



# Desarrollo de Videojuegos

Cuestiones avanzadas  
Optimización del rendimiento

# Motivación

- Buscamos **riqueza de entidades** en el mundo, **potencia audiovisual**, **fluidez...** pero todo funcionando en HW modesto



# Cuestiones habituales

- ¿Qué es “mejor”, Blueprints o C++?
  - Cada uno tiene sus *pros y contras*, pero hay juegos profesionales con 20% de Blueprints (Fortnite 60%)
- ¿Cómo asegurarnos de que el juego no presenta errores que lo hagan injugable?
  - Debemos usar bien las herramientas de depuración
- ¿Qué maneras hay de optimizar el rendimiento en Unreal Engine?
  - Muchas veces la clave está en *entender el problema*, lo que hace el motor por debajo y lo que le “cuesta” hacerlo



# Programación visual vs. textual

- Las clases Blueprint están **pensadas para usar clases de C++ (herencia)** y no al revés
  - Para llamar a código Blueprint desde C++ habría que usar **eventos** o **delegados**



# Programación en C++

```
#pragma once
```

```
#include "GameFramework/Actor.h"
```

```
#include "MyActor.generated.h"
```

```
UCLASS()
```

```
class AMyActor : public AActor
```

```
{
```

```
    GENERATED_BODY()
```

```
public:
```

```
    // Sets default values for this actor's properties
```

```
    AMyActor();
```

```
    // Called when the game starts or when spawned
```

```
    virtual void BeginPlay() override;
```

```
    // Called every frame
```

```
    virtual void Tick( float DeltaSeconds ) override;
```

```
};
```



<https://docs.unrealengine.com/en-US/Programming/Introduction/index.html>

# Programación en C++



Visual Studio 2019  
Professional



## ReSharper C++

**2020.3** RELEASED!

The Perfect Game Dev Companion  
for Unreal Engine



# Programación en C++

- Una jerarquía de **más de 4000 clases...**
- UObjectBase, interna (no debe usarse)
  - UObjectBaseUtility, interna (no debe usarse)
    - UObject, la clase base de todo objeto de UE4
      - AActor, objeto que se puede poner en la escena
      - UActorComponent, los componentes son subobjetos que se registran como partes de los actores (sin transformada ni geometría)
        - USceneComponent, se añade la transformada (uno de ellos actúa como componente raíz de los demás del mismo tipo en el actor)
          - UPrimitiveComponent, se añade la geometría/render
      - UWorld, mundo virtual posiblemente compuesto de varios niveles
      - UWorldComposition
      - ULevel, nivel con el que trabajamos con el editor
      - UField
        - UStruct
          - UClass
            - ...

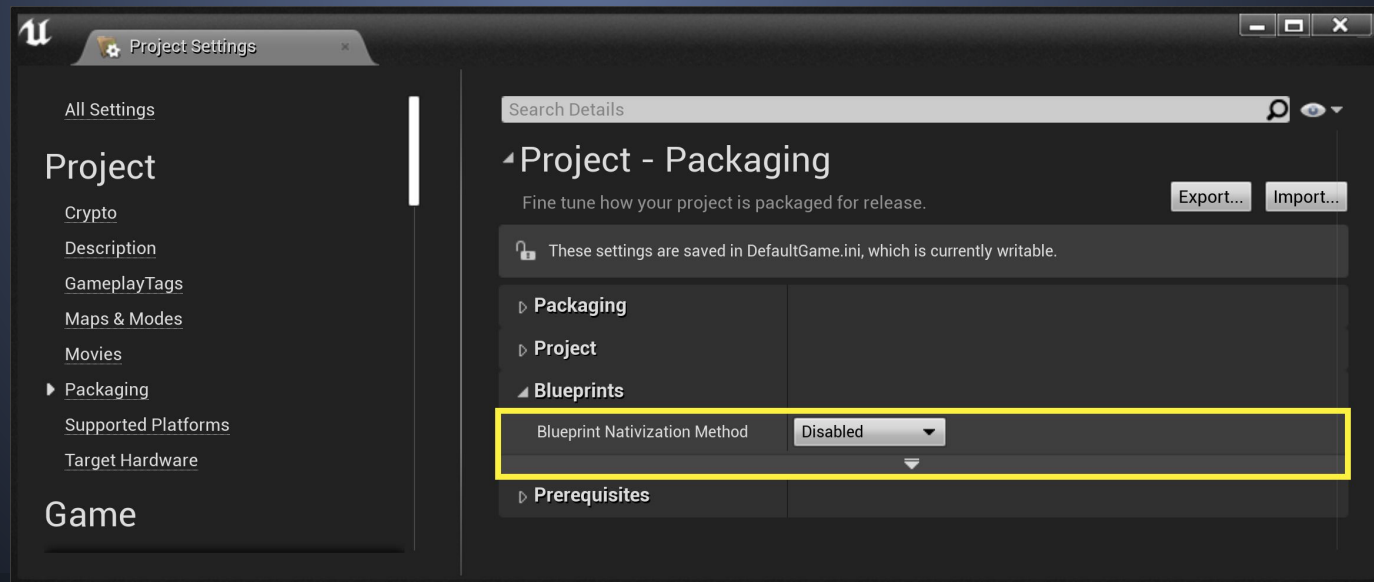
# Blueprints vs C++

- Ventajas y desventajas
  - En **Blueprints** todos los perfiles de desarrollador pueden colaborar de forma segura y controlada, se ve más claro el flujo del juego, y es más rápido hacer cambios y probar cosas nuevas (prototipos)
  - En **C++** tenemos más control sobre la API, los datos, el multijugador; ejecuta más rápido y es más cómodo para, por ejemplo, cálculos matemáticos o trabajar colaborativamente en un repositorio

<https://docs.unrealengine.com/en-US/Resources/SampleGames/ARPG/BalancingBlueprintAndCPP/index.html>

