Análisis Técnico y Pipeline de Automatización: Capacidades de IA, Agentes y Generación Procedural en Unreal Engine 5.7

Sección 1: Un Marco Estratégico para la Refactorización de Proyectos en UE 5.7

1.1 Análisis de la Solicitud y Mapeo Tecnológico

La migración de un proyecto de Unreal Engine 5.4 a 5.7 introduce un conjunto de herramientas de automatización, procedurales y de IA que redefinen fundamentalmente los flujos de trabajo de producción. La solicitud de una refactorización completa de *assets* (enemigos, música, estética) utilizando "IA, agentes y aprendizaje por refuerzo" es ambiciosa y apunta a una confluencia de varias tecnologías distintas.

Un análisis técnico de las capacidades de UE 5.7 revela que la

automatización de la refactorización de *assets* no es una tarea monolítica que deba ser manejada por una sola "IA". Más bien, la solicitud del usuario debe ser deconstruida y mapeada a un conjunto de herramientas especializadas.

- **Intención vs. Herramienta:** La solicitud de "IA" y "Agentes" para la modificación de *assets* está conceptualmente dividida. Las herramientas de IA nativas (como el nuevo AI Assistant) y las herramientas de Aprendizaje por Refuerzo (RL) (como el *plugin* Learning Agents) no están diseñadas para la manipulación directa de *assets* en el editor.
- **Mapeo de Intención a Herramienta:**
  - La tarea de **reemplazo masivo y consolidación de *assets*** (refactorización 1:1) se resuelve de manera más robusta mediante el *scripting* del editor con la **API de Python**.[1]
  - La tarea de **redefinir la estética y la colocación de enemigos** (refactorización 1:N) es el dominio principal del *framework* **Procedural Content Generation (PCG)**, que ahora está listo para producción en 5.7.[3]
  - La solicitud de **"agentes de IA para operar el editor"** se alinea con *plugins* comunitarios experimentales (ej. UnrealMCP) que conectan Modelos de Lenguaje Grandes (LLMs) externos al editor.[4]
  - La solicitud de **"Aprendizaje por Refuerzo" (RL)** se correlaciona con el *plugin* **Learning Agents**, cuyo propósito es entrenar el *comportamiento* de los NPC, no crear sus *assets*.[5]
  - La **"nueva capacidad de IA"** nativa de UE 5.7 es el **AI Assistant**, una herramienta de consulta de documentación y generación de código que no tiene acceso al contexto del proyecto para modificar *assets*.[7]

Este informe proporcionará, por lo tanto, una guía técnica de doble propósito: primero, los flujos de trabajo prácticos y robustos (Python y PCG) para ejecutar la refactorización de *assets* solicitada; y segundo, una investigación profunda sobre las capacidades reales de los agentes de IA y RL, aclarando su propósito y proporcionando guías de configuración experimentales.

1.2 Tabla: Matriz de Tecnología de Automatización de UE 5.7

Para clarificar el conjunto de herramientas, la siguiente matriz mapea las tareas de refactorización deseadas con las tecnologías de UE 5.7 apropiadas.

Tecnología	**Refactorización Masiva de Assets (Reemplazo 1:1, Consolidación)**	**Reemplazo de Estética/Enemigos (Colocación Procedural)**	**Creación de Nueva Lógica de IA para Enemigos (Comportamiento)**	**Asistencia de Código y Búsqueda de Documentación**	**Prototipado Rápido con Lenguaje Natural**
Scriptin	**Excelen**	Posible,	Inadecu	Inadecu	Inadecu

g de Python (API de Editor)	**te.** Método principal para la manipulación de *assets* por lotes.[2]	pero engorroso.[9]	ado.	ado.	ado.
Framework PCG (Production-Ready)	Inadecuado.	**Excelente.** Método principal para la generación de entornos y la colocación de actores.[3]	Inadecuado (pero puede *colocar* la IA).	Inadecuado.	Inadecuado.
Plugin Learning Agents (RL)	Inadecuado.	Inadecuado.	**Excelente.** Propósito principal para	Inadecuado.	Inadecuado.

			entrenar comportamientos de IA.[5]		
AI Assistant (Nativo)	Inadecuado (No puede modificar *assets*).[7]	Inadecuado.[7]	Puede ayudar a *escribir* código de IA.[11]	**Excelente.** Propósito principal.[12]	Inadecuado.
Agentes Externos (ej. Unreal MCP)	Experimental/Peligroso.[4]	Experimental. Puede ejecutar comandos de PCG.[4]	Experimental.	Inadecuado.	**Excelente.** Propósito principal.[13]

Sección 2: Automatización de Tareas Pesadas: Manipulación Masiva de *Assets* mediante Python

La base de cualquier refactorización seria de un proyecto reside en el *scripting* del editor. La API de Python de Unreal Engine proporciona acceso directo al AssetRegistry y a las herramientas de manipulación de *assets*, permitiendo operaciones por lotes que

ahorrarían cientos de horas de trabajo manual.[1]

2.1 Fundamentos del Pipeline: API de Python para la Refactorización del Editor

Antes de ejecutar cualquier *script* de modificación masiva, es fundamental comprender la regla de oro de la automatización de *assets* en Unreal: nunca se deben utilizar módulos de Python estándar como os o shutil para mover, renombrar o eliminar archivos de *assets* (archivos .uasset) directamente en el disco.[2]

Unreal Engine no opera sobre un sistema de rutas de archivos, sino sobre un sistema de referencias de *assets* (Object Paths) gestionado por el AssetRegistry. Usar os.rename rompería todas las referencias a ese *asset* dentro de otros *Blueprints*, niveles y materiales, causando una corrupción generalizada del proyecto.

La única forma segura de manipular *assets* es a través de las APIs de Unreal, que gestionan correctamente las referencias y crean ObjectRedirectors para garantizar que otros *assets* puedan encontrar los archivos movidos.

APIs Clave:

- unreal.EditorAssetLibrary: Para cargar, guardar, renombrar, duplicar y encontrar *assets*.[2]
- unreal.AssetTools: Para operaciones más complejas como consolidar, importar y crear *assets*.[2]
- unreal.EditorLevelLibrary: Para manipular actores y componentes dentro del nivel abierto actualmente.[8]

Configuración Requerida:

1. Activar el *plugin* **Python Editor Scripting** (normalmente activado por defecto).
2. Activar el *plugin* **Editor Scripting Utilities** para acceder a funciones de conveniencia adicionales.[1]

2.2 Guías de Código Prácticas para el Reemplazo de *Assets*

A continuación, se presentan *scripts* de Python prácticos para abordar las tareas de refactorización específicas (enemigos, música, estética).

2.2.1 Reemplazo y Consolidación de Referencias (El "Bisturí")

Este flujo de trabajo se utiliza cuando existen múltiples copias duplicadas de un mismo *asset* (ej. varias versiones de SM_Enemy_Base) y el objetivo es consolidar todas las referencias en una sola versión "maestra". Esto es un paso crucial de limpieza después de una migración.

