Universitat Oberta de Catalunya

Desarrollo de las nuevas tecnologías en el modelado procedural (TFG)

Autor: Alberto de la Guía Marín
Tutor: Manel Fernández Rodríguez
Profesor: Joan Arnedo Moreno

Grado de Multimedia
Área de Videojuegos
08 de Julio de 2024

Créditos/Copyright

FICHA DEL TRABAJO FINAL

Título del trabajo:	**Desarrollo de las nuevas tecnologías en el modelado procedural**
Nombre del autor:	*Alberto de la Guía Marín*
Nombre del colaborador/a docente:	*Manel Fernández Rodríguez*
Nombre del PRA:	*Joan Arnedo Moreno*
Fecha de entrega:	*07/2024*
Titulación o programa:	*Grado de Multimedia*
Área del Trabajo Final:	*TFG Videojuegos*
Idioma del trabajo:	*Español*
Palabras clave:	*Procedural, Videojuegos, TFG*

Resumen del Trabajo

Este Trabajo de Fin de Grado se centra en el desarrollo de una herramienta, o conjunto de herramientas, para la creación de terrenos procedurales en videojuegos. Utilizando Houdini SideFx y Unreal Engine como programas base, el proyecto busca facilitar el diseño y la personalización de escenarios naturales 3D mediante la generación procedural de diferentes tipos de terreno, así como la generación y colocación de elementos como rocas, plantas y árboles.

A través de la herramienta desarrollada, los usuarios podrán crear y editar terrenos de manera intuitiva y eficiente, gracias a la creación procedural de mapas de información que permiten distribuir los distintos elementos necesarios para la creación de escenarios naturales. Esta capacidad de automatización y personalización es crucial para satisfacer las crecientes demandas de la industria de los videojuegos, donde los escenarios son cada vez más grandes y detallados.

El objetivo de estas herramientas es ofrecer una solución a la necesidad de rapidez y eficiencia en la creación de escenarios. Las herramientas procedurales no solo aceleran el proceso de creación de terrenos, sino que también facilitan la realización de cambios y la generación de variaciones del mismo entorno mediante el uso de semillas (*seeds*). Este enfoque mejora significativamente los tiempos de desarrollo y potencia la creatividad, permitiendo a los diseñadores explorar una gama más amplia de posibilidades sin las limitaciones de los métodos tradicionales de diseño de terrenos.

Video demostrativo: https://vimeo.com/948111244
Video presentación:
Password: delaguia88TFG

Repositorio de la herramienta: https://github.com/AlbertodelaGuia/delaguia88TFG

Abstract:

This Final Degree Project focuses on the development of a tool, or a set of tools, for creating procedural terrains in video games. Using Houdini SideFx and Unreal Engine as the primary software, the project aims to facilitate the design and customisation of 3D natural environments through the procedural generation of various terrain types, as well as the generation and placement of elements such as rocks, plants, and trees.

By using this tool, users will be able to create and edit terrains intuitively and efficiently, thanks to the procedural creation of information maps that serve to distribute the different elements necessary for crafting natural environments. This automation and customisation capability is crucial for meeting the growing demands of the video game industry, where environments are becoming increasingly large and detailed.

The purpose of these tools is to provide a solution to the need for speed and efficiency in environment creation. Procedural tools not only accelerate the terrain creation process but also facilitate making changes and generating variations of the same environment using seeds. This approach significantly improves development times and enhances creativity, allowing designers to explore a wider range of possibilities without the constraints of traditional terrain design methods.

Dedicatoria/Cita

A mi mujer por su apoyo incondicional y por ser un ejemplo a seguir, sin ella nada de esto habría sido posible.

Agradecimientos

Me gustaría agradecer a mi familia, carnal y política y en especial a mis padres, por animarme e insistirme siempre para que no me rindiera. Agradecer también al tutor del trabajo, Manel Fernández toda su ayuda y colaboración.

Notaciones y Convenciones

La memoria ha sido escrita con la tipografía Arial de tamaño 11 con un interlineado de 1,15.

El estilo de citación usado en la bibliografía ha sido el Harvard.

