animator)]

TEMA 7. SCRIPTING

- Eventos del bucle de juego
  - Awake.
    - Primer método en ejecutarse (sólo una vez)
    - Es seguro para obtener referencias a otros objetos (métodos Find).
  - Start.
    - Posteriormente a la ejecución del Awake de TODOS los demás scripts (sólo una vez).
    - Segunda fase de inicialización.
  - Update.
    - Time.deltaTime
  - FixedUpdate.
  - OnCollision…
  - OnTrigger…

https://docs.unity3d.com/Manual/ExecutionOrder.html

TEMA 7. SCRIPTING

- Eventos del bucle de juego
  - OnApplicationQuit: se ejecuta sobre cada script cuando se cierra el juego.
  - OnDestroy: se ejecuta antes de la destrucción de un objeto.

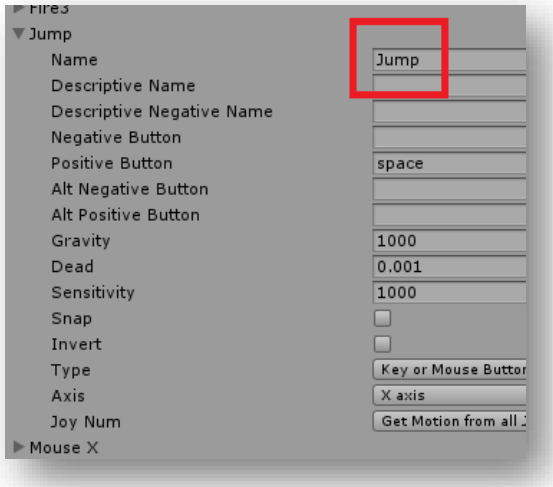
https://docs.unity3d.com/Manual/ExecutionOrder.html

TEMA 7. SCRIPTING

- Input mediante teclado, ratón y botones
  - La clase Input.
    - Se deben programar en el método Update.
    - Métodos:
      - GetKey (si la tecla está pulsada)
      - GetKeyDown (si la tecla se ha pulsado)
      - GetKeyUp (si la tecla se ha soltado)
      - GetKeyCode
      - GetButton
      - GetButtonDown
      - GetButtonUp
      - GetMouseButton
      - GetMouseButtonDown
      - GetMouseButtonUp

TEMA 7. SCRIPTING

- Input mediante teclado, ratón y botones
  - La clase Input.
    - Comprobación de tecla pulsada
      - Mediante GetKey… y el enum KeyCode:
        - if (Input.GetKeyDown(KeyCode.Alpha1)) //Tecla 1
      - Mediante GetButton… y el nombre del botón (Edit/Project Settings/Input)
        - Jump
        - Fire1, Fire2, Fire3
          - if (Input.GetButtonDown("Jump"))//



TEMA 7. SCRIPTING

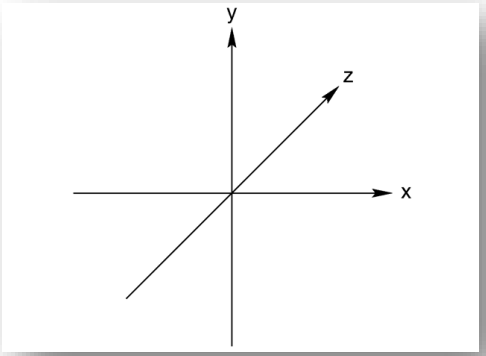
- Input mediante teclado, ratón y botones
  - La clase Input.
    - Input mediante ejes virtuales: El método GetAxis.
      - Válido para controles analógicos y digitales (teclado, ratón, gamepad).
      - Devuelve un float con rangos dependientes del control.
      - Ejes virtuales incluidos por defecto (ver Settings de InputManager):
        - "Horizontal", "Vertical", "Mouse X", "Mouse Y"