Si bien la herramienta manual **Replace References Tool** existe para esto [17], se puede automatizar usando unreal.AssetTools.consolidate_assets. Esta función toma un *asset* "maestro" y una lista de *assets* a reemplazar, y actualiza todas las referencias que apuntaban a los *assets* de la lista para que apunten

al maestro, eliminando de forma segura los *assets* reemplazados.[16]

Es importante destacar que la consolidación solo funciona entre *assets* de la misma clase (con una excepción para Materiales y Texturas).[16]

Script de Ejemplo (Python): Consolidar Mallas Duplicadas

Python

```
import unreal

# --- Configuración ---
# El asset que queremos mantener
asset_to_keep_path = "/Game/NewContent/Enemies/SM_Enemy_Master.SM_Enemy_Master"
# Lista de assets que queremos reemplazar y eliminar
assets_to_replace_paths =
# --------------------

asset_tools = unreal.AssetToolsHelpers.get_asset_tools()
eal = unreal.EditorAssetLibrary

# Cargar el asset maestro
asset_to_keep = eal.load_asset(asset_to_keep_path)

if not asset_to_keep:
        print(f"Error: No se pudo cargar el asset maestro: {asset_to_keep_path}")
```

```
else:
    # Cargar los assets a reemplazar
    assets_to_replace =
    for path in assets_to_replace_paths:
        asset = eal.load_asset(path)
        if asset:
            assets_to_replace.append(asset)
        else:
                    print(f"Advertencia: No se encontró el asset a reemplazar: {path}")

    if assets_to_replace:
        # Ejecutar la consolidación
        # Esto actualizará todas las referencias (en Blueprints, niveles, etc.)
        # que usaban assets_to_replace para que usen asset_to_keep.
                                asset_tools.consolidate_assets(asset_to_keep, assets_to_replace)
                print(f"Operación completada. {len(assets_to_replace)} assets consolidados en {asset_to_keep.get_name()}")
```

2.2.2 Reemplazo Masivo de Materiales (La Tarea "Estética")

Para reemplazar la "estética", existen dos enfoques: reemplazar materiales en actores *ya colocados* en un nivel, o reemplazar la asignación de material en el *asset* de malla base.

Método 1: Reemplazar Materiales en Actores del Nivel (No destructivo)

Este método utiliza unreal.EditorLevelLibrary.replace_mesh_components_materials_on_actors y es útil para cambiar la apariencia de un nivel sin modificar los assets base.8

Script de Ejemplo (Python): Reemplazar Materiales en un Nivel Abierto

Python

```
import unreal

# --- Configuración ---
old_material_path = "/Game/OldContent/Materials/M_Old_Metal.M_Old_Metal"
new_material_path = "/Game/NewContent/Substrate/M_Substrate_Metal_A.M_Substrate_Metal_A"
# --------------------

eal = unreal.EditorAssetLibrary
ell = unreal.EditorLevelLibrary

# Cargar los materiales
old_material = eal.load_asset(old_material_path)
new_material = eal.load_asset(new_material_path)

if not old_material or not new_material:
    print("Error: No se pudieron cargar los materiales. Verifique las rutas.")
```

```
else:
    # Obtener todos los StaticMeshActors en el nivel actual
    actor_list = ell.get_all_level_actors()
            sm_actors = unreal.EditorFilterLibrary.by_class(actor_list,
unreal.StaticMeshActor)

    if sm_actors:
        # Ejecutar el reemplazo
          ell.replace_mesh_components_materials_on_actors(sm_actors,
old_material, new_material)
        print(f"Materiales reemplazados en {len(sm_actors)} actores.")
    else:
        print("No se encontraron StaticMeshActors en el nivel.")
```

Método 2: Cambiar el Material en el Asset de Malla (Permanente)
Este método modifica permanentemente el UStaticMesh o USkeletalMesh, cambiando el material por defecto en cualquier lugar donde se utilice el asset. Esto es más probable lo que se necesita para una refactorización completa.19

Script de Ejemplo (Python): Cambiar Material en un *Asset* de Malla Estática

Python

```
import unreal

# --- Configuración ---
mesh_path                                                    =
```

```
"/Game/NewContent/Props/SM_Prop_Chest.SM_Prop_Chest"
material_slot_index = 0 # El índice del slot de material a cambiar
new_material_path                                                    =
"/Game/NewContent/Substrate/M_Substrate_Wood.M_Substrate_Wood"
# --------------------

eal = unreal.EditorAssetLibrary

# Cargar los assets
mesh = eal.load_asset(mesh_path)
new_material = eal.load_asset(new_material_path)

if mesh and new_material:
    if isinstance(mesh, unreal.StaticMesh):
        # Asignar el nuevo material al slot
        mesh.set_material(material_slot_index, new_material)
        # Marcar el asset como modificado y guardarlo
        eal.save_dirty_asset(mesh)
        print(f"Material actualizado en el asset: {mesh.get_name()}")
    else:
        print(f"Error: El asset no es un StaticMesh: {mesh_path}")
else:
    print("Error: No se pudieron cargar la malla o el material.")
```

2.2.3 Refactorización de *Assets* de Enemigos (Modificación de *Blueprints*)

Este es el desafío central. Un "enemigo" es típicamente un Blueprint

que contiene un SkeletalMeshComponent (y otros componentes). Para reemplazar la malla del enemigo, se debe modificar el *asset* Blueprint en sí mismo.

Esto es complejo porque los componentes de un Blueprint no se pueden modificar directamente. Se debe acceder a la plantilla del *Blueprint*, conocida como el **Class Default Object (CDO)**, o, de forma más robusta, usar el SubobjectDataSubsystem.[21]

El siguiente *script* utiliza el método del CDO, que funciona bien para componentes añadidos nativamente (como el componente Mesh en un Character Blueprint).[23]

Script de Ejemplo (Python): Cambiar la Malla de un Enemigo *Blueprint*

Python

```
import unreal

# --- Configuración ---
# El Blueprint del enemigo que queremos modificar
enemy_bp_path = "/Game/Enemies/BP_Enemy_Grunt.BP_Enemy_Grunt"
# La nueva malla esquelética que queremos asignar
new_mesh_path = "/Game/NewContent/Enemies/SK_New_Grunt.SK_New_Grunt"
# El nombre del componente de malla dentro del Blueprint (usualmente 'Mesh')
component_name = "Mesh"
# --------------------
```

```
eal = unreal.EditorAssetLibrary

# Cargar la nueva malla
new_mesh = eal.load_asset(new_mesh_path)
if not new_mesh:
    print(f"Error: No se pudo cargar la nueva malla: {new_mesh_path}")
else:
    # Cargar la CLASE del Blueprint y obtener su Class Default Object (CDO)
    bp_class = eal.load_blueprint_class(enemy_bp_path)
    if not bp_class:
        print(f"Error: No se pudo cargar la clase del Blueprint: {enemy_bp_path}")
    else:
        bp_cdo = unreal.get_default_object(bp_class)

        # Encontrar el componente de malla por su nombre
        mesh_component = bp_cdo.get_editor_property(component_name)

        if mesh_component and isinstance(mesh_component, unreal.SkeletalMeshComponent):
            # Asignar la nueva malla
            mesh_component.set_editor_property("skeletal_mesh", new_mesh)

            # Marcar el Blueprint original como modificado y guardarlo
            blueprint_asset = eal.load_asset(enemy_bp_path)
            eal.save_dirty_asset(blueprint_asset)
```

```
            print(f"Malla reemplazada exitosamente en {enemy_bp_path}")
        else:
            print(f"Error: No se encontró el SkeletalMeshComponent llamado '{component_name}' en el CDO de {enemy_bp_path}.")
```

2.2.4 Refactorización de *Assets* de Audio (La Tarea "Música")

La refactorización de la "música" y el audio a menudo implica corregir referencias rotas dentro de SoundCues.[24] Los SoundCues son gráficos que reproducen SoundWaves (los archivos de audio reales).[25] Si los SoundWaves se movieron durante la migración, los SoundCues quedarán en silencio.