Índice

1. Introducción 12
1.1. Introducción/Prefacio 12
1.2. Descripción/Definición 13
1.3. Objetivos generales 14
1.3.1. Objetivos principales 14
1.3.2. Objetivos secundarios 14
1.4. Metodología y proceso de trabajo 15
1.5. Planificación 16
1.5.1. Diagrama de Gantt 17
1.6. Presupuesto 18
1.6.1. Equipo Humano 18
1.6.2. Equipamiento técnico 19
1.6.3. Otros recursos 19
1.6.4. Presupuesto total 19
1.7. Estructura del resto del documento 20
2. Análisis de mercado 21
2.1. Público objetivo (*i.e. target audience*) y perfiles de usuario 21
2.1.1. Estudios de desarrollo de videojuegos 21
2.1.2. Estudios cine de imagen real (VFX) 22
2.1.3. Estudios de animación 3D 23
2.1.4. Agencias de publicidad 23
2.2. Fichas de usuario 24
2.3. Competencia/Antecedentes 26
2.3.1. Herramientas nativas de Unreal Engine 26
2.3.2. Plugins externos 28
2.3.3. Tabla comparativa 29
2.4. Análisis DAFO 30
3. Propuesta 31
3.1. Definición de objetivos y especificaciones del producto 31
3.1.1. Objetivos y características principales 31
3.1.2. Objetivos y características secundarias 32
3.1.3. Especificaciones técnicas 32
3.2. Modelo de negocio 33

3.3. Estrategia de marketing 34

4. Diseño 35

4.1. Arquitectura general de la aplicación/sistema/servicio 35

4.2. Arquitectura de la información y diagramas de navegación 37

4.2.1. Creación y configuración de máscaras. 37

4.2.2. Creación y configuración del scatter. 39

4.3. Diseño gráfico e interfaces 42

4.3.1. Usabilidad /UX 45

4.4. Lenguajes de programación y APIs utilizados 46

4.5. Desarrollo de la herramienta en Houdini SideFx 47

4.5.1. Árbol de nodos 47

4.5.2. Interfaz de usuario 52

5. Implementación 54

5.1. Requisitos de instalación 54

5.2. Instrucciones de instalación 55

6. Demostración 56

6.1. Instrucciones de uso 56

6.1.1. Preparación previa 56

6.1.2. Houdini *Parameters* 60

6.1.3. *Masks Options* 60

6.1.4. *Scatter Options* 65

6.1.5. *Assets Options* 67

6.2. Prototipos 68

6.3. Tests 69

6.3.1. Pruebas de Funcionamiento 69

6.3.2. Pruebas de Rendimiento 69

6.3.3. Pruebas de Usabilidad 69

6.4. Ejemplos de uso del producto 70

7. Conclusiones y líneas de futuro 74

7.1. Conclusiones 74

7.2. Líneas de futuro 75

7.2.1. Interactividad con Geometrías y Creación de Máscaras 75

7.2.2. Resolución de Intersecciones y Mejoras en la Usabilidad 75

Bibliografía 76

Anexos 76

Figuras y tablas

Índice de figuras

Figura 1: Imagen ficticia de terreno procedural 12
Figura 2: Diagrama de Gantt 17
Figura 3: Diagrama general de funcionamiento de la herramienta 36
Figura 4: Diagrama del funcionamiento de la herramienta. Sección máscaras 38
Figura 5: Diagrama del funcionamiento de la herramienta. Sección *scatter*. 41
Figura 6: Interfaz general de la herramienta 42
Figura 7: Elemento interfaz - Checkboxes 42
Figura 8: Elemento interfaz - Menú desplegable 42
Figura 9: Elemento interfaz - Curva de remapeo 43
Figura 10: Elemento interfaz - Deslizadores 43
Figura 11: Elemento interfaz - Campos de entrada numéricos 43
Figura 12: Elemento interfaz - Botones 43
Figura 13: Visualización de la geometría base 44
Figura 14: Visualización de máscara en viewport 44
Figura 15: Visualización de las instancias 44
Figura 16: Visualización del scatter 44
Figura 17: Houdini Setup – Árbol de nodos 47
Figura 18: Árbol de nodos. Sección vertex color 48
Figura 19: Árbol de nodos. Ejemplo *for each* loop 48
Figura 20: Árbol de nodos. Sección tipos de máscaras 49
Figura 21: Árbol de nodos. Sección modificar máscara 49
Figura 22: Código HScript. Offset position 50
Figura 23: Árbol de nodos. Sección scatter 50
Figura 24: Árbol de nodos. Sección assets 51
Figura 25: Árbol de nodos. Sección display options 51
Figura 26: Houdini Setup – Interfaz de usuario 52
Figura 27: Houdini Setup – Configuración de Interfaz de usuario 52
Figura 28: Houdini Setup – Vinculación de parámetros 53
Figura 29: Unreal – Importación de la herramienta 55
Figura 30: Geometría base 56
Figura 31: Blueprints para instanciar 56
Figura 32: Creación de material vertex color 57
Figura 33: Duplicado de geometría base 57
Figura 34: Aplicación de material vertex color 57
Figura 35: Creación de máscara manual 58
Figura 36: Inicialización de la herramienta 58
Figura 37: Asignación a la geometría base 59
Figura 38: Geometría base añadida 59
Figura 39: Secciones herramienta 60
Figura 40: Opciones de máscaras 60

