



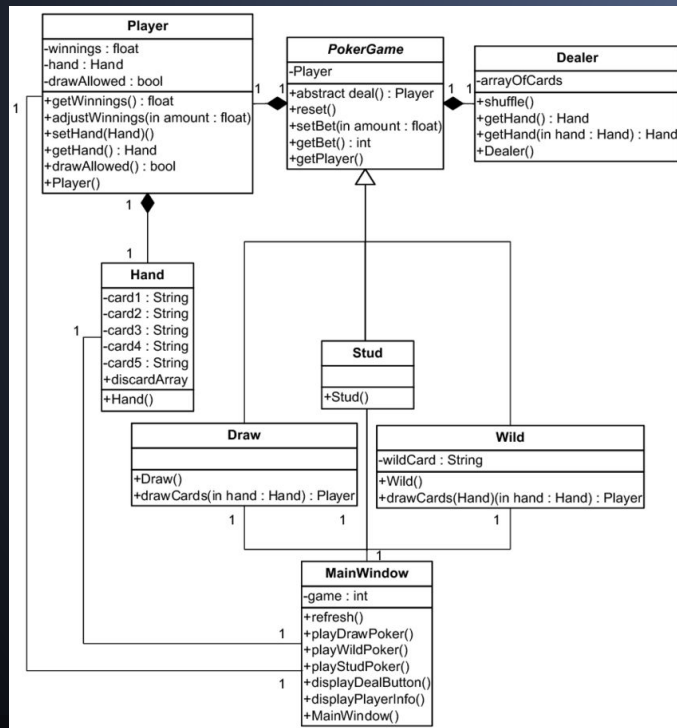
Desarrollo de Videojuegos

Introducción

Producción y proceso de desarrollo

Motivación

- El videojuego es a la vez **software** y **obra audiovisual**
 - Lo más *genérico* se asocia al desarrollo de **jugabilidad** y lo más *específico* al desarrollo de **contenido**



Contenido de un videojuego

- Ejemplo: **Matamarcianos**

El jugador controla una nave espacial que puede mover en dos dimensiones desde una vista cenital. La nave avanza por el espacio, atravesando zonas con asteroides y naves enemigas, evitando chocar con ellos y con los proyectiles enemigos a la vez que puede dispararles para tratar de destruirlos. En algunas zonas hay cápsulas que si chocan con la nave del jugador duplican su amplitud de disparo hasta un límite de tres duplicaciones. Cuando la nave choca tres veces la partida se reinicia, pero si se llega a la última zona, aparece un alienígena gigante, y si es destruido el jugador gana la partida.



Contenido de un videojuego

- Mundo y localizaciones
 - Conseguir la *estética* del **espacio exterior**, ofreciendo una vista cenital de un escenario 2D
 - Hacer que funcione la *mecánica* de que chocar tres veces (regla) supone reiniciar
 - Explicar al jugador la *dinámica* de ir controlando la nave con el objetivo de llegar hasta el final
 - Definir la *mecánica* de las **zonas**, localizaciones donde se distribuye todo el contenido y que vamos atravesando

Contenido de un videojuego

- Objetos y máquinas
 - Crear la **nave del jugador**, una máquina (vehículo) que controla el jugador, con acciones como moverse y disparar (sistemas del jugador)... lo que implica generar **proyectiles**
 - Añadir **asteroides**, que reaccionan a los proyectiles del jugador, siendo destruidos
 - Añadir **cápsulas**, reaccionan con la nave del jugador con la regla/restricción de duplicar la amplitud del disparo hasta tres veces
 - Completar los actores con **naves enemigas**, con acciones y reacciones propias (sistemas enemigos)

Contenido de un videojuego

- Criaturas y personas
 - Alienígena, tal vez animado con tentáculos, con su propia *dinámica* a la hora de jugar y quizá con *mecánicas* especiales de combate

Desarrollo de contenido

- En la primera *iteración*, si hay desarrollo es todavía **esquemático** (a nivel de **bloques**)
 - Entidades con algo de **geometría**, **material** y **luz**
 - También algo de jugabilidad, **comportamientos** reusables e independientes de cualquier tema



