



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA

# Gestión del tiempo en Unity3D

Ramón Mollá

Dpto. Sistemas Informáticos y Computación

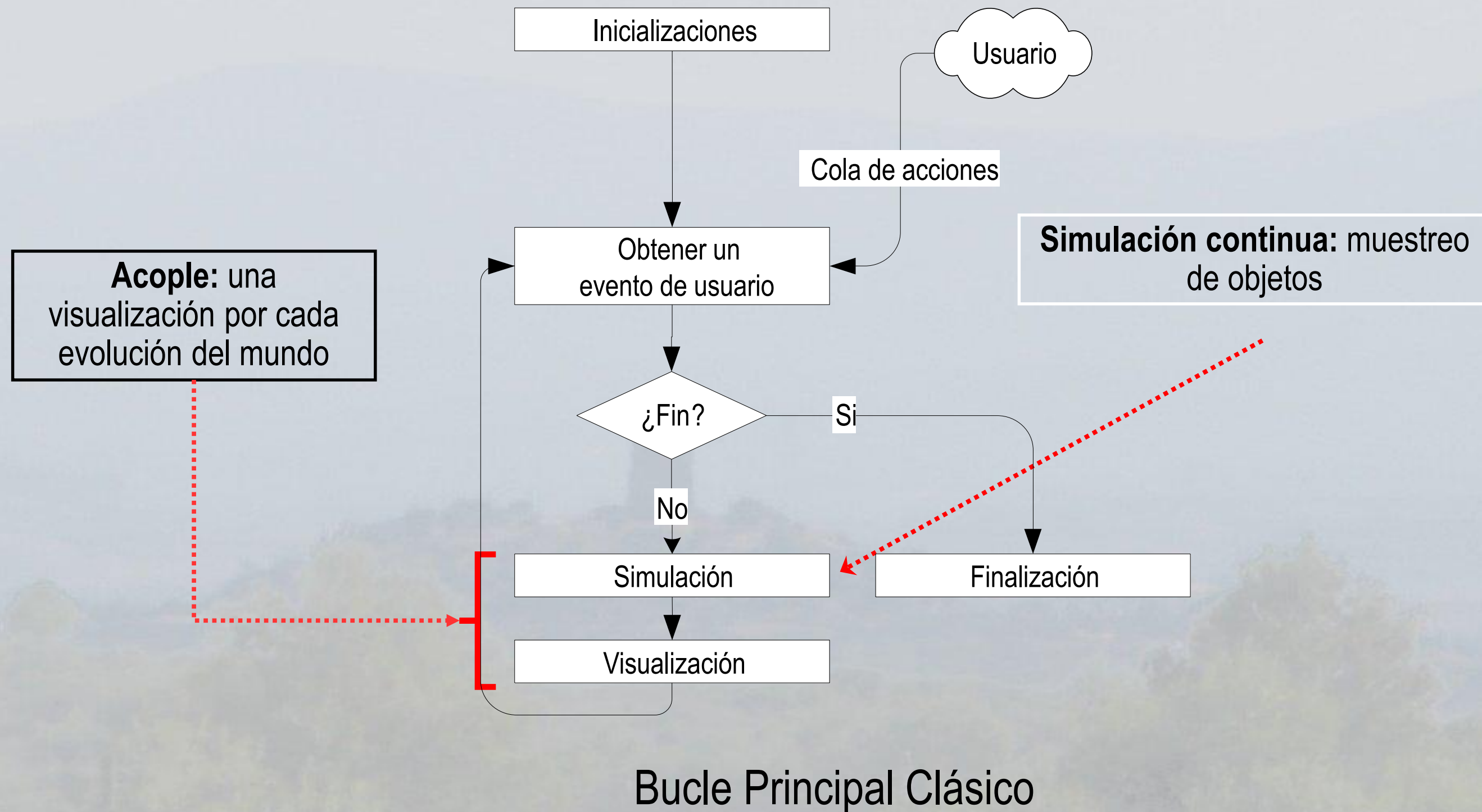
UPV

# Objetivos de aprendizaje

Describir cómo funciona la gestión interna de un videojuego para soportar el tiempo real