Figura 41: Rampa de color utilizada en la visualización de las máscaras 60
Figura 42: Visualización de la máscara final 61
Figura 43: Añadir nueva máscara 61
Figura 44: Opciones de las máscaras 61
Figura 45: Desenfocar máscara 62
Figura 46: Remapear máscara 62
Figura 47: Opciones de combinación 63
Figura 48: Tipos de máscarass 63
Figura 49: Máscara noise 63
Figura 50: Máscara direction 63
Figura 51: Máscara normal 64
Figura 52: Máscara Ambient Oclusion 64
Figura 53: Máscara height 64
Figura 54: Opciones de scatter 65
Figura 55: Opciones de densidad 65
Figura 56: Opciones de orientación 66
Figura 57: Opciones de escala 66
Figura 58: Opciones de posición 66
Figura 59: Opciones de assets 67
Figura 60: Ejemplo assets 67
Figura 61: Ejemplos de uso. Interfaz 70
Figura 62: Ejemplos de uso. Diferentes elementos 1 71
Figura 63: Ejemplos de uso. Diferentes elementos 2 72
Figura 64: Ejemplos de uso. Diferentes geometrías base 73

Índice de tablas

Tabla 1: Planificación general 16
Tabla 2: Planificación detallada 16
Tabla 3: Presupuesto equipo humano 18
Tabla 4: Presupuesto equipamiento técnico 19
Tabla 5: Presupuesto equipo humano 19
Tabla 6: Presupuesto total 19
Tabla 7: Comparativa competencia 29
Tabla 8: Análisis DAFO 30

1. Introducción

1.1. Introducción/Prefacio

En computación, **generación por procedimientos** o **procedural** es el método de creación de datos con algoritmos en lugar de forma manual. La generación procedural en el desarrollo de videojuegos ha marcado una evolución clave, ofreciendo métodos para crear mundos ricos y extensos con eficiencia. Esta técnica permite una diversidad y profundidad en los juegos previamente inalcanzables, cambiando por completo tanto cómo se diseñan los entornos en los videojuegos, como la experiencia de los jugadores.

Este Trabajo de Fin de Grado busca facilitar el diseño y la personalización de escenarios 3D mediante la generación procedural de diferentes tipos de terreno, así como la generación y colocación de los elementos que componen este tipo de escenarios como rocas, plantas, árboles, etc. Para ello se propone desarrollar herramientas avanzadas para la creación de terrenos, combinando eficiencia con la capacidad de personalización, con el objetivo de superar algunas de las limitaciones actuales y contribuir a la industria de los videojuegos.

Figura 1: Imagen ficticia de terreno procedural

1.2. Descripción/Definición

El avance de la generación procedural de terrenos en videojuegos es un testimonio del crecimiento tecnológico dentro de la industria, permitiendo el desarrollo de mundos inmersivos y detallados de manera eficiente. Originada en la década de los 80 con innovaciones en títulos como "*Rogue*" y "*Elite*", esta técnica ha evolucionado en los últimos años para abarcar una gama más amplia de aplicaciones, desde la creación de vastos universos en *"Minecraft*" y "*No Man's Sky*" hasta la optimización del proceso de desarrollo en juegos como "*Horizon: Zero Dawn*". Esta evolución refleja no solo un cambio en la capacidad de los juegos para generar contenido único y dinámico, sino también en cómo los desarrolladores abordan la creación de entornos de juego.