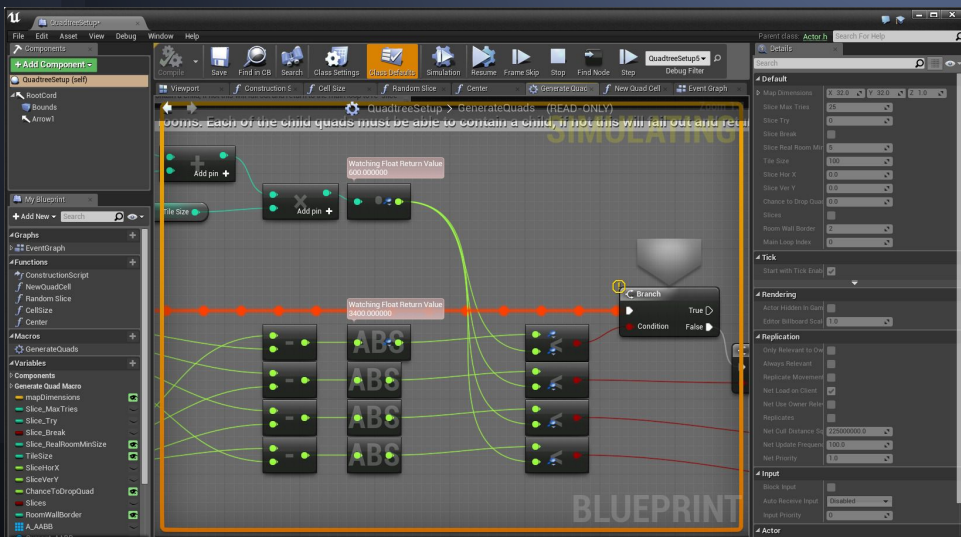
# Nativización de Blueprints

- A la hora de “empaquetar” el producto final, es posible pedirle a Unreal que **convierta el código Blueprints a C++ (código nativo)**, mejorando su rendimiento



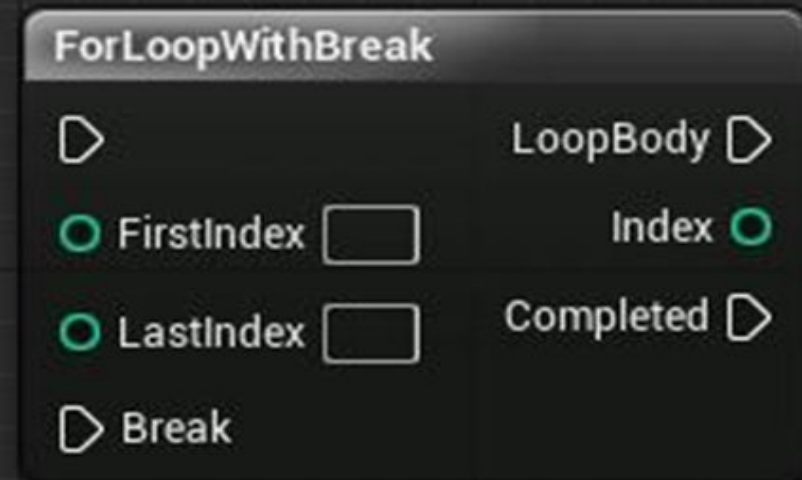
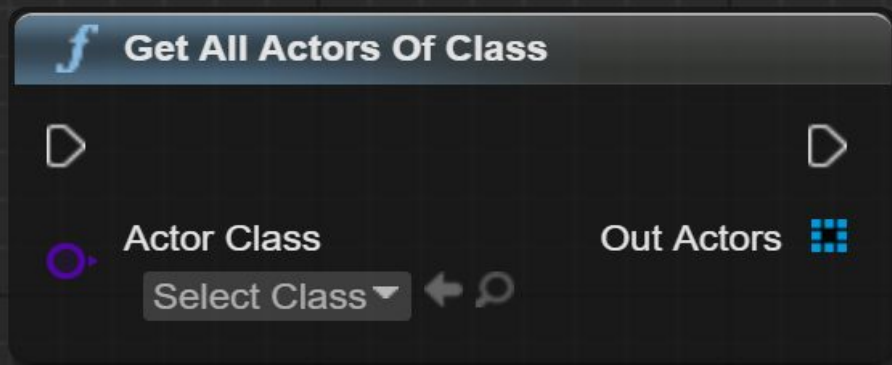
# Depuración de errores

- Conviene recordar lo cómodo que es **depurar visualmente** con Blueprints
  - Vemos la ejecución y por supuesto se pueden poner **puntos de ruptura, inspeccionar variables, avanzar fotograma a fotograma**, se pueden ver las mallas de colisión, info. de la IA...



# Malas prácticas

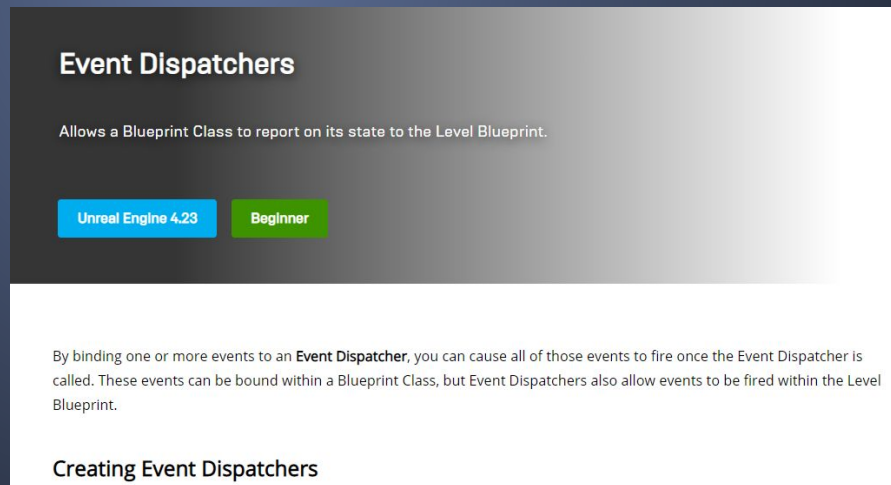
- Hay que **evitar malas prácticas** que causen estragos en el rendimiento, iterando mucho en **Event Tick**, usando **Get All...** varias veces, crear y destruir mucho, y cosas así



# Despachadores de eventos

¡Para evitar búsquedas!