Un *script* de refactorización de audio debe iterar sobre todos los *assets* SoundCue, encontrar todos los nodos SoundWavePlayer internos, verificar si su SoundWave asignado es nulo y, de ser así, buscar un SoundWave de reemplazo por nombre o en una nueva carpeta.

2.3 Herramientas de Producción: Editor Utility Widgets (EUW)

Para que estos *scripts* sean utilizables por un equipo, no deben ejecutarse desde la consola. Los **Editor Utility Widgets (EUW)** son la solución. Son UIs creadas con UMG que se ejecutan dentro del

editor.[26]

Se puede crear un EUW que proporcione campos de texto para "Carpeta de Materiales Antiguos" y "Carpeta de Materiales Nuevos", y un botón. Al hacer clic en el botón, el EUW puede ejecutar lógica de *Blueprint* que a su vez llama al nodo Execute Python Script [28], pasando las rutas de las carpetas como entradas al *script* de Python. Esto abstrae la complejidad del código y proporciona una herramienta de refactorización robusta y reutilizable para el equipo.

Sección 3: Refactorización Procedural: Uso de PCG para el Reemplazo Dinámico de *Assets* y Enemigos

Si bien los *scripts* de Python de la Sección 2 son excelentes para el reemplazo 1:1, la verdadera potencia de UE 5.7 para refactorizar "estética" y "enemigos" proviene del *framework* **Procedural Content Generation (PCG)**.

3.1 PCG como Herramienta de Refactorización de Niveles

En UE 5.7, PCG ha sido promovido a **Production-Ready**.[3] Esto es un cambio significativo desde UE 5.4 y da la confianza para usarlo

como un pilar central del diseño de niveles.

En lugar de simplemente reemplazar la *malla* de un enemigo (BP_Enemy_Grunt_V1 por BP_Enemy_Grunt_V2) usando Python, un enfoque de PCG implica eliminar por completo a los enemigos colocados manualmente en el nivel. En su lugar, se utilizan volúmenes de PCG para *colocar dinámicamente* a los enemigos basándose en un conjunto de reglas (ej. "colocar enemigos patrullando cerca de esta ruta", "colocar francotiradores en puntos altos", "no colocar enemigos cerca del jugador").

Este enfoque no solo refactoriza los *assets* existentes, sino que refactoriza la *metodología de diseño de niveles* hacia un sistema más dinámico, iterativo y no destructivo.

3.2 Flujo de Trabajo: Reemplazo de Enemigos con PCG y Blueprints

Un error común es usar PCG para intentar *spawnear* directamente una malla de enemigo. Un flujo de trabajo mucho más potente es usar PCG para *colocar* un Blueprint que contenga la lógica de *spawning*.[31]

Este enfoque combina el poder de PCG (para la colocación espacial) con el poder de *Blueprint* (para la lógica de *gameplay*).

Paso 1: Crear el Blueprint de Spawner (BP_EnemySpawner)
En lugar de que PCG decida qué tipo específico de enemigo spawnear, se crea un único Actor Blueprint llamado BP_EnemySpawner.

1. Crear un nuevo Actor Blueprint.
2. Añadir una variable: EnemyClasses (Tipo: Actor Class, Marcar como Array y Instance Editable).[34]
3. Poblar este *array* con las clases de *Blueprint* de los enemigos que este *spawner* puede crear (ej. BP_Enemy_Grunt, BP_Enemy_Brute).
4. En el Event BeginPlay (o en una función OnSpawn llamada desde el Construction Script [37]):
    - Obtener el *array* EnemyClasses.
    - Usar un nodo Get (a copy) con un Random Integer in Range (de 0 al Length del *array* - 1) para seleccionar una clase de enemigo al azar.[35]
    - Usar el nodo Spawn AI From Class (o Spawn Actor) para instanciar esa clase de enemigo seleccionada en la ubicación (Get Actor Transform) del *spawner*.
    - Opcionalmente, llamar a Destroy Actor sobre sí mismo (self) después de *spawnear* al enemigo.

Paso 2: Configurar el Grafo de PCG

1. En el nivel, añadir un PCG Volume. Crear un nuevo PCG Graph.
2. **Obtener Puntos:** Añadir un nodo Get Volume Data (o Get Spline Data para patrullas [39]) y conectarlo a un Surface Sampler.[10]
3. **Filtrar Puntos:** Usar nodos como Density Noise, Slope (para evitar pendientes pronunciadas) o Get Actor Data (para filtrar por distancia a actores etiquetados, como el PlayerStart [40]) para definir reglas de *gameplay* para la colocación.
4. **Instanciar el Spawner:** Conectar la salida filtrada a un nodo **Spawn Actor**.[31]
5. En el nodo Spawn Actor, en el panel de Details:
    - Establecer Template Actor Class en BP_EnemySpawner.

- (Opcional) Usar Spawn Actor Property Overwrite para pasar atributos desde PCG (ej. un float llamado SpawnDifficulty) a una variable pública correspondiente en el BP_EnemySpawner.[31] Esto permite que PCG controle la lógica del *Blueprint* que ejecuta.

Ahora, la colocación de enemigos está totalmente controlada por el grafo de PCG. "Reemplazar" enemigos es tan simple como modificar el *array* EnemyClasses en el BP_EnemySpawner o ajustar los parámetros en el grafo de PCG.

3.3 La Nueva Frontera: Nodo Execute Python Script en PCG (UE 5.7)

UE 5.7 introduce un nuevo nodo de PCG: **Execute Python Script**.[30] Esta es una característica de nivel experto que une los dos mundos de este informe: la automatización de *assets* (Python) y la generación de niveles (PCG).

Este nodo permite que un grafo de PCG ejecute un *script* de Python en el editor *durante* la generación. Los puntos de PCG pueden pasar sus datos (posiciones, atributos, etc.) al *script*. El *script* puede entonces realizar operaciones complejas, como crear un nuevo *asset* en el Content Browser (usando el código de la Sección 2), y luego devolver la ruta de ese nuevo *asset* como un atributo de salida en los puntos de PCG.