La necesidad de soluciones eficientes para la generación de terrenos se ha vuelto cada vez más crítica, dada la creciente demanda de videojuegos que ofrecen mundos ricos y expansivos. El enfoque tradicional, que a menudo requiere una considerable inversión de tiempo y recursos en el diseño manual de niveles, limita la capacidad de los desarrolladores para experimentar e iterar rápidamente. En este contexto, el proceduralismo se destaca como una estrategia valiosa para mejorar la eficiencia y fomentar la creatividad durante el desarrollo de videojuegos.

Este Trabajo de Fin de Grado se centra en abordar la necesidad de herramientas más accesibles y potentes para la generación procedural de terrenos, con el fin de facilitar el desarrollo de videojuegos modernos. El objetivo es crear una herramienta que no solo permita la generación de terrenos complejos y detallados de forma más eficiente, sino que también ofrezca a los desarrolladores la flexibilidad para personalizar y adaptar estos entornos a sus necesidades específicas. Al hacerlo, se espera que este proyecto contribuya significativamente a la metodología de desarrollo de videojuegos, permitiendo la creación de mundos que anteriormente estaban fuera del alcance debido a limitaciones de recursos o tecnología.

Como resultado de este trabajo, se entregará una suite de herramientas que integran capacidades avanzadas de generación procedural con una interfaz de usuario intuitiva, diseñadas para ser utilizadas en el desarrollo de videojuegos de última generación. Con estas herramientas se pretende no solo acelerar el proceso de creación de niveles, sino enriquecer la calidad y diversidad de los entornos de videojuegos.

1.3. Objetivos generales

1.3.1. Objetivos principales

Objetivos de la aplicación:

- Desarrollar en Houdini y Unreal Engine un conjunto de herramientas que sirvan para crear el mayor número de elementos procedurales para escenarios 3D de videojuegos.
- Desarrollar una herramienta que permita crear mapas de información, tanto de manera procedural como de manera manual, para crear los diferentes mapas de alturas, densidad, escala, rotación, etc., que posteriormente serán utilizados por el resto de herramientas para generar los *scatters* procedurales de los diferentes elementos 3D.
- Desarrollar una herramienta que lea mapas de información y construya *scatters* procedurales de diferentes elementos 3D, permitiendo a su vez, editar de manera procedural y asignando aleatoriedad a valores de rotación, escala, posición, elemento 3D, etc.

Objetivos para el cliente/usuario:

- Facilitar a los usuarios y empresas desarrolladoras de videojuegos la creación de escenarios 3D mediante técnicas procedurales.
- Ahorrar costes en el proceso de creación de grandes escenarios 3D.
- Mejorar los procesos de edición y rectificación de los escenarios 3D.

Objetivos personales del autor del TFG:

- Adquirir conocimientos y experiencia práctica en el manejo del motor de videojuego Unreal.
- Ampliar las oportunidades laborables, accediendo a un mercado en constante crecimiento y evolución.
- Completar con éxito el TFG, para poder continuar avanzando académicamente y laboralmente.

1.3.2. Objetivos secundarios

Objetivos adicionales de la aplicación:

- Desarrollar una herramienta que permita la generación de terrenos procedurales, permitiendo crear diferentes tipos de terrenos, como agua, desierto, montaña, etc.
- Desarrollar una herramienta que permita crear proceduralmente, los diferentes elementos 3D que posteriormente se utilizarán para dispersarlos por el escenario 3D.

1.4. Metodología y proceso de trabajo

El propósito de este TFG es crear una serie de herramientas que, aunque actualmente ya han sido desarrolladas por diferentes estudios, no son de dominio público, y por tanto no se puede acceder a ellas. Estas herramientas pretenden ser una primera versión de un conjunto de herramientas que sí tengan una aplicación real para el desarrollo de escenarios en el mundo de los videojuegos.