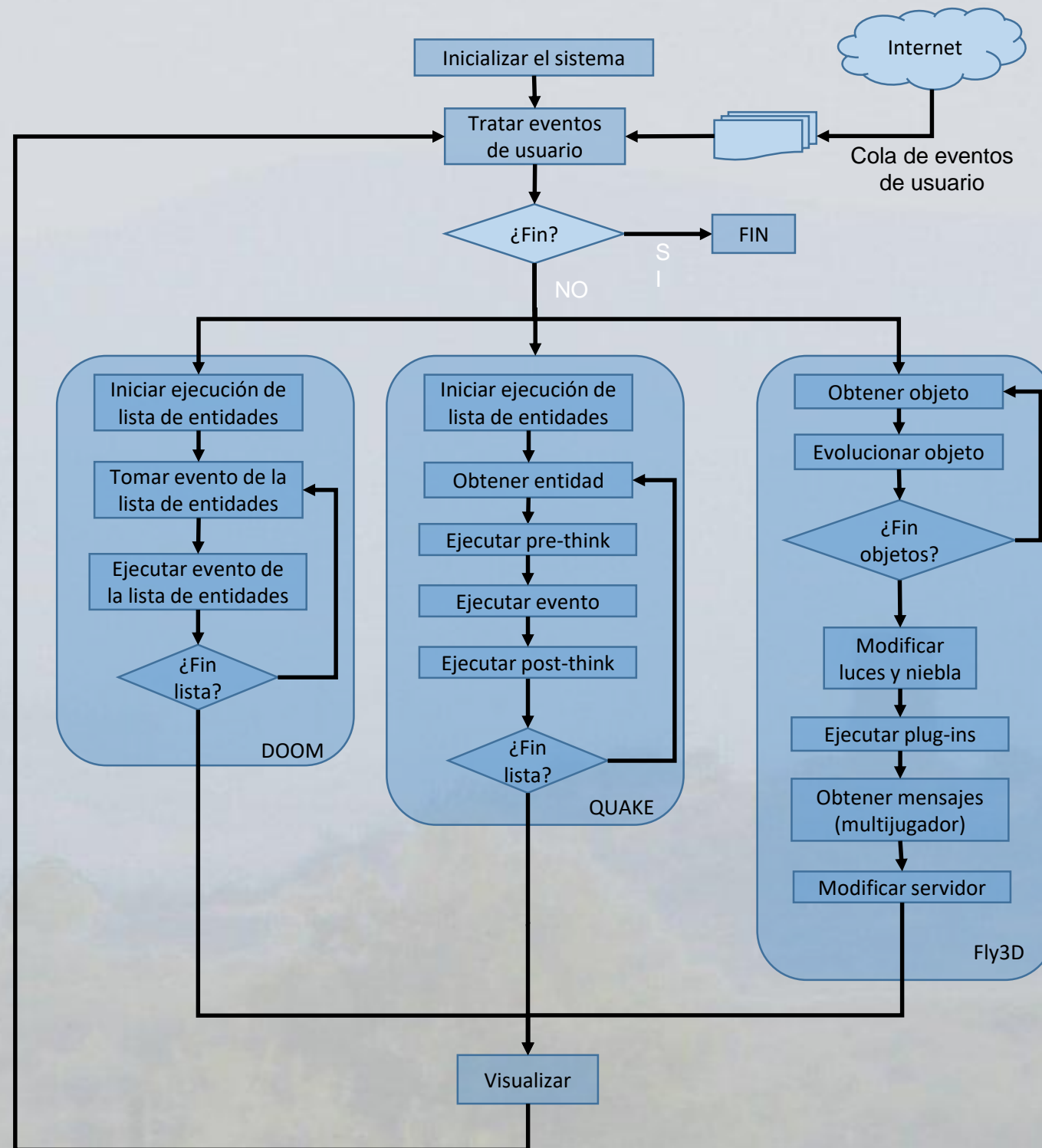
# Unity

## Gestión del tiempo (I)



# Unity

## Gestión del tiempo (II)





# Unity

## Gestión del tiempo (III)

Las unidades de trabajo en **Unity3D**, son metros y segundos

### **Tasa de Refresco (TR)**

Cantidad de veces, por unidad de tiempo, que se

- Actualiza la pantalla de juego
- Recorre el bucle principal de juego

Cada plataforma de juego presenta la suya

Dependiendo de la carga de trabajo tanto del videojuego como de la plataforma, la TR puede variar a lo largo de la ejecución del VJ

### **Objetivo**

Independizar evolución del videojuego de tasa de refresco (TR)

### **Solución**

Supóngase una plataforma que ejecuta el videojuego a 60 fps

El método **Update()** o **Render()** de cada objeto en el sistema se está invocando cada  $1/60$  segundos, por término medio

Es decir cada 16.66 ms

# Unity

## Gestión del tiempo (IV)

Existe una clase genérica **Time** (singleton) que contiene un atributo denominado **deltaTime**

**Time.deltaTime** contiene el tiempo en segundos desde la última vez que se invocó al método **Update()** del **GameObject**

$\text{Time.deltaTime} = 1/(\text{TR}) = 1/60 \text{ s}$

Es decir, que  $\text{Time.deltaTime} * \text{TR} = 1$  **SIEMPRE**, en todo momento y para cualquier plataforma

Si durante la ejecución del videojuego la TR cambia, entonces **Time.deltaTime** también

# Unity

## Gestión del tiempo (V)

Supóngase un objeto A que posee un atributo vector de tres dimensiones (Vector3) denominado *Speed*

*Speed* contiene una velocidad de 2.34 metros por segundo en la dimensión X. Es decir,  $S = (2.34, 0.0, 0.0)$

Existe otro atributo denominado *Position* que almacena dónde está situado el objeto

El método Update de ese objeto se definirá como

```