Caso de Uso de Refactorización Avanzada:

1. Un grafo de PCG utiliza el nuevo nodo Scene Capture (también

nuevo en 5.7 [30]) para muestrear el color de la iluminación en un punto de *spawn* de un enemigo.
2. Pasa este valor de color al nodo Execute Python Script.
3. El *script* de Python recibe el color, crea un nuevo MaterialInstanceConstant en /Game/Generated/Materials/, y establece su parámetro BodyColor en el color muestreado.
4. El *script* devuelve la ruta del nuevo material (ej. /Game/Generated/Materials/MI_Enemy_Cam_01.MI_Enemy_Cam_01) como un atributo DynamicMaterial.
5. El nodo Spawn Actor (que instancia el BP_EnemySpawner) utiliza Spawn Actor Property Overwrite para pasar este nuevo atributo DynamicMaterial al *Blueprint*.
6. El BP_EnemySpawner recibe la ruta, la carga y la aplica al enemigo que *spawnea*.

Este flujo de trabajo permite la creación de *assets* generativos persistentes, uniendo completamente la generación procedural con la automatización de *assets*.

3.4 Tabla: Nodos PCG Clave para la Generación de *Gameplay*

Para una refactorización de *gameplay*, el enfoque debe estar en los nodos de PCG que van más allá de la dispersión de vegetación.

Nodo	Función y Caso de Uso de	Fuente(s)

	Refactorización	
Spawn Actor	**El nodo más importante.** Instancia *Blueprints* (ej. BP_EnemySpawner, BP_LootCrate) en cada punto de PCG, en lugar de mallas estáticas.	31
Get Actor Data	"Lee" el mundo. Permite a PCG obtener la ubicación de actores etiquetados (ej. 'PlayerStart', 'ExitPoint') y usar esos datos para filtrar puntos (ej. "no *spawnear* enemigos a 10m del PlayerStart").	40
Execute Python Script	(Nuevo en 5.7) Ejecuta *scripts* de automatización del editor *dentro* del grafo. Permite la	30

	generación de *assets* (Materiales, etc.) sobre la marcha.	
Get Spline Data	Genera puntos a lo largo de un *spline*. Fundamental para reemplazar patrullas de enemigos, colocar vallas o crear muros procedurales.	[39]
Scene Capture	(Nuevo en 5.7) Captura el color o la profundidad de la escena. Permite que PCG reaccione a la iluminación (ej. "colocar enemigos de sigilo solo en áreas oscuras") o a otros *assets*.	[30]

Sección 4: La Frontera de la IA: Configuración de Agentes para el Control

del Editor

Esta sección aborda directamente la solicitud de "configurar agentes de IA para que operen y redefinan *assets*". Es crucial diferenciar entre el sistema nativo de UE 5.7 y los *frameworks* de agentes externos experimentales.

4.1 Parte 1: El "Guía" - El Asistente de IA Nativo de UE 5.7

Unreal Engine 5.7 introduce un **AI Assistant** integrado.[3] Este es un LLM entrenado en la documentación de Unreal Engine y en ejemplos de código.

Capacidades:

- **Asistencia de Documentación:** Puede responder preguntas sobre conceptos del motor (ej. "¿Cómo funciona MegaLights?", "¿Qué es Substrate?").[12]
- **Generación de Código:** Puede generar fragmentos de código C++ y *Blueprint*.[11] Por ejemplo, se le puede pedir que "escriba un *script* de Python para renombrar *assets* en una carpeta", y proporcionará un punto de partida para los *scripts* de la Sección 2.[30]

Limitaciones Críticas (La "Pared"):
El AI Assistant nativo NO es un agente operativo.

- **Sin Acceso al Proyecto:** No tiene conocimiento de su proyecto.

No puede leer sus archivos, examinar sus *assets* en el Content Browser ni modificar actores en su nivel.[7]

- **Alucinaciones:** Como todos los LLMs, puede "alucinar" o inventar APIs o nodos que no existen.[12]

Veredicto para la Refactorización: El AI Assistant es un *asistente* de programación, no un *agente* de automatización. Puede ayudar a *escribir* los *scripts* de Python de la Sección 2, pero no puede *ejecutarlos* ni *operar* en el proyecto.

4.2 Parte 2: El "Operador" - Agentes de IA Externos y MCP (Machine Control Protocol)

Lo que se alinea más estrechamente con la solicitud de "configurar agentes de IA para que operen" es un campo experimental que utiliza el **Modelo de Protocolo de Contexto (MCP)**. MCP es un estándar emergente para permitir que los LLMs interactúen con aplicaciones de escritorio.

El *plugin* de código abierto **UnrealMCP** es una implementación de este concepto.[4] Conecta una IA externa (como **Claude for Desktop**) al editor de Unreal a través de un servidor TCP, permitiéndole ejecutar comandos, manipular actores y ejecutar código Python en respuesta a instrucciones en lenguaje natural.[4]

Guía de Configuración Técnica (Experimental):
La configuración de UnrealMCP (basada en el README del proyecto y tutoriales de la comunidad) es un proceso técnico 4:

1. **Requisitos:**

- Unreal Engine 5.5+ (UE 5.7 es compatible).
- Un proyecto C++ (se puede añadir una clase C++ vacía a un proyecto de *Blueprint* para convertirlo).
- Python 3.7+ instalado en el sistema.
- Un cliente de IA compatible con MCP, como **Claude for Desktop**.[13]

2. **Instalación del Plugin:**
   - Clonar el repositorio de UnrealMCP: git clone https://github.com/kvick-games/UnrealMCP.git Plugins/UnrealMCP en el directorio raíz del proyecto.[4]
   - Habilitar los *plugins* **UnrealMCP** y **Python Editor Scripting** en el editor (Editar > Plugins).[4]
3. **Construcción del Proyecto:**
   - Hacer clic derecho en el archivo .uproject y seleccionar "Generate Visual Studio project files".[4]
   - Abrir la solución .sln en Visual Studio y construir (Build) el proyecto.[4]
4. **Configuración del Cliente (Claude for Desktop):**
   - Instalar Claude for Desktop.
   - Localizar el archivo claude_desktop_config.json. (En Windows, típicamente en C:\Users\\AppData\Roaming\Claude).[4]
   - Abrir el archivo JSON y añadir la entrada mcpServers, asegurándose de que la ruta en "command" apunta al archivo .bat dentro del *plugin* en *su* proyecto:

```
JSON
{
  "mcpServers": {
    "unreal": {
                                                "command": "C:\\Ruta\\Completa\\A\\Tu\\Proyecto\\Plugins\\UnrealMCP\\MCP\\run_unreal_mcp.bat",
```

```
        "args":
      }
    }
}
```

- Reiniciar Claude for Desktop para cargar la nueva configuración.[48]

5. **Ejecución:**
   - Abrir el proyecto de Unreal Engine.
   - Aparecerá un nuevo botón de MCP en la barra de herramientas. Hacer clic en él para iniciar el servidor TCP.[4]
   - Abrir Claude. Ahora se le pueden dar instrucciones como: "Crea un cubo en (0, 0, 100)" o "Ejecuta el script de Python 'C:/Temp/refactor.py'".[13]

Ejemplos de Comandos (Cliente Python):
El plugin también expone un cliente de Python para control programático 4:

Python

```
from unreal_mcp_client import UnrealMCPClient
client = UnrealMCPClient("localhost", 13377)

# Crear un actor
client.create_object(
    class_name="StaticMeshActor",
    asset_path="/Engine/BasicShapes/Cube.Cube",
    location=(0, 0, 100),
    name="MCP_Cube"
```

```
)

# Ejecutar Python dentro de Unreal
client.execute_python("import unreal; unreal.EditorAssetLibrary.save_current_level()")
```

Análisis de Viabilidad y Riesgos:

- **¡ADVERTENCIA!:** Este es un *software* altamente experimental. El propio autor del *plugin* advierte: "Utilice este *plugin* bajo su propio riesgo" y "Los archivos podrían ser eliminados o modificados accidentalmente".[4]
- **Veredicto:** Esta tecnología es **inadecuada para la refactorización de producción** debido a su naturaleza experimental y al riesgo de corrupción de datos. Es, sin embargo, una herramienta de **prototipado rápido** extremadamente potente. No debe usarse para ejecutar la refactorización principal (para eso está la Sección 2), sino para experimentar creativamente *después* de que la refactorización esté completa.