Para este TFG la estrategia seleccionada es el desarrollo incremental, concretamente se utilizará la metodología de desarrollo SCRUM, ya que permite dividir el desarrollo del producto en pequeños segmentos e ir añadiendo características y funcionalidades en cada iteración que se actualizan con cada nueva versión. Esta metodología además de ser idónea para el tipo de proyecto se ajusta perfectamente con las entregas correspondientes del TFG

Las herramientas se crean utilizando Houdini como software principal, para el desarrollo de las herramientas procedurales y Unreal Engine, como motor gráfico para su implementación en el sector de los videojuegos. La elección de estos softwares viene dada de varias razones, por un lado: Houdini es actualmente y desde hace años, el software de referencia en cuanto a creación de elementos procedurales para el sector del cine y videojuegos. Por otro lado, Unreal Engine, es actualmente el software de referencia del sector, y aunque compite con Unity3D, Unreal ha demostrado tener una mayor calidad técnica y visual. Además, Houdini cuenta con una integración directa en Unreal Engine, lo que permite utilizar las herramientas desarrolladas en Houdini directamente en Unreal, siendo este último punto un hecho fundamental para utilizar ambos softwares.

El desarrollo se divide en tres fases principales:

1. **Fase de Diseño:**
   - Diseño de las herramientas, definiendo los mínimos aceptables y los máximos deseables.

2. **Fase de Desarrollo:**
   - Desarrollo de las diferentes herramientas en Houdini, empezando por prototipos simples y posterior desarrollo.

3. **Fase de Implementación:**
   - Implementación de las herramientas desarrolladas en Unreal, configurando y desarrollando la parte final de las herramientas para su correcto funcionamiento en el motor gráfico.

1.5. Planificación

Siguiendo la metodología SCRUM, las diferentes fases de desarrollo (SPRINT) se corresponderán con las diferentes entregas del TFG:

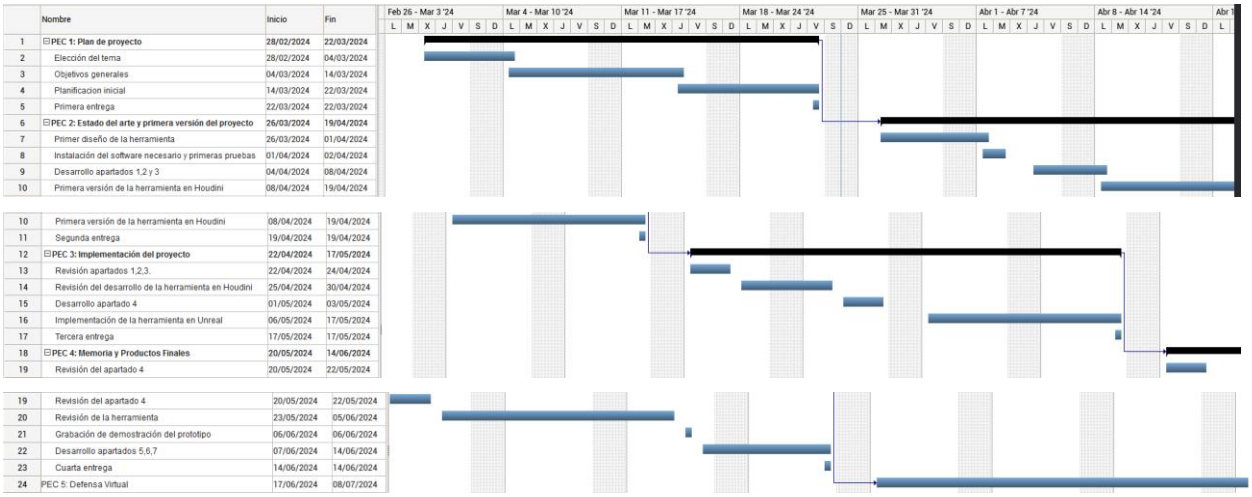
SPRINT	Nombre	Inicio	Fin
1	PEC 1: Plan de proyecto	28/02/24	24/03/24
2	PEC 2: Estado del arte y primera versión del proyecto	25/03/24	21/04/24
3	PEC 3: Implementación del proyecto	22/04/24	19/05/24
4	PEC 4: Memoria y Productos Finales	20/05/24	16/06/24
5	PEC 5: Defensa Virtual	17/06/24	08/07/24

