

# Módulo 8

Código: 489

## Programación multimedia y dispositivos móviles

Técnico Superior en Desarrollo de  
Aplicaciones Multiplataforma



# UNIDAD 6

## Programación multimedia y juegos – Desarrollo de juegos 3D

Contenidos teóricos



1. Objetivos generales de la unidad
2. Competencias y contribución en la unidad
3. Contenidos conceptuales y procedimentales
4. Evaluación
5. Distribución de los contenidos
6. Sesiones

# 1. Objetivos generales de la unidad

## Objetivos curriculares

1. Identificar los elementos principales de un juego.

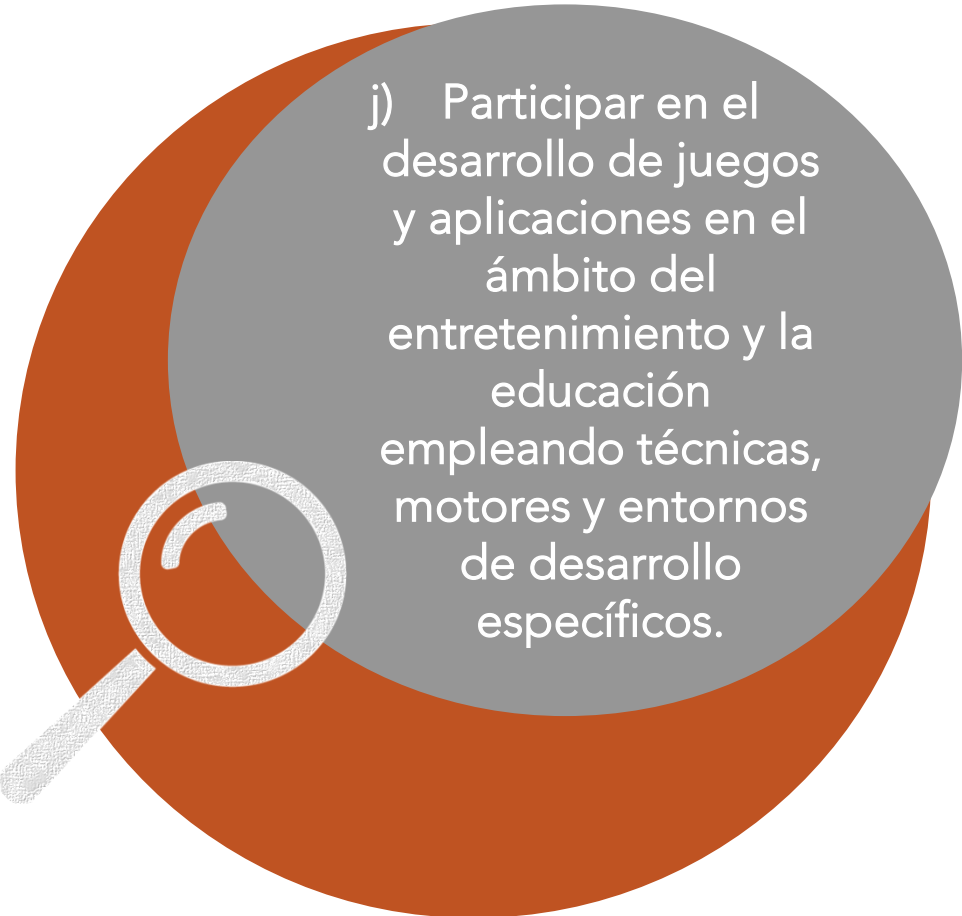
2. Comprobar los beneficios que aportan los motores de juegos.

3. Identificar los elementos principales de un juego.

4. Apreciar las características diferenciales del software para juegos

5. Construir desde cero las funcionalidades básicas de un videojuego.

## 2. Competencias y contribución en la unidad



j) Participar en el desarrollo de juegos y aplicaciones en el ámbito del entretenimiento y la educación empleando técnicas, motores y entornos de desarrollo específicos.

	<p>Aprendiendo a desarrollar un videojuego en 3D, conociendo el entorno de desarrollo junto al motor seleccionado, y usándolo con este propósito.</p>	
--	---	--

# 3. Contenidos

CONTENIDOS CONCEPTUALES	CONTENIDOS PROCEDIMENTALES
<ul style="list-style-type: none"><li>• Modelo</li><li>• Mapeado</li><li>• Joint</li></ul>	<ul style="list-style-type: none"><li>• Animación 3D</li><li>• APIs gráficos 3D</li><li>• Librerías que proporcionan las funciones básicas de un Motor 3D</li><li>• Propiedades de los objetos: luz, texturas, reflejos, sombras.</li><li>• Análisis de ejecución. Optimización de código.</li></ul>