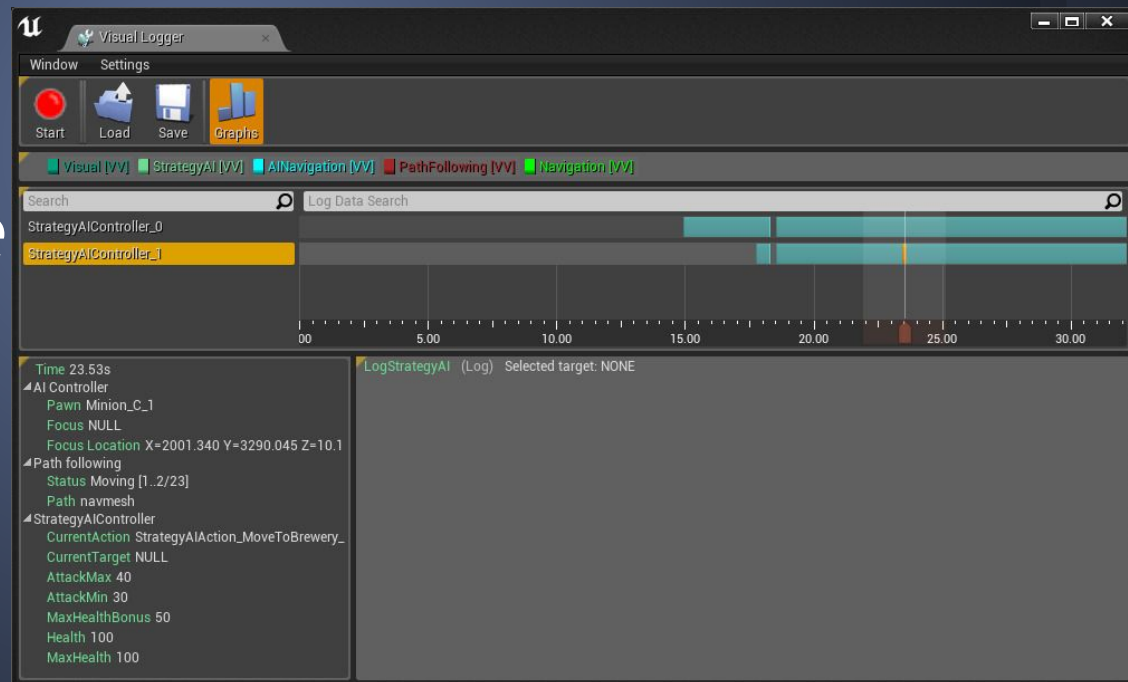
- Son **eventos especiales** que al llamarlos, llaman directamente a todos los eventos (de cualquier otro actor) que estén en la lista de **eventos vinculados** al despachador



<https://docs.unrealengine.com/en-US/ProgrammingAndScripting/Blueprints/UserGuide/EventDispatcher/index.html>

# Registrador visual

- Existe una herramienta muy útil para encontrar fallos del estado del juego, con información visual que puede revisarse *a posteriori*



<https://docs.unrealengine.com/4.27/en-US/TestingAndOptimization/VisualLogger/>

# Pruebas automáticas

- También hay herramientas de **pruebas automáticas** (específicas para jugabilidad)

## Automation System Overview

Overview of the automation system used for unit testing, feature testing, and content stress testing.

Unreal Engine 4.17

The Automation System is built on top of the Functional Testing Framework, which is designed to do gameplay level testing, which works by performing one or more automated tests. Most tests that are written will be functional tests, low-level core or Editor tests that need to be written for using the Automation Framework system. These tests that are written can be broken down into the following categories depending on their purpose or function:

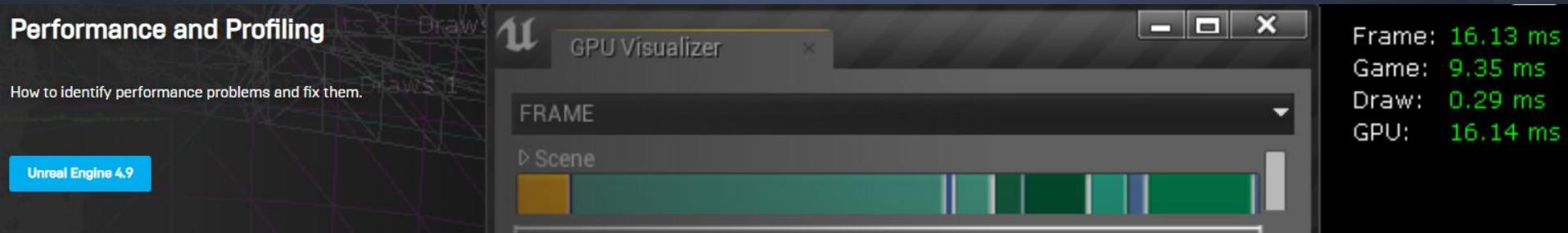
<https://docs.unrealengine.com/en-US/Programming/Automation/index.html>

# Participación

- ¿Qué “mala práctica” hay que evitar para **mantener buen rendimiento** en el juego?
  - A. Programar todo en Blueprints
  - B. Repetir el uso de GetAll, usar mucho el Tick...
  - C. Usar el spawn actor
  - D. Usar mucho el Tick y no depurar visualmente
  - E. Programar todo en C++



# Optimización del rendimiento



**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.



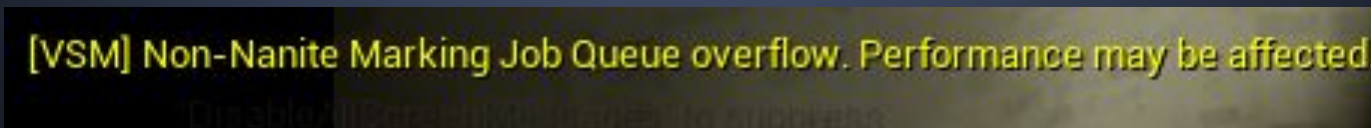
Brickadia (Brickadia, LLC)