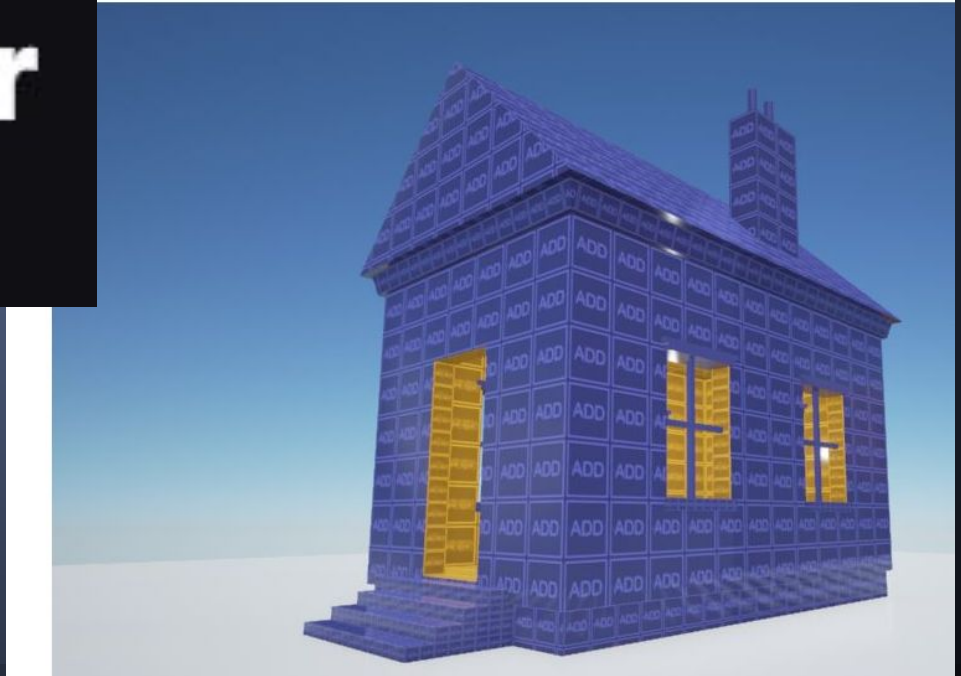
Escenas y entidades

Geometry Brush Actors

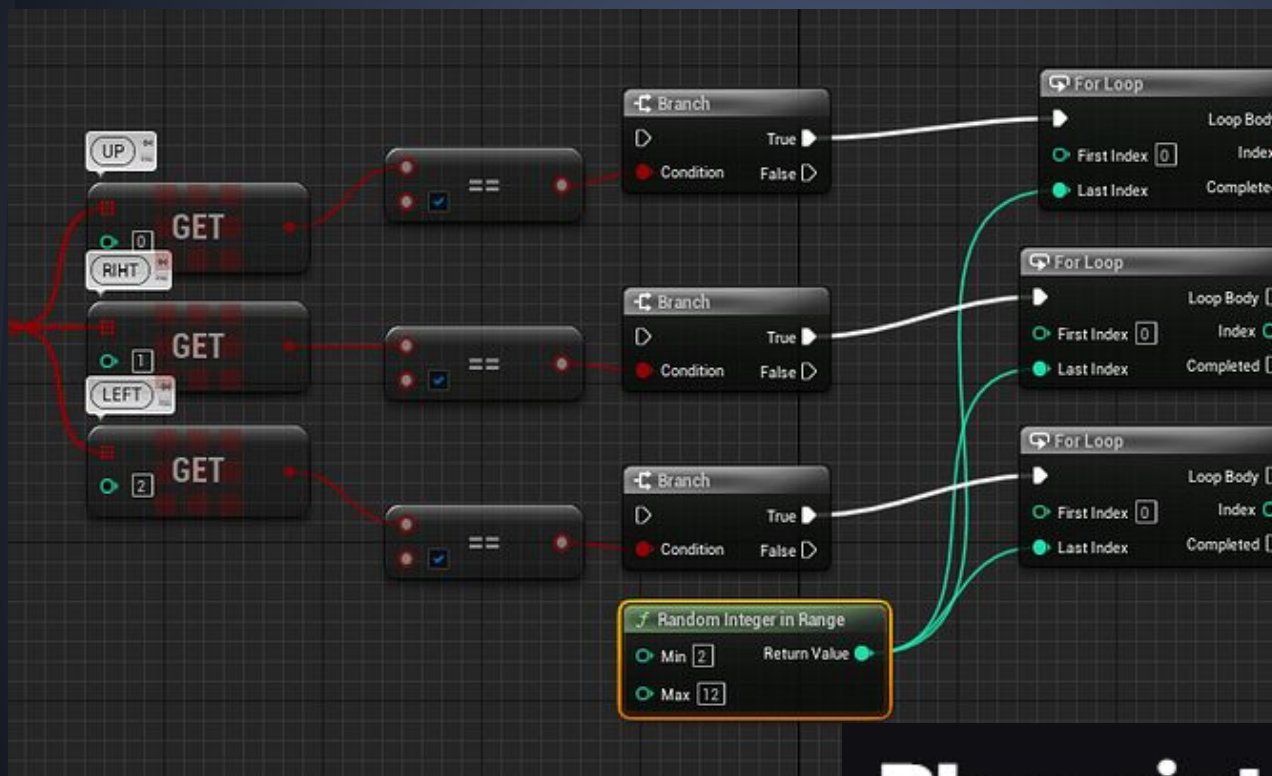
Guide to using Brushes to create level geometry in Unreal Editor.