## 4. Evaluación

a) Se han identificado los elementos que componen la arquitectura de un juego 3D

b) Se han creado objetos y definido los fondos

c) Se han instalado y utilizado extensiones para el manejo de escenas

d) Se han utilizado instrucciones para determinar las propiedades finales de la superficie de un objeto o imagen

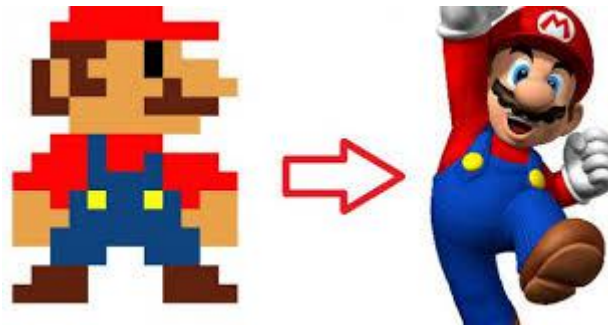
e) Se ha incorporado sonido a los diferentes eventos del juego

f) Se han documentado las fases de diseño y desarrollo de los juegos creados

## 1. DIFERENCIAS CON JUEGOS 2D

Ya hemos visto cómo crear un juego en 2D en la unidad anterior. En esta, se plantea la creación de un juego 3D, que cuenta con una serie de diferencias:

- **Producción:** la producción del videojuego contendrá modelos 3D de los distintos elementos, mientras que en la producción 2D se utilizaban sprites. Además, la música normalmente tendrá mayor calidad y profundidad en el espacio.
- **Realismo visual:** en una producción 3D se tiene que tener más en cuenta la calidad visual, buscar el realismo a la hora de realizar el mapeado, el modelado...
- **Movimiento:** se usan los tres ejes del espacio, mientras que en los videojuegos 2D, el movimiento es bidimensional.





## 2. MOTORES DE VIDEOJUEGOS

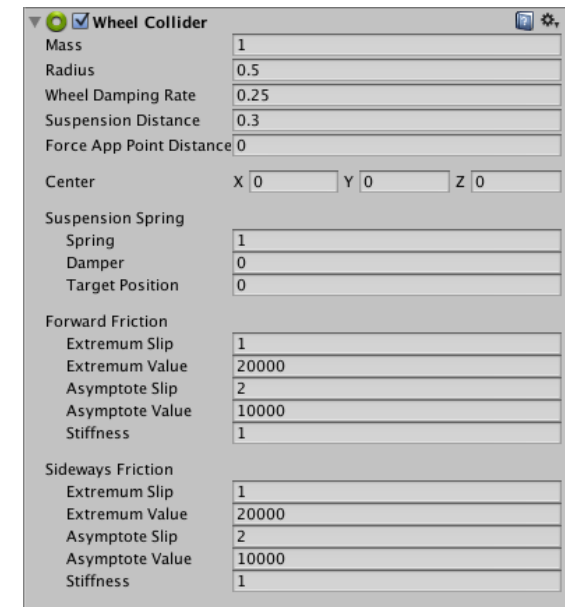
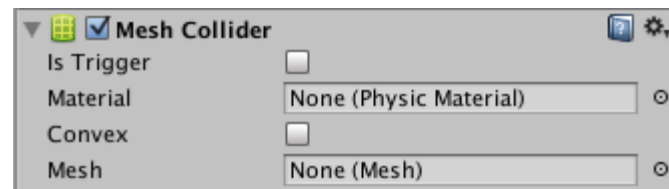
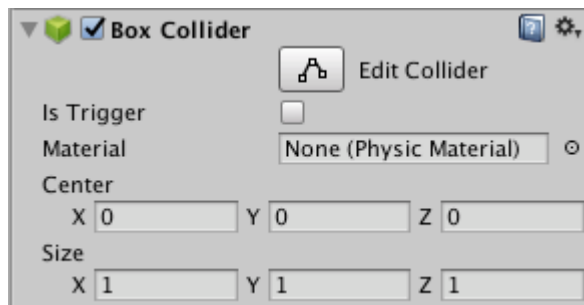
Actualmente, los motores de juegos 3D más utilizados comparten características con los motores 2D, y nos permiten realizar todo el proceso de creación desde el mismo motor. En nuestro caso, seguiremos utilizando Unity, aunque Unreal Engine sería una alternativa más que viable para este cometido. Ambos tienen un entorno de desarrollo completo incluido en el programa.