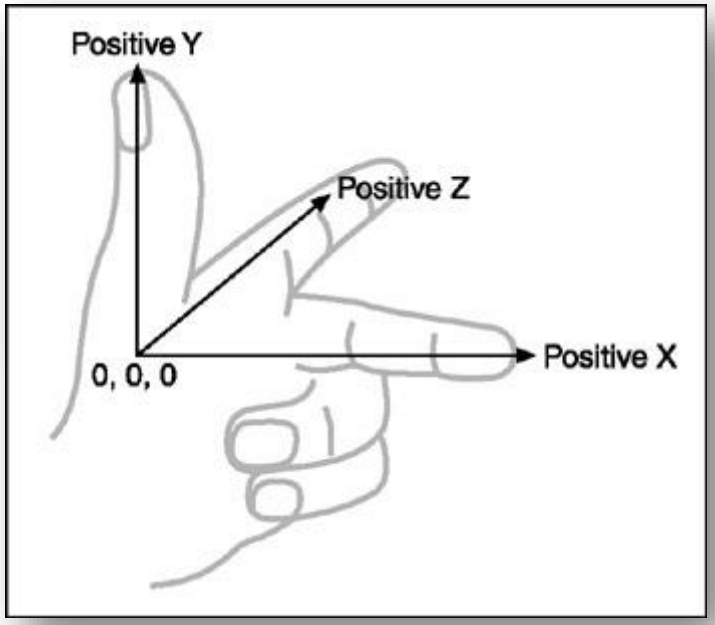
TEMA 7. SCRIPTING

- Input mediante teclado, ratón y botones
  - Ratón (métodos de MonoBehaviour)
    - OnMouseDown
    - OnMouseDrag
    - OnMouseEnter
    - OnMouseExit
    - OnMouseOver
    - OnMouseUp
    - OnMouseUpAsButton

TEMA 7. SCRIPTING

- Transformaciones, coordenadas y vectores.
  - Sistema de coordenadas: *left-handed coordinate system*





TEMA 7. SCRIPTING

- Transformaciones, coordenadas y vectores.
  - Sistema de coordenadas:
    - Transformaciones geométricas:
      - Traslación
      - Rotación
      - Escalado
    - Transformaciones locales vs globales
      - Space.Self
      - Space.World

TEMA 7. SCRIPTING

- Transformaciones, coordenadas y vectores.
  - Puntos
  - Vectores
  - Direcciones (vectores normalizados)
  - La clase Vector3
    - Compuesto por tres coordenadas: x, y, z.
    - Constructor
    - Propiedades
      - magnitude
      - sqrMagnitude
      - normalized
      - x, y, z

TEMA 7. SCRIPTING

- Transformaciones, coordenadas y vectores.
  - La clase Vector3: el método Distance
  - La clase Vector3: sqrMagnitude vs magnitude
    - Para comparaciones de distancia, es más rápido usar sqrMagnitude y comparar con el cuadrado de la distancia de referencia.

```
using UnityEngine;
using System.Collections;

public class ExampleClass : MonoBehaviour
{
    // detects when the other transform is closer than closeDistance
    // this is faster than using Vector3.magnitude
    public Transform other;
    public float closeDistance = 5.0f;

    void Update()
    {
        if (other)
        {
            Vector3 offset = other.position - transform.position;
            float sqrLen = offset.sqrMagnitude;

            // square the distance we compare with
            if (sqrLen < closeDistance * closeDistance)
            {
                print("The other transform is close to me!");
            }
        }
    }
}
```

https://docs.unity3d.com/ScriptReference/Vector3-sqrMagnitude.html

TEMA 7. SCRIPTING

- Transformaciones, coordenadas y vectores.
  - La clase Vector3: vectores predefinidos. Son vectores normalizados que representan direcciones de los ejes.

Alias	Vector
Vector3.forward	Vector3(0,0,1)
Vector3.back	Vector3(0,0,-1)
Vector3.up	Vector3(0,1,0)
Vector3.down	Vector3(0,-1,0);
Vector3.right	Vector3(1,0,0)
Vector3.left	Vector3(-1,0,0)

TEMA 7. SCRIPTING

- Transformaciones, coordenadas y vectores.
  - La clase Vector3. Operaciones:
    - Suma de vectores.
      - Devuelve un vector.
      - Permite desplazar un objeto desde el primer vector a la posición resultante de la suma de los vectores.
    - Resta de vectores.
      - Devuelve un vector.
      - Permite obtener el vector a aplicar para ir desde un GameObject a otro.
    - Multiplicación del vector por una magnitud.
      - Devuelve un vector.
      - Permite mover en un sentido un GameObject a una velocidad dada utilizando Translate.
    - Normalización y módulo:
      - Permite obtener la distancia entre dos puntos y la dirección (vector normalizado).
      - Atributo magnitude: devuelve un float.
      - Atributo normalized: devuelve la dirección.