# Optimización del rendimiento

- Recuerda que la calidad de lo visto en el editor es *escalable* por si trabajas en un PC poco potente



- Otros mensajes pueden ser por tener muchas luces y mallas en pantalla interactuando (sin usar Nanite)

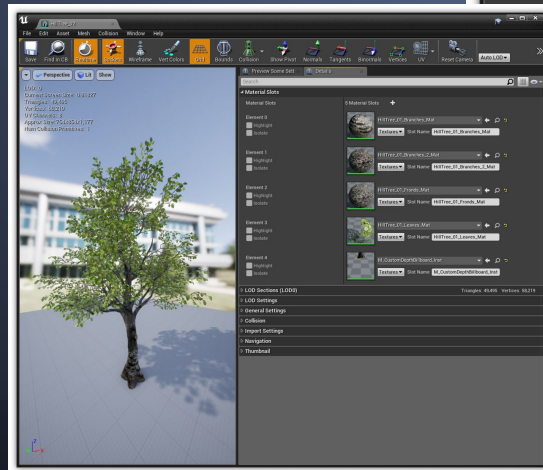
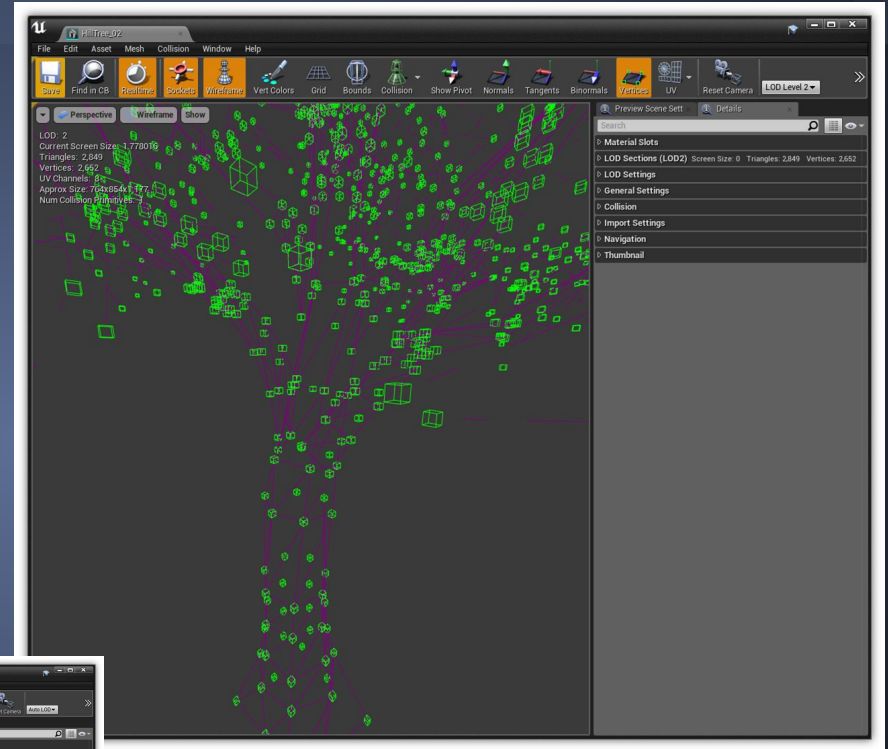


# Optimización del rendimiento

- Un paso previo a todo esto es probar el **ejecutable Shipping** sin nada más arrancado ...y confirmar que NO estamos limitando *a propósito* los FPS
  - `r.VSync`
  - `t.MaxFPS`
  - `bSmoothFrameRate`
  - ...

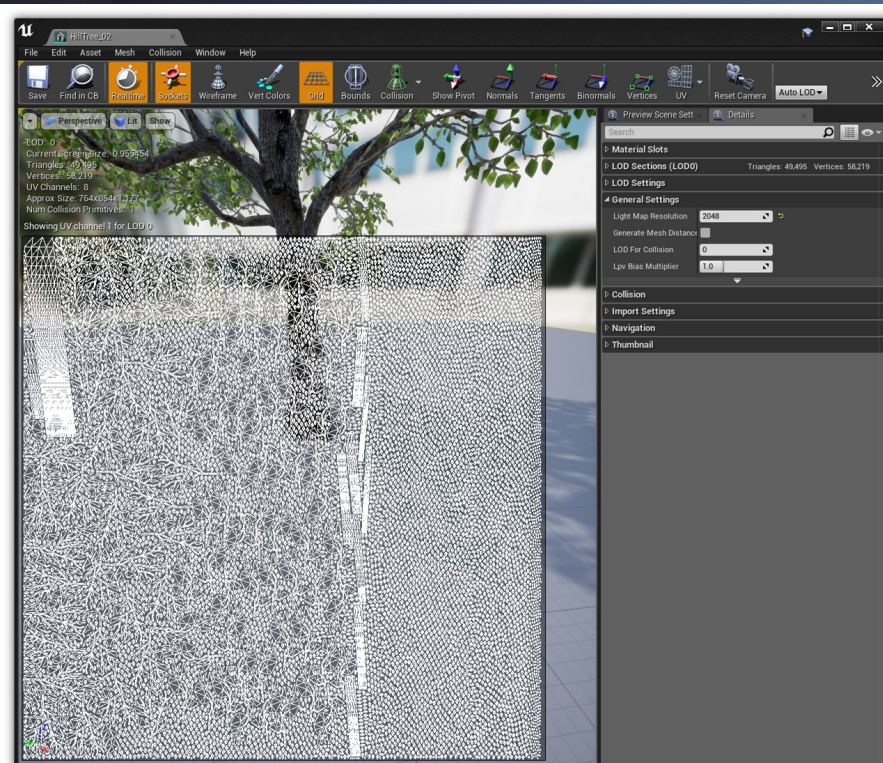
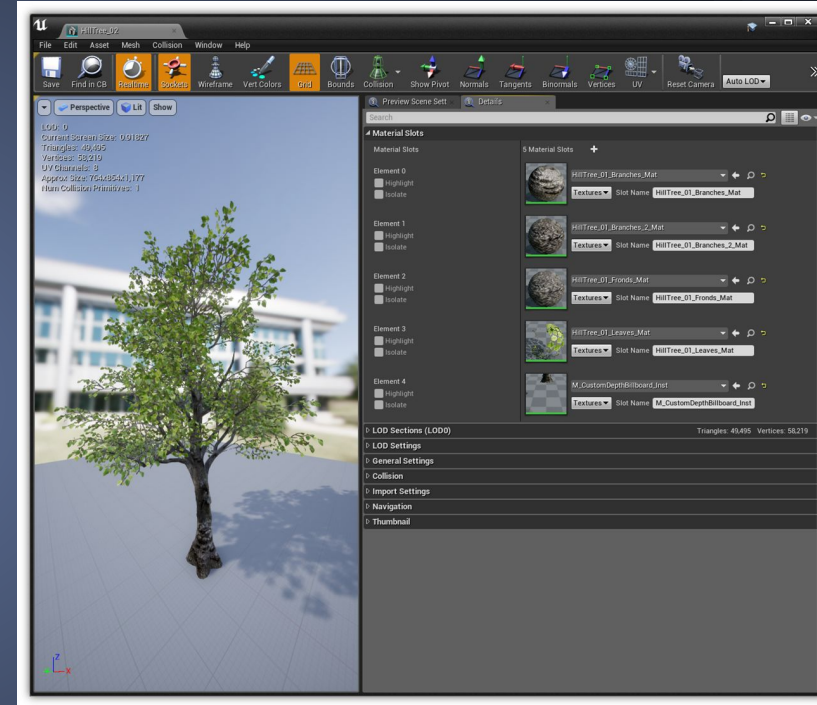
# Optimización