## 1. FÍSICAS EN 3D

Si bien ya hemos visto en la unidad anterior los componentes relacionados con las físicas 2D, veremos cómo se extienden para el uso en elementos 3D. Disponemos de los siguientes:

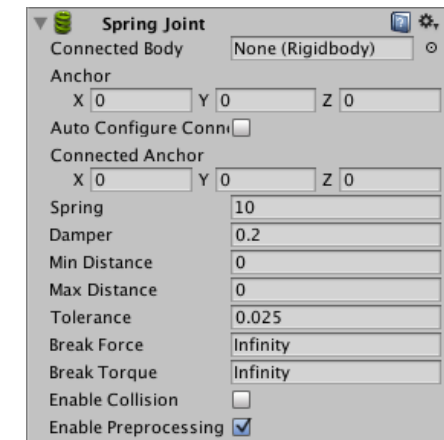
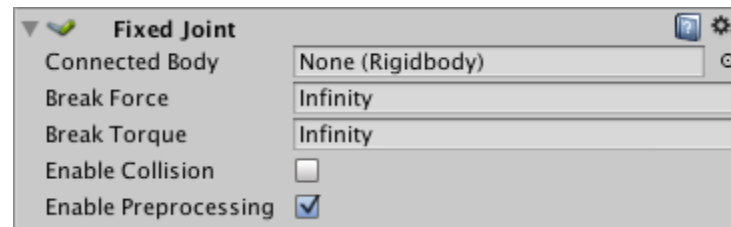
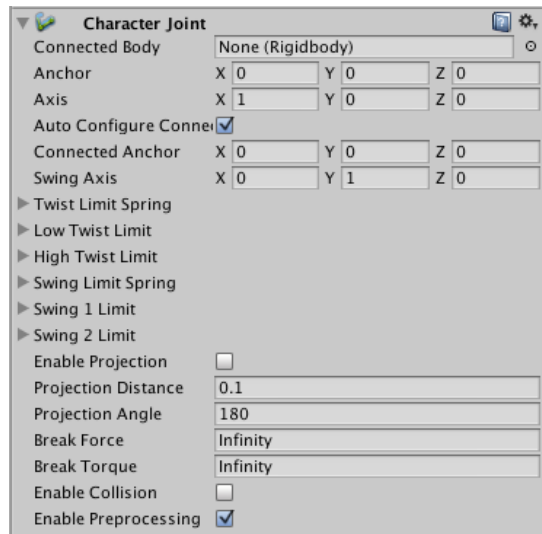
- **Box Collider:** permite establecer un collider 3D en forma de caja.
- **Mesh Collider:** permite ver si existen colisiones con un Mesh.
- **Sphere Collider:** collider en forma de esfera.
- **Wheel Collider:** collider en forma de rueda, pensado para juegos de conducción.
- **Terrain Collider:** toma colisiones con la forma del terreno asociado al objeto.



## 1. FÍSICAS EN 3D

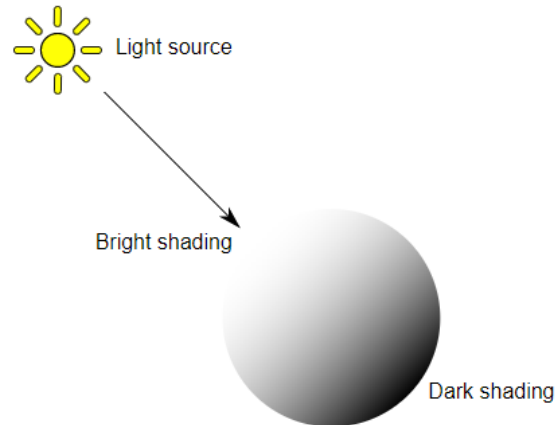
Además de los colliders, tenemos diferentes joints para restringir el movimiento de un objeto respecto a otro:

- **Character Joint:** para crear físicas similares a un animal.
- **Configurable Joint:** nos da muchas opciones de configuración.
- **Fixed Joint:** para unir el comportamiento de dos objetos.
- **Hinge Joint:** agrupa dos Rigidbody y se suele usar en puertas, péndulos...
- **Spring Joint:** agrupa dos Rigidbody, pero permite cierta distancia entre ellos.



## 2. ILUMINACIÓN

Para calcular el sombreado de un objeto 3D, Unity necesita conocer la intensidad, dirección y color de la luz que incide sobre éste.



Estas propiedades son proporcionadas por objetos Light en la escena. Los varios tipos de luz emiten su color asignado de manera diferentes; algunas luces disminuyen con la distancia desde la fuente, y tienen diferentes reglas acerca del ángulo de luz recibida desde la fuente.

## 2. ILUMINACIÓN

Técnicas de iluminación:

- **Iluminación en tiempo real:** se actualizan a cada frame. Es la manera más básica de iluminar objetos dentro de la escena y es útil para iluminar personajes u otra geometría movable.
- **Efectos de iluminación estáticos complejos:** conocido en inglés como baking lightmap, consiste en preprocesar fuentes de luz estáticas y se superponen a las texturas donde incidirían. Por ejemplo, en la siguiente imagen tenemos a la izquierda una escena simple lightmapped, mientras que a la derecha, la textura lightmap generada por Unity.
- **Precomputed realtime global illumination:** nos permite actualizar los efectos estáticos dentro de una escena.