Unreal Engine 4.23

Level Editor Gameplay Levels



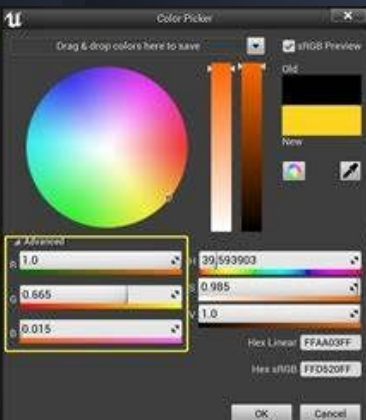
Eventos y comportamientos



Blueprint Editor
Blueprints

Desarrollo de contenido

- En las iteraciones de la fase de desarrollo se añaden elementos nuevos
 - **Mallas iniciales** (menor detalle que las finales)
 - **Recursos físicos** (para mejorar la jugabilidad)
 - **Iluminación inicial** con cierto colorido (por temas de dinámica - para **orientar al jugador**-)
 - **Otros elementos** como la niebla (relevantes sólo si afectan a la jugabilidad/visibilidad)



Colisiones

Physics Asset Editor

Physics

Collision Overview

An overview of how Collision and Collision Responses operate in Unreal Engine 4.

Unreal Engine 4.22

Collision Responses and **Trace Responses** form the basis for how Unreal Engine 4 handles collision and ray casting during run time. Every object that can collide gets an **Object Type** and a series of responses that define how it interacts with all other object types. When a collision or overlap event occurs, both (or all) objects involved can be set to affect or be affected by blocking, overlapping, or ignoring each other.

Trace Responses work basically the same way, except the trace (ray cast) itself can be defined as one of the Trace Response types, thusly allowing Actors to either block it or ignore it based on *their* Trace Responses.

Interactions

Desarrollo de contenido

- Para las **mallas iniciales**, o cuentas con una buena dirección artística o usas un *pack* coherente de recursos ya existentes
 - Con las mallas puestas se aprecia el **tema** del nivel



Mallas estáticas

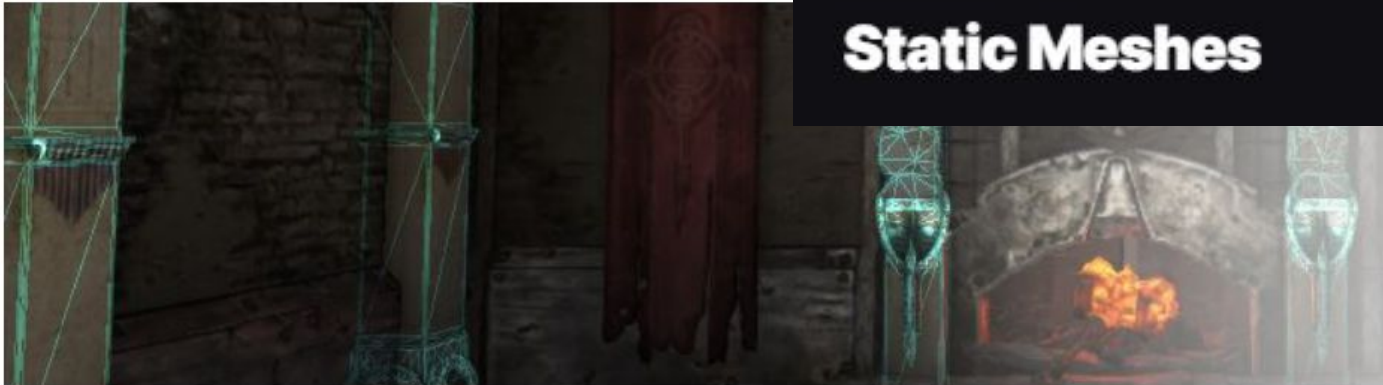
Static Meshes

Static geometry which can be cached in video memory and rendered by the graphics card.