- Conteo de vértices
- Niveles de detalle (LOD)



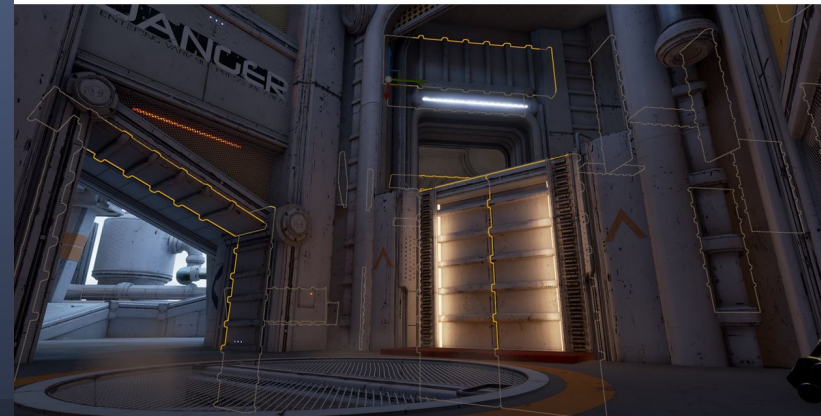
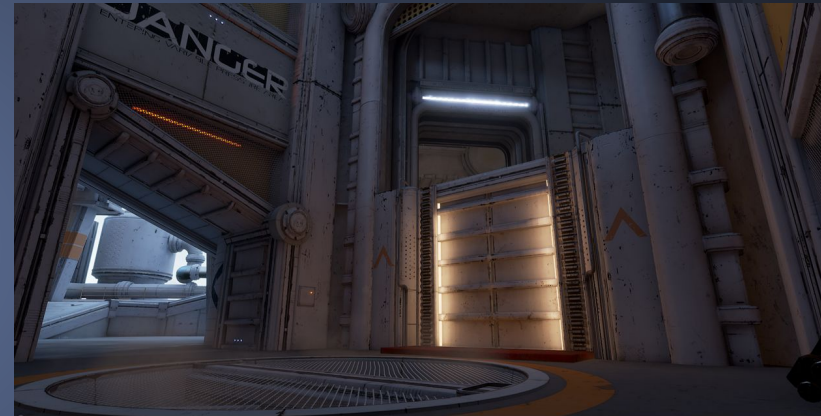
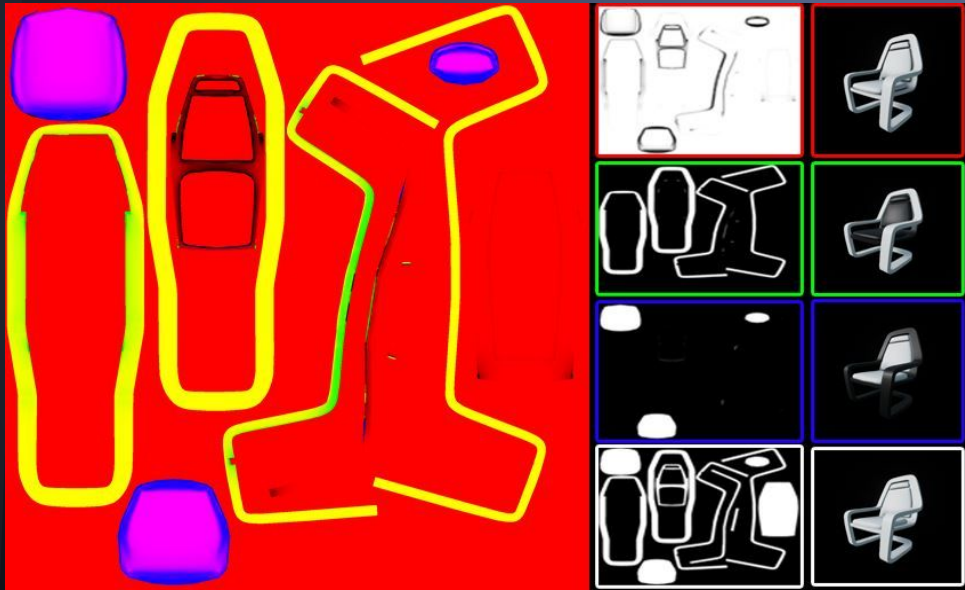
# Optimización