4.3 Alternativas (Ludus AI)

Existen otros *plugins* de IA de terceros, como Ludus AI.[51] A diferencia de MCP (un *controlador* del editor), Ludus se centra en la *generación* de *assets* (ej. texto a modelo 3D) y asistencia de *Blueprint*.[52] Sin embargo, el estado actual de la generación de mallas 3D de alta calidad listas para producción a partir de texto sigue siendo un área de intensa investigación, y los resultados pueden

variar.[53]

Sección 5: Reevaluación del Aprendizaje por Refuerzo: El *Plugin* 'Learning Agents'

La solicitud de usar "aprendizaje por refuerzo" (RL) para la refactorización de *assets* apunta a una confusión común entre la *generación* de contenido y el *entrenamiento* de comportamiento.

5.1 Clarificación del Propósito: Comportamiento vs. Creación

La investigación académica ha explorado el concepto de **PCGRL (Generación de Contenido Procedural vía Aprendizaje por Refuerzo)**, donde un agente de RL aprende a *diseñar* niveles.[54]

Sin embargo, el *plugin* oficial de Epic Games, **Learning Agents**, no está diseñado para este propósito.[57] El *plugin* Learning Agents es un *framework* para entrenar el **comportamiento de los NPC** (o personajes) dentro del juego utilizando RL e Imitation Learning (IL).[5]

Los ejemplos oficiales de Epic se centran en tareas de *comportamiento*:

- Entrenar a un agente para conducir un coche alrededor de una pista.[6]
- Crear animaciones basadas en físicas (ej. un personaje

aprendiendo a mantener el equilibrio).
- Reemplazar Behavior Trees y State Machines tradicionales por una red neuronal entrenada.[5]

5.2 Aplicación para el Usuario: Entrenamiento de Nuevos Enemigos

Aunque Learning Agents no puede *crear* los *assets* de los enemigos, es la herramienta perfecta para crear la *nueva lógica de IA* para los enemigos que se están reemplazando.

En lugar de simplemente reemplazar la malla del enemigo y reutilizar el antiguo Behavior Tree de UE 5.4, se puede usar Learning Agents para entrenar un cerebro de IA completamente nuevo para esos enemigos. Esto es ideal para:

- Enemigos que aprenden a navegar por los entornos complejos generados por PCG.[61]
- IA de escuadrón que aprende tácticas de flanqueo.
- Enemigos con comportamientos de evasión y respuesta altamente dinámicos que serían difíciles de programar manualmente.

5.3 El Pipeline de IA/ML Completo de UE 5.7

Al sintetizar las secciones de este informe, emerge un *pipeline* de desarrollo de IA de próxima generación:

1. **Fase 1 (Assets):** Se utiliza **Python (Sección 2)** para limpiar, consolidar y refactorizar programáticamente los *assets* de los enemigos (mallas, materiales, *Blueprints*).[8]
2. **Fase 2 (Colocación):** Se utiliza **PCG (Sección 3)** para generar proceduralmente el mundo y *colocar* los BP_Spawners de enemigos basándose en reglas de diseño de niveles.[3]
3. **Fase 3 (Comportamiento):** Se utiliza el **Plugin Learning Agents (Sección 5)** para entrenar los sistemas de IA de esos enemigos, enseñándoles a navegar y luchar en los entornos generados por PCG.[5]
4. **Fase 4 (Pruebas/Iteración):** Se utilizan **Agentes Externos (MCP) (Sección 4)** para dar comandos en lenguaje natural como "Spawnear 50 enemigos entrenados y hacer que ataquen al jugador", permitiendo un ciclo de pruebas de IA rápido.[4]

Sección 6: Activación de las Actualizaciones Visuales y de Rendimiento de UE 5.7

La migración de un proyecto de 5.4 a 5.7 no activa automáticamente las nuevas características de renderizado. Deben ser habilitadas y los *assets* existentes (especialmente los materiales) deben ser convertidos manualmente.

6.1 Contexto de Migración: De 5.4 a 5.7

Un proyecto migrado de 5.4 mantendrá su *pipeline* de renderizado legado por defecto.[62] Para habilitar las nuevas características de iluminación y materiales, se requieren pasos explícitos. Se recomienda encarecidamente **hacer una copia de seguridad completa del proyecto** antes de proceder con la conversión de Substrate, ya que es destructiva.

6.2 Guía de Implementación: MegaLights (Beta)

- **Qué es:** Un nuevo *path* de iluminación directa que permite renderizar "órdenes de magnitud" más de luces dinámicas con sombras (especialmente luces de área) de lo que era posible anteriormente.[3]
- **Estado en 5.7:** Promovido a **Beta**. El soporte se ha expandido para incluir Luces Direccionales, partículas de Niagara, translucidez y *Hair Strands*.[3]
- **Cómo Habilitarlo (Guía de Configuración):**
  1. Ir a **Project Settings > Rendering > Direct Lighting**.
  2. Activar la casilla **Enable Mega lights**.[64]
  3. El editor solicitará activar **Support Hardware Ray Tracing**. Aceptar, ya que es recomendado.[64]
  4. Reiniciar el editor.
- **Uso y Optimización:**
  - Todas las luces locales ahora usarán el sistema MegaLights por defecto.[66]
  - **Control por Luz:** Se puede desactivar MegaLights en luces individuales desmarcando Allow MegaLights en el componente de luz.[64]

- **Control de Sombras:** Se puede elegir el método de sombra por luz (MegaLights Shadow Method) entre Ray Tracing (por defecto, mejor para sombras suaves) o Virtual Shadow Maps (VSMs) (mejor para detalles de Nanite).[64]
- **Mejores Prácticas:** Para minimizar el ruido, se debe evitar colocar luces dentro de la geometría, optimizar el Attenuation Radius de las luces y fusionar grupos de luces pequeñas en una sola luz de área grande.[64]