Tabla 1: Planificación general

SPRINT	Nombre	Inicio	Fin
1	**PEC 1: Plan de proyecto**		
	Elección del tema	28/02/24	03/03/24
	Objetivos generales	04/03/24	14/03/24
	Planificación inicial	14/03/24	24/03/24
	Primera entrega	24/03/24	24/03/24
2	**PEC 2: Estado del arte y primera versión del proyecto**		
	Primer diseño de la herramienta	25/03/24	31/03/24
	Instalación del software necesario y primeras pruebas	01/04/24	02/04/24
	Desarrollo apartados 1,2 y 3	03/04/24	07/04/24
	Primera versión de la herramienta en Houdini	08/04/24	21/04/24
	Segunda entrega	21/04/24	21/04/24
3	**PEC 3: Implementación del proyecto**		
	Revisión apartados 1,2,3.	22/04/24	24/04/24
	Revisión del desarrollo de la herramienta en Houdini	25/04/24	30/04/24
	Desarrollo apartado 4	01/05/24	04/05/24
	Implementación de la herramienta en Unreal	05/05/24	19/05/24
	Tercera entrega	19/05/24	19/05/24
4	**PEC 4: Memoria y Productos Finales**		
	Revisión del apartado 4	20/05/24	22/05/24
	Revisión de la herramienta	23/05/24	05/06/24
	Grabación de demostración del prototipo	06/06/24	06/06/24
	Desarrollo apartados 5,6,7	07/06/24	16/06/24
	Cuarta entrega	16/06/24	16/06/24
5	**PEC 5: Defensa Virtual**	17/06/24	08/07/24

Tabla 2: Planificación detallada

1.5.1.Diagrama de Gantt



	Nombre	Inicio	Fin
1	**PEC 1: Plan de proyecto**	28/02/2024	22/03/2024
2	Elección del tema	28/02/2024	04/03/2024
3	Objetivos generales	04/03/2024	14/03/2024
4	Planificacion inicial	14/03/2024	22/03/2024
5	Primera entrega	22/03/2024	22/03/2024
6	**PEC 2: Estado del arte y primera versión del proyecto**	26/03/2024	19/04/2024
7	Primer diseño de la herramienta	26/03/2024	01/04/2024
8	Instalación del software necesario y primeras pruebas	01/04/2024	02/04/2024
9	Desarrollo apartados 1,2 y 3	04/04/2024	08/04/2024
10	Primera versión de la herramienta en Houdini	08/04/2024	19/04/2024
10	Primera versión de la herramienta en Houdini	08/04/2024	19/04/2024
11	Segunda entrega	19/04/2024	19/04/2024
12	**PEC 3: Implementación del proyecto**	22/04/2024	17/05/2024
13	Revisión apartados 1,2,3.	22/04/2024	24/04/2024
14	Revisión del desarrollo de la herramienta en Houdini	25/04/2024	30/04/2024
15	Desarrollo apartado 4	01/05/2024	03/05/2024
16	Implementación de la herramienta en Unreal	06/05/2024	17/05/2024
17	Tercera entrega	17/05/2024	17/05/2024
18	**PEC 4: Memoria y Productos Finales**	20/05/2024	14/06/2024
19	Revisión del apartado 4	20/05/2024	22/05/2024
19	Revisión del apartado 4	20/05/2024	22/05/2024
20	Revisión de la herramienta	23/05/2024	05/06/2024
21	Grabación de demostración del prototipo	06/06/2024	06/06/2024
22	Desarrollo apartados 5,6,7	07/06/2024	14/06/2024
23	Cuarta entrega	14/06/2024	14/06/2024
24	PEC 5: Defensa Virtual	17/06/2024	08/07/2024

Figura 2: Diagrama de Gantt

1.6. Presupuesto

El planteamiento del desarrollo de la herramienta está confeccionado para desarrollarla por una sola persona, asumiendo los diferentes roles necesarios en cada etapa del proyecto. Este enfoque implica que el desarrollador se encargará de todas las fases, asumiendo los diferentes roles necesarios en cada etapa del proyecto.