- Conteo de materiales
- Resolución de mapa de luces



# Optimización

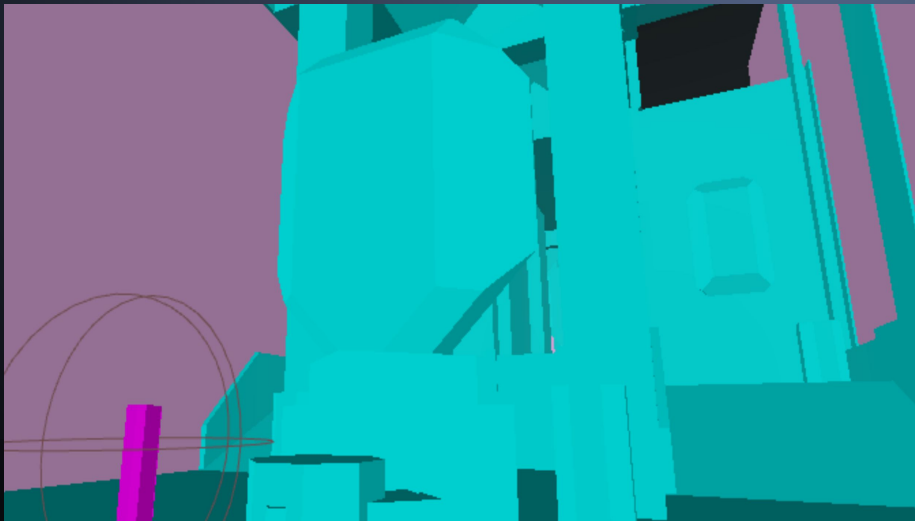
- Empaquetado de texturas
- Reutilizando recursos



# Perfiles de rendimiento

*Como el Foliage...*

- Tras colocar mallas definitivas, *agruparlas* (menos **DrawCalls**) y comprobar
  - Que no haya *fallos de colisiones* en ninguna parte, para que los personajes puedan moverse bien
  - Que el juego funcione en máquinas de requisitos mínimos, haciendo *perfiles de rendimiento*



# Profiling

- Análisis para obtener un **perfil de la ejecución**

- FRAME** ○ Tiempo de **Fotograma** = Tiempo **CPU** + Tiempo **GPU**
- Tiempo **CPU** = Tiempo Lógica y Física **Juego** + Tiempo Llamadas **Dibujado**  
**GAME** **DRAW**

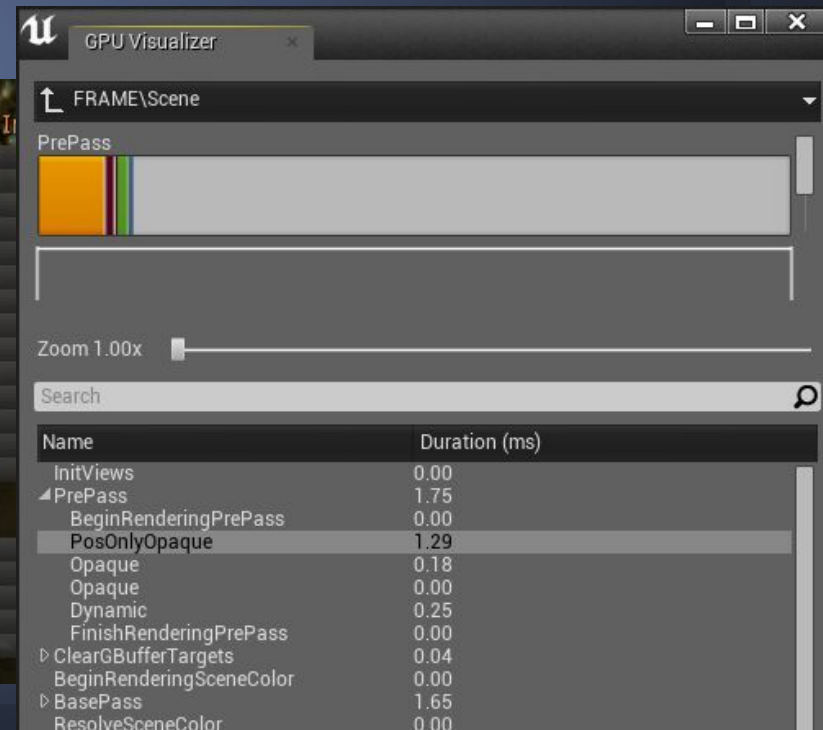


Scene Rendering [STATGROUP\_SceneRendering]

| Cycle counters (flat)     | CallCount |
|---------------------------|-----------|
| Base pass drawing         | 1         |
| BeginOcclusionTests       | 1         |
| Cache Uniform Expressions | 0         |
| Depth drawing             | 1         |
| Dynamic Primitive drawing | 1         |
| Dynamic shadow setup      | 1         |
| FinishRenderTarget        | 1         |
| InitViews                 | 1         |
| Lighting drawing          | 1         |
| Proj Shadow drawing       | 2         |
| RenderVelocities          | 1         |
| StaticDrawList drawing    | 1         |
| RenderViewFamily          | 1         |
| Translucency drawing      | 1         |

| Counters                | Average  |
|-------------------------|----------|
| Dynamic path draw calls | 41.00    |
| Mesh draw calls         | 457.27   |
| Present time            | 10.51 ms |
| Lights in scene         |          |
| Static list draw calls  | 359.27   |