6.3 Guía de Migración: Substrate Materials (Production-Ready)

- **Qué es:** Un *framework* de materiales modular y físicamente correcto que reemplaza los modelos de sombreado fijos (Unlit, Default Lit, Clear Coat, etc.).[3] Permite la creación de materiales complejos y multicapa, como pintura de coche con barniz, metal rayado, o piel con sudor, con verdadera precisión física.[3]
- **Estado en 5.7:** Promovido a **Production-Ready**. Está activado por defecto en proyectos *nuevos*, pero **desactivado** en proyectos migrados.[3]
- **Cómo Migrar un Proyecto Existente (Guía de Conversión):**
  1. **¡HACER COPIA DE SEGURIDAD!** Este proceso es irreversible.
  2. Ir a **Project Settings > Rendering**.
  3. Activar la casilla **Enable Substrate materials**.[67]
  4. Reiniciar el editor.[68]
  5. Abrir un material "legado" existente. El material *no* se convierte automáticamente.[67]
  6. En el Material Editor, hacer clic derecho en el nodo de

resultado Material y seleccionar **Convert to Substrate**.[67]

7. Esto creará automáticamente un nodo Substrate Slab y conectará las entradas antiguas (Base Color, Roughness, etc.) a él.[67]

- **¡ADVERTENCIAS CRÍTICAS DE MIGRACIÓN!**
  - La conversión es **PERMANENTE E IRREVERSIBLE**. Un material guardado como Substrate no puede volver a ser un material legado.[67]
  - Si Substrate se desactiva después de la conversión, todos los materiales de Substrate se renderizarán como **NEGRO**.[67]
  - No se pueden mezclar materiales legados y de Substrate en el mismo proyecto.[68] Se debe convertir todo o nada.
  - Se recomienda encarecidamente probar la conversión en una copia del proyecto primero.

6.4 Guía de Implementación: Nanite Foliage (Experimental)

- **Qué es:** Un nuevo sistema de renderizado de geometría diseñado específicamente para vegetación densa.[3] Utiliza dos tecnologías:
  1. **Nanite Voxels:** Renderiza la vegetación distante (ej. copas de árboles) como vóxeles, eliminando el *pop-in* de LOD y el *overdraw* de las tarjetas de alfa.[3]
  2. **Nanite Assemblies:** Un método para instanciar partes de mallas (ej. ramas, hojas) que reduce drásticamente el uso de memoria y disco.[30]
- **Estado en 5.7: Experimental**.[3]

- **Cómo Habilitarlo y Usarlo (Flujo de Trabajo del PVE):**
  1. Ir a **Edit > Plugins** y activar el *plugin* **Procedural Vegetation Editor (PVE)**.[70]
  2. Ir a **Project Settings** y activar **Nanite Foliage**.[70]
  3. Reiniciar el editor.
  4. Hacer clic derecho en el Content Browser > **Foliage > Procedural Vegetation**.[71]
  5. Se abre el **Procedural Vegetation Editor**. Este es un editor de grafos (similar a PCG) que permite construir un árbol desde cero o modificar presets.[71]
  6. **Nodos Clave del PVE:** Carve (para podar), Gravity (para simular la caída de ramas), Mesh Builder (para generar la geometría del tronco), Foliage Distributor (para colocar las hojas).[70]
  7. **Exportación:** En el nodo de salida, configurar la ruta de exportación. Es crucial:
     - Asegurarse de que **"Nanite Foliage" está activado**.[70]
     - Seleccionar **"voxelization"** para el máximo rendimiento a distancia.[70]
  8. El resultado es un SkeletalMesh que puede ser animado con el nuevo sistema Dynamic Wind.[30]

6.5 Tabla: Lista de Verificación de Migración de Renderizado de UE 5.7

Característic	Estado (5.7)	Acción de	Riesgo de

a		Habilitación	Migración
MegaLights	**Beta** [3]	Project Settings > Rendering > Enable Mega lights [64]	**Bajo.** Reversible. Se puede activar/desactivar por luz.[64]
Substrate Materials	**Production-Ready** [3]	Project Settings > Rendering > Enable Substrate materials Y Right-click > Convert to Substrate en cada material.	**CRÍTICO.** La conversión de materiales es **permanente e irreversible**.[67] Requiere una copia de seguridad completa del proyecto.
Nanite Foliage	**Experimental** [3]	Habilitar *plugin* PVE y Nanite Foliage en Project Settings.[70]	**Medio.** Requiere la regeneración de todos los *assets* de vegetación a través del nuevo flujo de trabajo del PVE.[71]

Sección 7: Recomendaciones y Pipeline Estratégico Final

La migración a UE 5.7 no es una simple actualización de versión; es una oportunidad para una revisión completa del *pipeline*. El éxito de la refactorización no provendrá de una sola "herramienta de IA", sino de la orquestación inteligente de cuatro sistemas distintos: **Automatización (Python)**, **Generación (PCG)**, **Comportamiento (Learning Agents)** y **Asistencia (AI Assistant / MCP)**.

7.1 Síntesis y Flujo de Trabajo de Refactorización Recomendado

Se recomienda el siguiente plan de acción estratégico, dividido en fases:

Fase 1: Respaldo y Limpieza (Automatización con Python)

1. **RESPALDO COMPLETO:** Crear un respaldo completo del proyecto 5.7 migrado.
2. **LIMPIEZA DE BAJO RIESGO:** Ejecutar los *scripts* de Python de la **Sección 2.2.1** para consolidar *assets* duplicados. Esto reducirá el desorden antes de la refactorización principal.
3. **AUDITORÍA DE REFERENCIAS:** Ejecutar *scripts* para auditar referencias rotas (especialmente SoundCues [24]) e identificar el alcance del trabajo.

Fase 2: Conversión del Renderizado (La "Puerta de un Solo Sentido")

1. **HABILITAR SUBSTRATE:** Habilitar Substrate en la configuración del proyecto.[67] Identificar los Materiales Maestros. Convertir *un* material maestro, probarlo exhaustivamente en todos los escenarios de iluminación. Una vez validado, usar un *script* de Python para automatizar la conversión (Convert to Substrate) de los materiales restantes.
2. **HABILITAR MEGALIGHTS:** Habilitar MegaLights.[64] Auditar la iluminación del nivel. Es probable que se puedan reemplazar clústeres de luces falsas por luces de área únicas con sombras, mejorando la fidelidad visual y simplificando la escena.