TEMA 7. SCRIPTING

- Transformaciones, coordenadas y vectores
  - El componente Transform
    - Método Translate
    - Método Rotate
    - Método LookAt
  - Accederemos al componente Transform a través de la propiedad transform.
  - Propiedad transform:
    - Accesible desde TODOS los componentes de un GameObject.
    - Propiedades:
      - position
      - localPosition
      - rotation
      - localRotation
      - localScale
      - eulerAngles
      - localEulerAngles

TEMA 7. SCRIPTING

- Transformaciones, coordenadas y vectores.
  - El componente transform: vectores predefinidos. Son vectores que representan las direcciones de los ejes locales del transform. Útiles para obtener vectores de dirección.

Alias
transform.forward
-transform.forward
transform.up
-transform.up
transform.right
-transform.right

TEMA 7. SCRIPTING

- Transformaciones, coordenadas y vectores. Movimiento de objetos.
  - Asignando una posición (cambio instantáneo):
    - go.transform.position = new Vector3(10,0,15);
  - Mediante una traslación, utilizando el método Translate (en función del Delta Time para conseguir cambios suaves):
    - Ejemplo: avanza hacia el forward del objeto (Space.Self vs Space.World).
    - go.transform.Translate(dirección * incremento * Time.deltaTime * Space.Self)
  - Mediante el método MoveTowards de la clase Vector3. Genera un avance en dirección del target:
    - transform.position = Vector3.MoveTowards(transform.position, target.position, speed * time.deltaTime);

TEMA 7. SCRIPTING

- Transformaciones, coordenadas y vectores. Rotación de objetos.
  - Mediante ángulos de Euler (*eulerAngles)*:
    - Ventaja: más comprensible.
    - Ventaja: Permiten rotaciones de más de 180 grados.
    - Desventajea: bloqueo de cardán.
  - Mediante cuaterniones (*Quaternions)*.
    - Ventaja: no provoca el bloqueo de cardán.
    - Desventaja: no permite rotaciones de más de 180 grados.
    - Desventaja: no es fácilmente comprensible.
  - Unity almacena las rotaciones como Quaternions.
  - Unity muestra las rotaciones en el inspector en ángulos de Euler (Unity los convierte).

TEMA 7. SCRIPTING

- Transformaciones, coordenadas y vectores. Rotación de objetos con ángulos de Euler.
  - Modificando la propiedad *eulerAngles* de la clase Transform:
    - Ejemplo: asigna 10 al eje de rotación y
    - transform.eulerAngles = new Vector3(0, 10, 0);
  - Utilizando el método Rotate de la clase Transform.
    - Ejemplo: rota 1 grado sobre el eje y.
    - transform.Rotate(new Vector3(0, 1, 0));
  - Utilizando el método LookAt de la clase Transform.
    - El objeto se orienta inmediatamente hacia otro objeto.
    - transform.LookAt(GameObject.Find("Torreta").transform);

TEMA 7. SCRIPTING

- Transformaciones, coordenadas y vectores. Rotación de objetos con Quaternions.
  - Modificando la propiedad *rotation*:
    - La propiedad se expresa en *Quaternions*, pero estos se pueden crear a partir de ángulos de Euler.
    - Ejemplo:
      - go.transform.rotation = Quaternion.Euler(10,0,15);
    - Quaternion.identity indica una "no rotación" sobre los ejes del mundo.
  - Quaternion.LookRotation. Crea una rotación a partir de un Vector3.
  - Quaternion.RotateTowards. Rota hacia una rotación.