GPU Visualizer

FRAME\Scene

PrePass

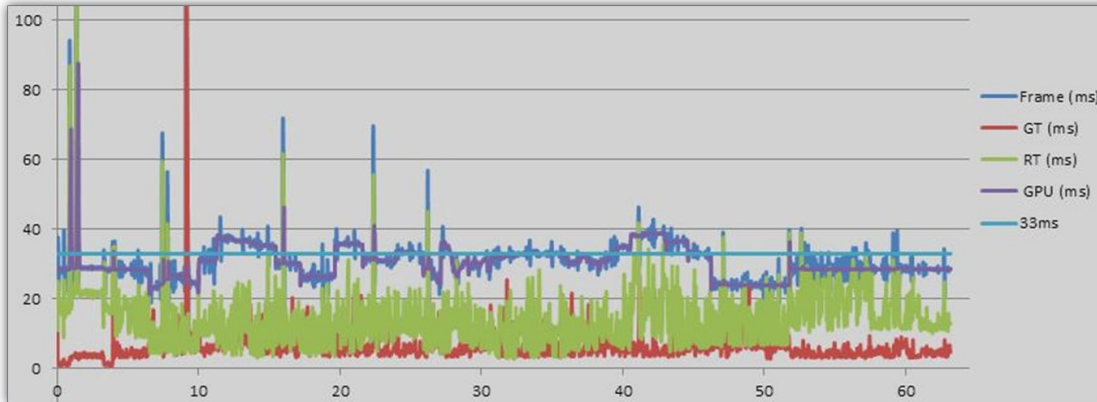
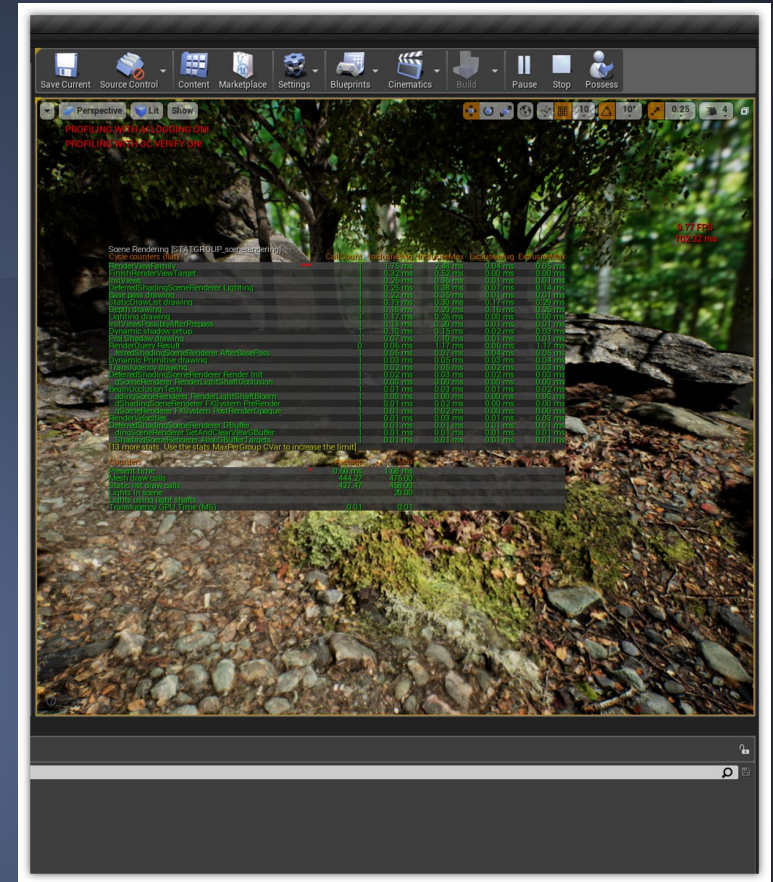
Zoom 1.00x

Search

| Name                     | Duration (ms) |
|--------------------------|---------------|
| InitViews                | 0.00          |
| PrePass                  | 1.75          |
| BeginRenderingPrePass    | 0.00          |
| PosOnlyOpaque            | 1.29          |
| Opaque                   | 0.18          |
| Opaque                   | 0.00          |
| Dynamic                  | 0.25          |
| FinishRenderingPrePass   | 0.00          |
| ClearRenderTargetes      | 0.04          |
| BeginRenderingSceneColor | 0.00          |
| BasePass                 | 1.65          |
| ResolveSceneColor        | 0.00          |

# Profiling

- Consola de comandos
  - Stat FPS
  - Stat Unit
  - Stat SceneRendering
  - Stat RHI



