UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS D'ALCOI

# Scripts y Métodos básicos

Jordi Linares Pellicer

unity

# Plantilla de un script

```
using UnityEngine;
using System.Collections;

public class Ejemplo : MonoBehaviour {
   // Use this for initialization
   void Start () {
   }

   // Update is called once per frame
   void Update ()  {
   }
}
```

- En **UnityEngine** disponemos de la clase **MonoBehaviour**, básica para definir scripts que serán asociados a **GameObjects** o **Prefabs**
- **System.Collections** pertenece a **.NET** y nos permite definir listas, arrays dinámicos, tablas hash etc.

# Desarrollando scripts en C#

- En la programación de juegos en **Unity3D**, y en general, no todas las características de **C#** son adecuadas
  - Problemas de rendimiento
  - Problemas con la integración del editor
  - Particularidades al derivar de **MonoBehaviour**
  - Características no implementadas completamente en **Unity3D**
  - etc.
- No obstante, la mayoría están disponibles e incluso las más avanzadas pueden resultar muy útiles en muchos contextos (delegados y eventos, expresiones lambda, linq, métodos de extensión etc.)
- En cualquier caso, hay que tener mucho cuidado con aquello que se ejecuta en métodos como **Update(), FixedUpdate(), LateUpdate()** etc. debido a que son ejecutados de forma repetida y con mucha frecuencia => problemas de eficiencia y pocos frames por segundo

# Desarrollando scripts en C#

- Los scripts en **Unity3D** derivan de la clase **MonoBehaviour**
- Es importante tener en cuenta que un script es un componente de un GameObject, y por tanto, **this** en este contexto hace referencia al **script** y no al **GameObject**
- Podemos asignar cuantos queramos a un GameObject
- Métodos más destacados:
  - **Awake(), Start()**
  - **Update(), FixedUpdate(), LateUpdate()**
  - **OnCollisionEnter(), OnCollisionStay(), OnCollisionExit() y sus equivalentes OnTrigger()**
  - **OnBecameVisible(), OnBecameInvisible()**
  - **OnMouseDown(), OnMouseOver()**
  - **OnEnable(), OnDisable()** ...

# Awake() y Start()

**void Awake() {**
**...**
**}**

- El Awake() es el primer método que se ejecuta cuando el objeto es inicializado
- DEBE ser utilizado como substituto del constructor
- Siempre se ejecuta antes que cualquier Start() de todos los objetos
- Resulta perfecto para inicializar enlaces cruzados entre objetos

**void Start() {**
**...**
**}**

- El método Start() supone un segundo paso de inicialización (tras todos los Awake() de los objetos) y todos los Start() se ejecutarán antes que cualquier Update()
- Mediante Awake() y Start() podemos inicializar todos nuestros objetos en 2 pasos
- Start() sólo se llama si el GameObject está activado (aparece 'clickado' en el Inspector)

# Update(), FixedUpdate(), LateUpdate()

- Uno de los principales objetivos de un script consiste en actualizar de alguna forma el GameObject al que está asignado
- Esta actualización consiste generalmente en cambiar su posición, escala, rotación, la aplicación de fuerzas, cambios de colores, de estado etc.
- Para lleva a cabo estas tareas, **Unity3D** nos proporciona varias alternativas: **Update(), FixedUpdate(), LateUpdate()**

```
void Update() {
...
}
```

- **Update** es la más común y se caracteriza porque será llamada en cada frame
- El buen rendimiento del código que pongamos en este método (en todos los objetos de la escena) será uno de los aspectos que más influirán en los FPS finales
- Perfecto para actualizar objetos no físicos y recibir interacciones de los usuarios
- Sólo se ejecutará si el objeto está activo (enabled)

# Update(), FixedUpdate(), LateUpdate()

```
void FixedUpdate() {
...
}
```

- **FixedUpdate** se llamará en intervalos regulares y todas las actualizaciones físicas (paso del motor físico) se llevarán a cabo tras el FixedUpdate
- Por tanto, es el sitio perfecto para todo lo relacionado con cambios en el Rigidbody del GameObject, es decir, aplicar fuerzas por ejemplo
- FixedUpdate puede ser invocado con más o menos frecuencia que Update, dado que éste último lo lleva a cabo en un ratio variable, mientras que FixedUpdate lo hace siempre de forma constante (este ratio puede cambiarse)

# Update(), FixedUpdate(), LateUpdate()

```
void LateUpdate() {
...
}
```

- El **LateUpdate** de un Gameobject se invocará cuando todos los Update de todos los objetos hayan sido ejecutados
- Por tanto, es como una nueva oportunidad de actualizar posiciones de objetos cuando todos los objetos de la escena ya han llevado a cabo sus actualizaciones mediante su Update
- Al igual que Update, se ejecuta en cada frame y suele utilizarse cuando un objeto sigue, por ejemplo a otro en la escena, de forma que tenemos la seguridad de que el primero se ha movido y el que le sigue se actualiza tras el movimiento del primero (así no tenemos que preocuparnos respecto al orden en el que los Update se ejecutan)

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS D'ALCOI

# Scripts y Métodos básicos

Jordi Linares Pellicer

unity