Vector3 Position = new Vector3();
```

```
void Update() {  
    Position.X = Position.X + Speed.X*Time.deltaTime;  
    transform.Translate(Position);  
}
```

# Unity

## Gestión del tiempo (VI). Ejemplo

El método **GetAxis()** devuelve un valor en el intervalo [-1, +1]. así que si se multiplica por **Time.deltaTime** se puede conseguir que el objeto avance 1 metro por segundo en una dirección u otra sin más que añadir el siguiente código

```
// Update is called once per frame
void Update () {
    transform.Translate(new Vector3(Input.GetAxis("Horizontal") * Time.deltaTime, 0.0f, 0.0f));
}
```



# Bibliografía

Unity Game Development Essentials, Will Goldstone, Ed.  
Packt publishing, Cap. I. ISBN: 978-1-847198-18-1

Unity online manual:

<http://docs.unity3d.com/Manual/index.html>






Documentación generada por  
Dr. Ramón Mollá Vayá  
Sección de Informática Gráfica  
Departamento de Sistemas Informáticos y Computación  
Universidad Politécnica de Valencia

## Reconocimiento-NoComercial-CompartirIgual 2.5

### Usted es libre de:

copiar, distribuir y comunicar públicamente la obra  
hacer obras derivadas bajo las condiciones siguientes:

-  **Reconocimiento.** Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciador.
-  **No comercial.** No puede utilizar esta obra para fines comerciales.
-  **Compartir bajo la misma licencia.** Si altera o transforma esta obra, o genera una obra derivada, sólo puede distribuir la obra generada bajo una licencia idéntica a ésta.

Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra.

Alguna de estas condiciones puede no aplicarse si se obtiene el permiso del titular de los derechos de autor

**Los derechos derivados de usos legítimos u otras limitaciones reconocidas por ley no se ven afectados por lo anterior.**