Fase 3: Refactorización Procedural (PCG)

1. **ELIMINAR ASSETS MANUALES:** En un nivel de prueba, eliminar sistemáticamente la colocación manual de enemigos y *assets* de entorno (rocas, escombros, etc.).
2. **CREAR SPAMNERS:** Crear los BP_Spawners (como el BP_EnemySpawner de la **Sección 3.2**) que contienen la lógica de qué *spawnear*.[35]
3. **REPOBLAR CON PCG:** Crear Volúmenes de PCG para repoblar el nivel usando el nodo Spawn Actor para instanciar los BP_Spawners. Usar las reglas de PCG (filtros de *slope*, ruido, distancia) para controlar el diseño del *gameplay*.[33]
4. **(Opcional) REEMPLAZAR VEGETACIÓN:** Si el rendimiento del follaje es un problema, usar el *plugin* PVE (Sección 6.4) para convertir los *assets* de vegetación existentes al nuevo *pipeline* de Nanite Foliage.[71]

Fase 4: Desarrollo de Comportamiento (Learning Agents)

1. **REEMPLAZAR IA:** Con los enemigos ahora colocados por PCG, usar el *plugin* **Learning Agents (Sección 5)** para construir nuevos "cerebros" de IA. Entrenar a los agentes para que naveguen por los entornos generados por PCG y exhiban comportamientos de combate avanzados.[5]

Fase 5: Prototipado e Iteración (MCP Experimental)

1. **SOLO PARA I+D:** Para tareas de prototipado rápido *no* relacionadas con la refactorización principal, configurar **UnrealMCP (Sección 4.2)**. Usarlo para probar ideas rápidamente, como "colocar 50 agentes entrenados cerca del jugador y hacer que ataquen".[4]

7.2 Conclusión Final

Unreal Engine 5.7 proporciona todas las herramientas necesarias para lograr la refactorización completa solicitada. La clave del éxito es aplicar la herramienta correcta a la tarea correcta: usar **Python** para la manipulación quirúrgica de *assets* por lotes, usar **PCG** para la generación procedural de niveles y colocación de actores, y usar **Learning Agents** para el entrenamiento de comportamientos de IA. Las herramientas de "Agente de IA" (AI Assistant y MCP) actúan como asistentes de productividad y herramientas de prototipado rápido, en lugar de ser las soluciones centrales para la refactorización. La implementación de las nuevas características de renderizado (Substrate, MegaLights) modernizará la base visual del proyecto, pero requiere una migración cuidadosa y, en el caso de Substrate, irreversible.

Fuentes citadas

1. Scripting and Automating the Unreal Editor | Unreal Engine 5.7 Documentation, acceso: noviembre 16, 2025, https://dev.epicgames.com/documentation/en-us/unreal-engine/scripting-and-automating-the-unreal-editor
2. Scripting the Unreal Editor Using Python | Unreal Engine 5.7 Documentation, acceso: noviembre 16, 2025, https://dev.epicgames.com/documentation/en-us/unreal-engine/scripting-the-unreal-editor-using-python
3. Unreal Engine 5.7 is now available, acceso: noviembre 16, 2025, https://www.unrealengine.com/en-US/news/unreal-engine-5-7-is-now-available
4. kvick-games/UnrealMCP: MCP to allow AI agents to control Unreal - GitHub, acceso: noviembre 16, 2025, https://github.com/kvick-games/UnrealMCP
5. Learning Agents | Unreal Fest 2024 - YouTube, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=FYgJsN_fMr8
6. Learning to Drive (5.3) - Learning Agents (5.3) - Epic Games Developers, acceso: noviembre 16, 2025, https://dev.epicgames.com/community/learning/courses/M3D/unreal-engine-learning-agents-getting-started/qj2O/unreal-engine-learning-to-drive
7. Unreal Engine 5.7 NEW AI Assistant Plugin Tutorial - YouTube, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=aUqrqXygmU8
8. Replace Material Python Script - Unreal Engine Forums, acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/replace-material-python-script/272903
9. Python: Convert actors to static mesh - Editor Scripting -

Unreal Engine Forums, acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/python-convert-actors-to-static-mesh/152073
10. Procedural Content Generation Overview | Unreal Engine 5.7 Documentation, acceso: noviembre 16, 2025, https://dev.epicgames.com/documentation/en-us/unreal-engine/procedural-content-generation-overview
11. nobody's going to talk about the OFFICIAL UE AI Assistant? : r/unrealengine - Reddit, acceso: noviembre 16, 2025, https://www.reddit.com/r/unrealengine/comments/1nqdscq/nobodys_going_to_talk_about_the_official_ue_ai/
12. I Tried The New Unreal Engine 5.7 AI Assistant So You Don't Have To - Reddit, acceso: noviembre 16, 2025, https://www.reddit.com/r/unrealengine/comments/1nss0so/i_tried_the_new_unreal_engine_57_ai_assistant_so/
13. Unreal Engine MCP Installation Guide - Automate World Creation - YouTube, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=ct5dNJC-Hx4
14. 5 lessons to start vibe coding with Unreal MCP (works with Cursor, Windsurf, Claude Desktop) - YouTube, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=Rcj62DqrWXM
15. unreal.EditorAssetLibrary — Unreal Python 4.27 (Experimental) documentation - Epic Games Developers, acceso: noviembre 16, 2025, https://dev.epicgames.com/documentation/en-us/unreal-engine/python-api/class/EditorAssetLibrary?application_version=4.27
16. Consolidating Assets in Unreal Engine - Epic Games Developers, acceso: noviembre 16, 2025, https://dev.epicgames.com/documentation/en-us/unreal-engine/consolidating-assets-in-unreal-engine

17. Asset Replacement Tool - Asset Creation - Epic Developer ..., acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/asset-replacement-tool/621765
18. Replace Material Python Script?? - Unreal Engine Forums, acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/replace-material-python-script/120028
19. Help, what is the Python Script API to change a StaticMesh's Material? - Pipeline & Plugins, acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/help-what-is-the-python-script-api-to-change-a-staticmeshs-material/112887
20. Apply or change materials on skeletal meshes with Python - Unreal Engine Forums, acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/apply-or-change-materials-on-skeletal-meshes-with-python/117351
21. How to modify a static mesh component in python - Unreal Engine Forums, acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/how-to-modify-a-static-mesh-component-in-python/731291
22. Unreal 5.1 python edit blueprint components - Programming & Scripting, acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/unreal-5-1-python-edit-blueprint-components/742732
23. how to edit skeletal mesh component in Blueprint via python? - Unreal Engine Forums, acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/how-to-edit-skeletal-mesh-component-in-blueprint-via-python/155896
24. Replace references in bulk? - Programming & Scripting - Unreal Engine Forums, acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/replace-references-in-bul

k/446543
25. Audio Engine Overview in Unreal Engine - Epic Games Developers, acceso: noviembre 16, 2025, https://dev.epicgames.com/documentation/en-us/unreal-engine/audio-engine-overview-in-unreal-engine
26. Batch Rename Editor Widget Tutorial - Ryan DowlingSoka, acceso: noviembre 16, 2025, https://ryandowlingsoka.com/unreal/batch-rename-tool/
27. Batch Rename & Organize Like a Pro – Editor Utility Widget Tutorial in UE5 - YouTube, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=dzJXVPF0nSg
28. How to do editor scripting to modify Blueprint Assets ? : r/unrealengine - Reddit, acceso: noviembre 16, 2025, https://www.reddit.com/r/unrealengine/comments/1g3m9ob/how_to_do_editor_scripting_to_modify_blueprint/
29. Basic UE4 Editor Scripting 1 : Creating Objects with Python in an Editor Utility Widget, acceso: noviembre 16, 2025, https://www.artstation.com/blogs/b00merang/yPVv/basic-ue4-editor-scripting-1-creating-objects-with-python-in-an-editor-utility-widget
30. Unreal Engine 5.7 Release Notes - Epic Games Developers, acceso: noviembre 16, 2025, https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-7-release-notes
31. Passing Values from Unreal PCG to Actor Blueprints. Unreal Engine Tutorial - YouTube, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=JDkcGWKwQQk
32. PCG into Blueprints - Programming & Scripting - Epic Developer Community Forums, acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/pcg-into-blueprints/1731347
33. Use the Power of PCG to Randomly Spawn Enemies - YouTube, acceso: noviembre 16, 2025,