Unreal Engine 4.9

Static Mesh Editor

Static Meshes



Participación

- ¿Qué *elementos del escenario* necesitamos en las primeras iteraciones del desarrollo?
 - A. Bloques, algunas mallas y actores, luz de trabajo
 - B. Terreno, vegetación y decorados urbanísticos
 - C. Geometría, comportamientos de los actores
 - D. Mallas básicas, actores y efectos visuales
 - E. Mecánicas básicas y estética general



Desarrollo de contenido

- Para la **iluminación inicial** se retira la básica (la de trabajo) y se colocan luces buscando ya la *estética deseada* para el juego
 - Si cambia la iluminación, cambian los **materiales**
 - También se pueden aplicar **efectos de postprocesado** (Ej. profundidad de campo) y algunos **sistemas de generación de partículas** (Ej. fuego de antorchas)



Iluminación

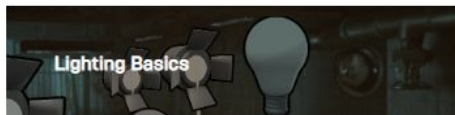
Lighting the Environment

Using Lighting and shadowing geometry, using Global Illumination, and setting up reflections.

Unreal Engine 4.18

Material Editor Materials

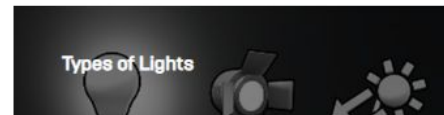
Essentials



The basics of placing lights into levels and setting them up.



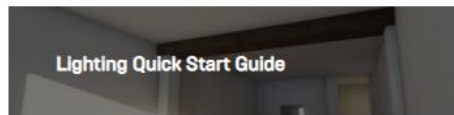
Light Mobility and how it affects lighting and shadowing.



The types of lights available in Unreal Engine 4.



High level overview of shadows.