A continuación, se presenta una estimación detallada de los costos asociados al proyecto, desglosada en tres categorías principales: equipo humano, equipamiento técnico y otros recursos necesarios para la ejecución exitosa del mismo. El presupuesto se ha elaborado considerando las horas dedicadas a cada tarea específica y los costos de los recursos técnicos esenciales para el desarrollo e implementación de la herramienta.

1.6.1.Equipo Humano

Concepto	Cantidad horas	Precio / Hora (€)	Costo Total (€)
Planificación y diseño			
Elección del tema	4	30 €	120 €
Objetivos generales	8	30 €	240 €
Planificación inicial	24	30 €	720 €
Presupuesto	8	30 €	240 €
Subtotal	44		**1.320 €**
Estudio de mercado			
Público objetivo	16	30 €	480 €
Fichas de usuario	8	30 €	240 €
Competencia	40	30 €	1.200 €
Análisis DAFO	4	30 €	120 €
Subtotal	68		**2.040 €**
Desarrollo y programación			
Primera versión de la herramienta - Houdini	80	30 €	2.400 €
Primera implementación - Unreal	24	30 €	720 €
Segunda versión de la herramienta - Houdini	40	30 €	1.200 €
Segunda implementación - Unreal	24	30 €	720 €
Subtotal	168		**5.040 €**
Testeos y demostración			
Testeos	24	30 €	720 €
Demostración	32	30 €	960 €
Documentación	16	30 €	480 €
Subtotal	72		**2.160 €**
Total equipo humano	308		**10.560 €**

Tabla 3: Presupuesto equipo humano

1.6.2.Equipamiento técnico

Concepto	Cantidad	Precio unitario	Costo Total (€)
Hardware			
Estación de trabajo de altas prestaciones	1	3.000 €	3.000 €
Servidor NAS	1	500 €	500 €
Subtotal Hardware			**3.500 €**
Software			
Houdini SideFx	1	270 €	270 €
Unreal Engine	1	0 €	0 €
Windows	1	250 €	250 €
Subtotal Software			**520 €**
Total equipamiento técnico			**4.020 €**

Tabla 4: Presupuesto equipamiento técnico

1.6.3.Otros recursos

Concepto	Cantidad (Meses)	Precio unitario	Costo Total (€)
Alquiler oficina	6	600 €	3.600 €
Electricidad	6	60 €	360 €
Internet	6	60 €	360 €
Total otros			**4.320 €**

Tabla 5: Presupuesto equipo humano

1.6.4.Presupuesto total

Equipo humano	10.560 €
Equipamiento técnico	4.020 €
Otros recursos	4.320 €
Total presupuesto	**18.900 €**

Tabla 6: Presupuesto total

1.7. Estructura del resto del documento

El documento que sigue se estructura en varias secciones clave que abarcan desde el análisis del mercado y la propuesta del proyecto hasta el diseño, implementación, demostración y conclusiones del mismo. A continuación, se detalla el contenido de cada capítulo y su relación con el trabajo en general.

2.Análisis de Mercado
Este capítulo analiza el entorno actual del mercado de herramientas de generación procedural, identificando el público objetivo y los perfiles de usuario, así como la competencia y antecedentes en el sector. También se realiza un análisis DAFO para identificar fortalezas, debilidades, oportunidades y amenazas.

3.Propuesta
En esta sección se define la propuesta de la herramienta desarrollada, describiendo sus objetivos y especificaciones técnicas. Se detallan las características principales y secundarias, así como el modelo de negocio y la estrategia de marketing.

4.Diseño
Se detalla la arquitectura general de la aplicación, la arquitectura de la información, el diseño gráfico y las interfaces, así como los lenguajes de programación y APIs utilizados.

5.Implementación
Aquí se describen los requisitos de instalación y las instrucciones para la implementación de la herramienta. Se explica cómo instalar y configurar el software necesario para utilizar la herramienta desarrollada.

6.Demostración
Se incluyen instrucciones de uso detalladas, prototipos y pruebas realizadas para validar el funcionamiento de la herramienta. También se proporcionan ejemplos de uso.

7.Conclusiones y Líneas de Futuro
Se presentan las conclusiones del proyecto, reflexionando sobre los resultados obtenidos y el aprendizaje adquirido. Además, se proponen posibles líneas de futuro para continuar el desarrollo de la herramienta y mejorar sus capacidades.