https://www.youtube.com/watch?v=DBzfVPLPVVA
34. Spawn random actor - Blueprint - Unreal Engine Forums, acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/spawn-random-actor/96925
35. How to spawn random actor? - Programming & Scripting - Unreal Engine Forums, acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/how-to-spawn-random-actor/1798431
36. Randomising your character's Skeletal Mesh on spawn in Unreal Engine - jay versluis, acceso: noviembre 16, 2025, https://www.versluis.com/2023/03/randomising-your-characters-skeletal-mesh-on-spawn-in-unreal-engine/
37. Unreal Engine 5 - Spawning Blueprints With PCG - YouTube, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=LgojHgebiNs
38. Select Random Mesh On Spawn - C++ - Unreal Engine Forums, acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/select-random-mesh-on-spawn/12830
39. Create Modular Buildings with PCG in Unreal Engine 5 – Full Step-by-Step Tutorial, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=pc4lKekw1Z8
40. Spawn on ANY Object with Unreal Engine PCG - YouTube, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=2PDEX_b_48I
41. Unreal Engine 5.7 Now Allows PCG To Be Masked By Shadows! - YouTube, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=Q6RrmQfgLyM
42. Epic Games Launches Unreal Engine 5.7 With Major Upgrades to Open Worlds, Rendering, and MetaHuman Tools - Inven Global, acceso: noviembre 16, 2025, https://www.invenglobal.com/articles/19890/epic-games-lau

nches-unreal-engine-57-with-major-upgrades-to-open-worlds-rendering-and-metahuman-tools
43. 3 Use Cases for Unreal Engine's New AI Assistant | UE 5.7 - YouTube, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=BpJYei2PboA
44. Unreal Engine 5.7 AI Assistant Test - YouTube, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=mbycsK5-YaI
45. Unreal Python Assistant - Powered by OpenAI - YouTube, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=iUWXfNiZccY
46. How to Use Unreal Engine MCP Server - Apidog, acceso: noviembre 16, 2025, https://apidog.com/blog/unreal-engine-mcp-server/
47. Setting up MCP Server for AI assisted game development on Unreal Engine -Claude Desktop and Windsurf - YouTube, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=pt52IUF-mIQ
48. Unreal Engine MCP server for Claude Desktop (early alpha preview) - GitHub, acceso: noviembre 16, 2025, https://github.com/runeape-sats/unreal-mcp
49. Unreal Editor, Now Talk to AI with Unreal-MCPython! - Game Development, acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/unreal-editor-now-talk-to-ai-with-unreal-mcpython/2555149
50. Vibe code in Unreal Engine 5 with this MCP | Blueprints+Scripting | Cursor, Windsurf, Claude Desktop - YouTube, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=ei2gulox9Yo
51. Ludus AI - Unreal Engine AI toolkit, acceso: noviembre 16, 2025, https://ludusengine.com/
52. Use AI Inside of Unreal Engine 5 to Create 3D Models and Help You With Blueprints, acceso: noviembre 16, 2025,

https://www.youtube.com/watch?v=c-S4ea4_NiQ
53. Ludus AI update for Unreal Engine (Scam or Gamechanger?) : r/unrealengine - Reddit, acceso: noviembre 16, 2025, https://www.reddit.com/r/unrealengine/comments/1eresai/ludus_ai_update_for_unreal_engine_scam_or/
54. Procedural Game Level Design with Deep Reinforcement Learning - arXiv, acceso: noviembre 16, 2025, https://arxiv.org/html/2510.15120v1
55. View of PCGRL: Procedural Content Generation via Reinforcement Learning, acceso: noviembre 16, 2025, https://ojs.aaai.org/index.php/AIIDE/article/view/7416/7341
56. [2001.09212] PCGRL: Procedural Content Generation via Reinforcement Learning - arXiv, acceso: noviembre 16, 2025, https://arxiv.org/abs/2001.09212
57. Learning Agents | Unreal Engine 5.7 Documentation | Epic Developer Community, acceso: noviembre 16, 2025, https://dev.epicgames.com/documentation/en-us/unreal-engine/API/PluginIndex/LearningAgents
58. Tutorial: Learning Agents Introduction - Unreal Engine Forums, acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/tutorial-learning-agents-introduction/835859
59. Learning to Drive (5.5) - Learning Agents (5.5) | Epic Developer Community, acceso: noviembre 16, 2025, https://dev.epicgames.com/community/learning/courses/GAR/unreal-engine-learning-agents-5-5/7dmy/unreal-engine-learning-to-drive-5-5
60. Tutorial: Learning to Drive - Unreal Engine Forums, acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/tutorial-learning-to-drive/1275490
61. Looking for using Unreal Engine 5 for Reinforcement

Learning simulations. Capabilities and limitations? : r/unrealengine - Reddit, acceso: noviembre 16, 2025, https://www.reddit.com/r/unrealengine/comments/1obmfme/looking_for_using_unreal_engine_5_for/
62. What's New in Unreal Engine 5.7: Full Breakdown of New Features and Upgrades - Vagon, acceso: noviembre 16, 2025, https://vagon.io/blog/what-s-new-in-unreal-engine-5-7
63. Updating Old Unreal Projects to NEWEST (& Fixing Broken Assets) - YouTube, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=pzbX4punBEg
64. MegaLights in Unreal Engine | Unreal Engine 5.7 Documentation ..., acceso: noviembre 16, 2025, https://dev.epicgames.com/documentation/en-us/unreal-engine/megalights-in-unreal-engine
65. Unreal Engine 5.7 Preview - Announcements - Epic Developer Community Forums, acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/unreal-engine-5-7-preview/2658958
66. Megalights in Unreal Engine 5: Everything You Need to Know - YouTube, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=VK1oFIoEVGE
67. Overview of Substrate Materials in Unreal Engine | Unreal Engine ..., acceso: noviembre 16, 2025, https://dev.epicgames.com/documentation/en-us/unreal-engine/overview-of-substrate-materials-in-unreal-engine
68. #UE5 Series: Substrate Materials | The Beginner's Guide Part 1 - YouTube, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=YDeptWduHNc
69. Unreal Engine 5.7 Released - Announcements - Epic Developer Community Forums, acceso: noviembre 16, 2025, https://forums.unrealengine.com/t/unreal-engine-5-7-released/2673913
70. Unreal Engine 5.7 Nanite Foliage: A Game-Changer for

Real-Time ..., acceso: noviembre 16, 2025, https://medium.com/@thirdspaceinteractive/unreal-engine-5-7-nanite-foliage-a-game-changer-for-real-time-vegetation-c8e9692df3b5

71. Unreal Engine 5.7 (Preview) – Nanite Foliage | Procedural Vegetation Editor Explained!, acceso: noviembre 16, 2025, https://www.youtube.com/watch?v=CBOk3ycCeN4