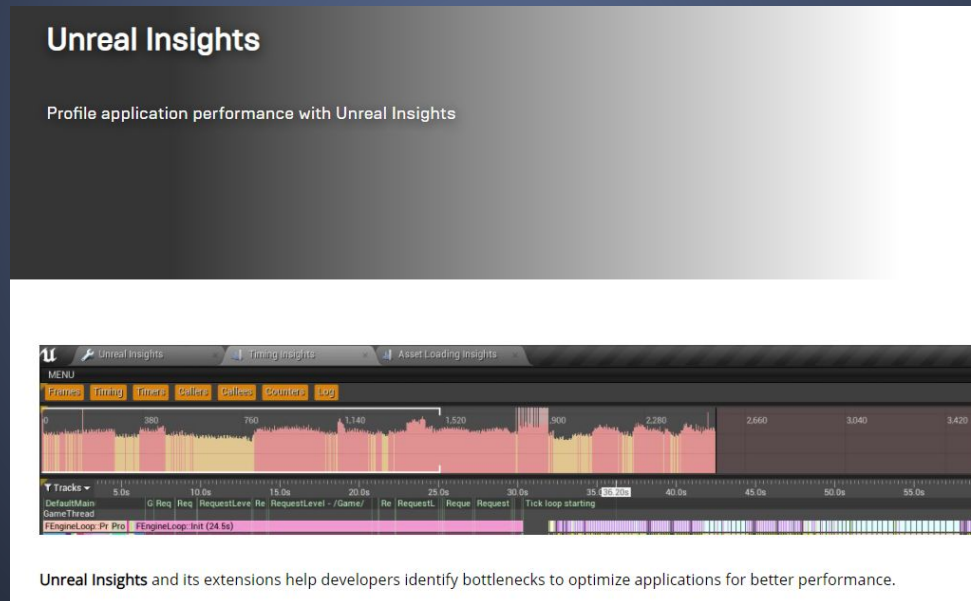
# Herramientas de profiling

|                       |                                                                                                                                                                                                                                                                                                                                                                        |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Console Command Bar   | This text field is used to input console commands within the editor.                                                                                                                                                                                                                                                                                                   |
| Source Control Status | This shows the current status of the integrated Perforce server.                                                                                                                                                                                                                                                                                                       |
| Performance Data      | <p>This region displays the following information:</p> <ul style="list-style-type: none"><li>▪ <b>FPS:</b> Shows the frames per second, as well as the current delta time (time between each frame).</li><li>▪ <b>Mem:</b> Shows the current amount of memory in use by the engine.</li><li>▪ <b>Objs:</b> Shows the current number of objects in the scene.</li></ul> |
| 20 Second Playback    | <p>This saves a 20-second video of the last 20 seconds of work. A bar will appear at the bottom of the Level Editor showing the path to the video.</p> <div> Video capture saved as <code>E:/UE4/ExampleGame/Saved/VideoCaptures/CaptureVideo_02.avi</code></div>                   |

<https://docs.unrealengine.com/en-US/TestingAndOptimization/PerformanceTools/index.html>

# Herramientas de profiling

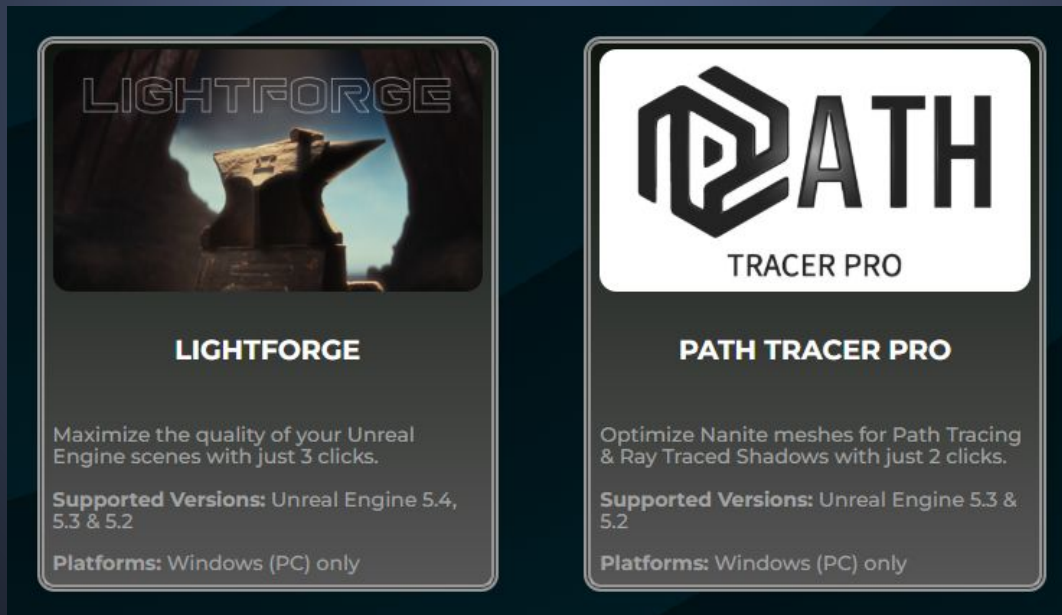
- **Unreal Insights** es un nuevo conjunto de herramientas para analizar *más a fondo*



<https://docs.unrealengine.com/en-US/TestingAndOptimization/PerformanceAndProfiling/UnrealInsights/index.html>

# Otras herramientas de optimización

- Existen plugins que te ayudan a configurar proyectos de Unreal Engine
  - Ej. **Boundless Entertainment**



# Referencias

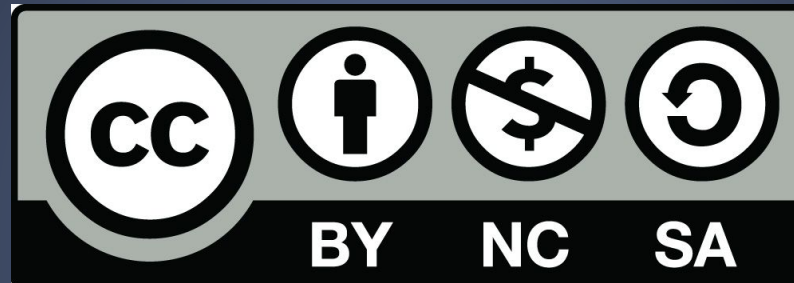
- Epic Games - Cursos

- <https://dev.epicgames.com/community/learning/talks-and-demos/opPx/unreal-engine-optimizacion-de-ue5-renderizacion-avanzada-rendimiento-grafico-y-gestion-de-la-memoria-unreal-fest-2024>
- <https://dev.epicgames.com/community/learning/talks-and-demos/PZz7/unreal-engine-optimizacion-de-ue5-replanteamiento-de-los-paradigmas-de-rendimiento-para-elementos-visuales-de-alta-calidad-parte-1-nanite-y-lumen-unreal-fest-2023>
- <https://dev.epicgames.com/community/learning/talks-and-demos/jZ5V/unreal-engine-optimizacion-de-ue5-replanteamiento-de-los-paradigmas-de-rendimiento-para-elementos-visuales-de-alta-calidad-parte-2-soporte-a-sistemas-unreal-fest-2023>

- Rabin, S.: DevDeck, 144 cartas con consejos  
<https://steverabindevdecks.com/>

- Tech Art Aid  
<https://techartaid.com/>

# Críticas, dudas, sugerencias...



*\* Licencia sólo aplicable al texto original de estas diapositivas*

Federico Peinado (2019-2026)

[www.federicopeinado.es](http://www.federicopeinado.es)