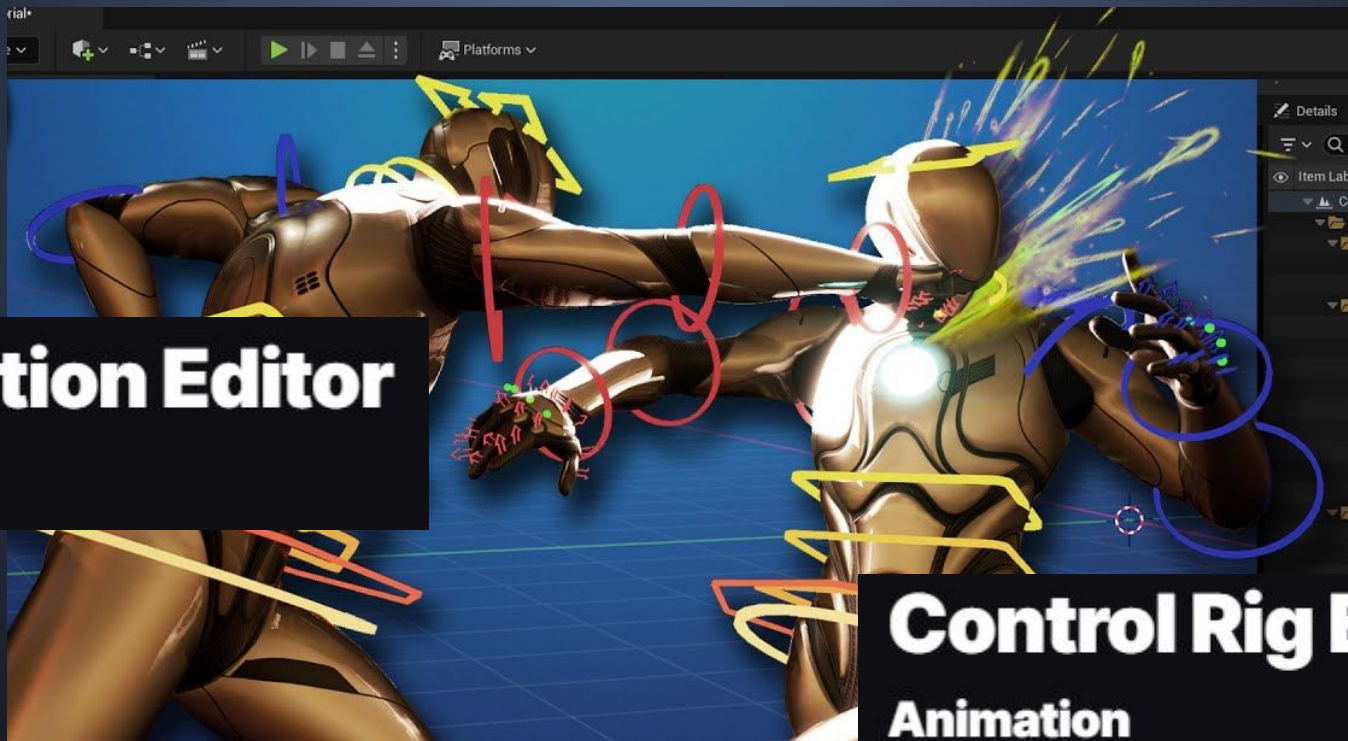


In this guide you will create and light a small apartment using different types of lighting techniques.

Desarrollo de contenido

Sound Cue Editor
Sound Cues

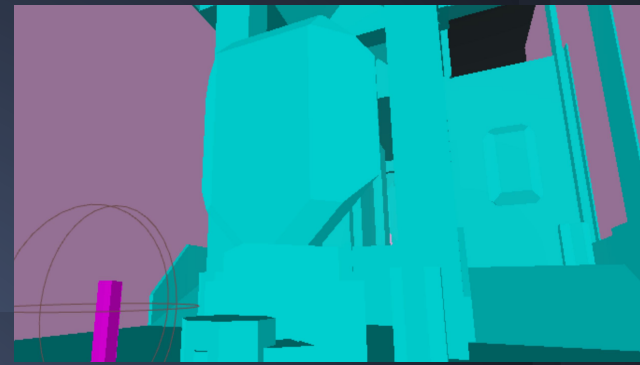
- Se trabajan más las **animaciones**, con todos los elementos que incluyen... el sonido



Animation Editor
Animation

Control Rig Editor
Animation

Desarrollo de contenido



- Tras colocar mallas definitivas, comprobar:
 - Que no haya *fallos de colisiones* en ninguna parte
 - Que la IA y la dificultad sean correctas
 - Que haya menús e interfaces correctas
 - Que el juego funcione en máquinas de requisitos mínimos, haciendo *perfiles de rendimiento*



UMG UI Editor
User Interface

Behavior Tree Editor
AI Behavior



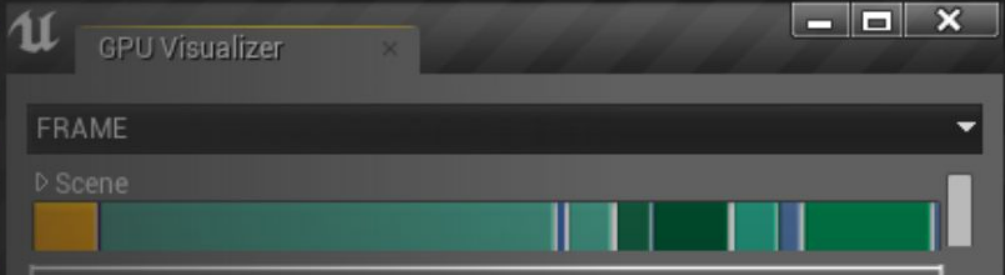
Producción y proceso de desarrollo

Rendimiento

Performance and Profiling

How to identify performance problems and fix them.

Unreal Engine 4.9



Frame: 16.13 ms
Game: 9.35 ms
Draw: 0.29 ms
GPU: 16.14 ms

Performance is an omnipresent topic in making real-time games. In order to create the illusion of moving images, we need a frame rate of at least 15 frames per second. Depending on the platform and game, 30, 60, or even more frames per second may be the target.

Unreal Engine provides many features and they have different performance characteristics. In order to optimize the content or code to achieve the required performance, you need to see where the performance is spent. For that, you can use the engine profiling tools. Every case is different and some knowledge about the internals of hardware and software is needed.. The Performance and Profiling pages linked below will guide you through profiling your project.