TEMA 7. SCRIPTING

- Transformaciones, coordenadas y vectores. Rotación de objetos con Quaternions.
  - Rotando un transform hacia la dirección de otro (a una velocidad determinada):
    - Ejemplo:

```
Quaternion q = Quaternion.LookRotation(enemigo.position - eje.position);

eje.rotation = Quaternion.RotateTowards(eje.rotation, q, speed * Time.deltaTime);
```

TEMA 7. SCRIPTING

- Transformaciones, coordenadas y vectores. Escalado de objetos.
  - La escala NO es lo mismo que el tamaño.
  - Siempre se realiza en coordenadas locales (con respecto al padre en jararquía).
  - Modificando la propiedad localScale
    - go.transform.localeScale = new Vector3(2f, 2f, 1f);

TEMA 7. SCRIPTING

- **Creación** y destrucción de objetos.
  - La creación de los objetos se realiza a través del método Instantiate.
  - Se puede indicar de quién es hijo.
  - Se puede indicar su posición y rotación.
  - Algunas sobrecargas:
    - public static Object Instantiate(Object original);
      - Crea el objeto sin vincular a ningún otro objeto.
    - public static Object Instantiate(Object original, Transform parent);
      - Crea el objeto como hijo del que lo crea.
    - public static Object Instantiate(Object original, Vector3 position, Quaternion rotation);
      - Indica posición y rotación.
      - Quaternion.identity = Alineado con los ejes del mundo.
    - public static Object Instantiate(Object original, Vector3 position, Quaternion rotation, Transform parent);
      - Crea el objeto como hijo de un objeto determinado.

TEMA 7. SCRIPTING

- Creación y **destrucción** de objetos.
  - La destrucción de los objetos se realiza a través del método Destroy.
  - No se culmina la destrucción hasta que no termina el frame en el que se realiza (durante la invocación al método OnDestroy).
  - Si la referencia utilizada es de un GameObject, éste se elimina junto con todos sus descendientes.
    - Destroy(this.gameObject);
  - Si la referencia utilizada es de un componente, éste será eliminado pero no el objeto como tal.
  - Si se desea destruir el objeto de manera inmediata se debe utilizar el método DestroyInmediate.
  - El método Destroy puede ir acompañado de un *delay* expresado en segundos:
    - Destroy(this.gameObject, 5)

TEMA 7. SCRIPTING

- Comunicación entre objetos
  - Obtención de referencias a otros objetos (sin utilizar físicas):
    - Arrastrando los GameObjects a atributos públicos (o SerializeField) en el Inspector.
      - Podemos generalizar indicando que el atributo es GameObject.
      - Podemos particularizar indicando que el atributo es de un tipo concreto de componente.
        - Ahorramos la invocación a GetComponent.
        - "Protegemos" el código.
      - **LIMITACIÓN**: un prefab no puede hacer referencias a objetos en la escena (sí a otros prefab).

TEMA 7. SCRIPTING

- Comunicación entre objetos
  - Obtención de referencias a otros objetos (sin utilizar físicas):
    - Buscando por nombre:
      - GameObject.Find("NombreDelObjeto").
      - Útil cuando el nombre del objeto es único.
      - Dificultad cuando el nombre del objeto no es único por existir "clones".
    - Buscando por etiqueta:
      - Devuelve un array de objetos.
      - GameObject.FindGameObjectsWithTag("NombreEtiqueta");

TEMA 7. SCRIPTING

- Comunicación entre objetos
  - Obtención de componentes a partir de un GameObject:
    - Método GetComponent:
      - gameobject.GetComponent<RigidBody>();
    - Método GetComponentInParent
    - Método GetComponentInChildren
    - Método GetComponentsInParent
    - Método GetComponentsInChildren
  - NOTA: Los scripts son componentes. Esto implica que se podrán obtener referencias a los scripts de otros GameObject e invocar sus métodos públicos.

TEMA 7. SCRIPTING

- Depuración
  - Debug.logger.logEnabled = true/false
  - Debug.Log / print
  - Debug.LogWarning
  - Debug.LogError
  - Debug.DrawLine
  - Debug.DrawRay