Desarrollo de contenido

- Finalmente llega la fase de **pulido**, iteraciones para dar los últimos toques
 - Reflejos en algunas superficies
 - Haces de luz
 - Más efectos especiales decorativos
 - Revisar textos, cinemáticas o videos
 - ...

Media Editor
External Media Playback

Font Editor
Fonts



Efectos especiales

Reflection Environment

System for capturing and displaying local glossy reflections.

Unreal Engine 4.8

The **Reflection Environment** feature provides efficient glossy reflections in every area of the level. Many important materials like metals rely on having reflections in all directions, which the Reflection Environment provides. It is targeted toward consoles and mid spec PC, so it must run very fast. Support for reflections of dynamic objects or sharp reflections is supported but require additional memory overhead.



Fog Sheet and Light Beams

An overview of the Fog Sheet Blueprint.

Unreal Engine 4.9



Particle Effects

An overview of the Effects example included with UE4.

Unreal Engine 4.9



Niagara Editor Particle Effects

Desarrollo de contenido

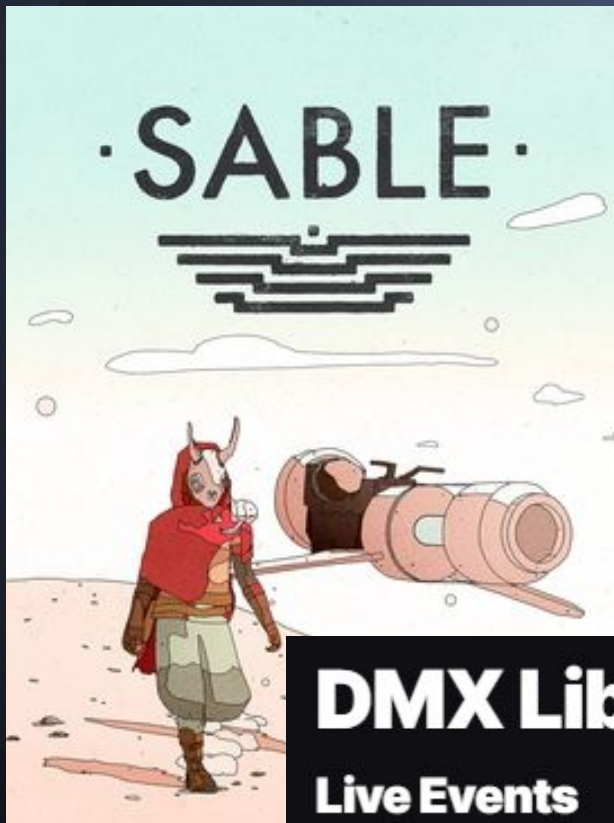
- En la práctica, los recursos que usamos puede provenir de muchas **fuentes**:
 - El propio IDE (*Unreal Engine Content*)
 - Propietarios del IDE (*Samples, Content Examples* y otros materiales de *Epic Games...*)
 - Terceras empresas (*Fab.com / Unreal Engine Marketplace...* otras tiendas o sitios web)
 - Mundo real (*Quixel MegaScans* y otros)
 - IA Generativa
 - Nosotros (del proyecto actual... o anteriores)
 - Nuestros propios usuarios / jugadores

nDisplay 3D Config Editor

Virtual Production and Live Events

Y mucho más

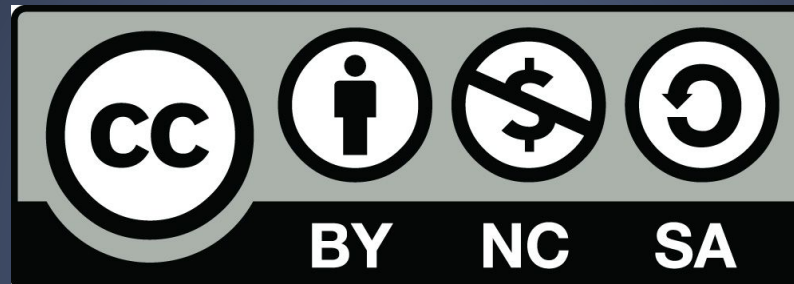
- ¡No sólo se hacen juegos realistas para ver en una tele!



DMX Library Editor

Live Events

Críticas, dudas, sugerencias...



* Excepto el contenido multimedia de terceros, como las imágenes o el material original de *Desarrollo de Juegos con UE4* creado por Epic Games

Federico Peinado (2019-2026)

www.federicopeinado.es