2. Análisis de mercado

En la actualidad, la industria del contenido visual exige métodos más rápidos y detallados para la creación de entornos digitales. En este marco, el proyecto que ha sido diseñado bajo un modelo empresa a empresa “B2B”, ofrece una herramienta desarrollada específicamente para Unreal Engine, que permite realizar scatters procedurales de elementos, optimizando así la creación eficaz de estos escenarios.

2.1. Público objetivo (*i.e. target audience*) y perfiles de usuario

El público objetivo de este análisis de mercado, abarca empresas en sectores como el desarrollo de videojuegos, la producción cinematográfica y la publicidad. Las cuales operan o podrían estar interesadas en operar con Unreal Engine y que encontrarán en esta innovación tecnológica una solución valiosa para incorporar en sus flujos de trabajo.

2.1.1. Estudios de desarrollo de videojuegos

Es el público objetivo más habitual y directo, ya que son los que necesariamente tienen que utilizar motores gráficos de videojuegos para realizar sus productos. Entre los diferentes estudios de desarrollo de videojuegos generalmente encontramos dos tipos de perfiles:

- **Estudios indie:** Estos pequeños equipos o individuos con presupuestos limitados, buscan soluciones que les permitan competir en un mercado dominado por grandes estudios y títulos con presupuestos millonarios. Por lo general, no tienen los recursos para desarrollar sus propias herramientas, por lo que suelen interesarse en adquirir este tipo herramientas de manera externa.

- **Estudios grandes:** Empresas establecidas en la industria del videojuego con mayores recursos, equipos especializados y proyectos más ambiciosos. Están interesados en herramientas que puedan integrarse sin problemas en sus pipelines establecidos, ofrecer altos niveles de personalización, y soportar la creación de mundos vastos y complejos para videojuegos AAA. Además, buscan optimizaciones que reduzcan la carga de renderizado y mejoren la experiencia del jugador en diversas plataformas de hardware.

2.1.2. Estudios cine de imagen real (VFX)

Los estudios de cine están constantemente buscando herramientas que permitan innovar en la manera de crear y visualizar secuencias complejas antes y durante el rodaje. El uso de herramientas que permiten la visualización en tiempo real permite la planificación de todo el rodaje, además de la optimización y ejecución de las escenas más complejas.

En este marco encontramos diferentes aplicaciones:

- **Previs (previsualizaciones 3D):** Previsualizaciones 3D de la película que se realiza antes del rodaje para planificarlo. En ella los directores y el equipo planifican las secuencias, eligen las cámaras, su duración, el *dressing* del plano, la iluminación, etc. Permite modificaciones rápidas y eficientes ahorrando costes y mejorando la calidad del producto final.

- **Techvis (previsualizaciones técnicas):** Las secuencias que tienen una complejidad técnica muy grande suelen ser también las más costosas. En las *techvis* se crean escenarios que posteriormente se construirán, así como los movimientos de cámaras. Una vez finalizada se extraen todos estos datos del modelo 3D para su posterior recreación en el "mundo real". Planos de construcción y medidas, modelos 3D para impresión o incluso movimientos de cámara son algunos de los elementos que se obtienen de una *techvis*. Este proceso permite afrontar con éxito secuencias muy complejas además de abaratar costes y minimizar imprevistos.

- **Visualización de escenarios en tiempo real:** Como método alternativo a las tradicionales pantallas verdes, se puede construir un escenario de pantallas en las que permita visualizar el escenario 3D en tiempo real. Este tipo de sets de rodaje ofrece a los actores, cámaras y directores un entorno más inmersivo el cual sirve para mejorar la actuación y grabación en el set. Además, uno de los beneficios más importantes es que los elementos reales y los actores se ven afectados por la iluminación, colores, etc. del escenario 3D, dando como resultado efectos visuales mucho más integrados y naturales.

2.1.3. Estudios de animación 3D

Estos estudios se especializan en la creación de contenido animado para cine, televisión y medios digitales. Con el notable avance en las tecnologías de renderizado, como el *Ray Tracing* y el *Path Tracing* en tiempo real, estos estudios están explorando cada vez más el uso de motores gráficos de videojuegos (como Unreal Engine) para producir animaciones. Estos avances les permiten alcanzar niveles de realismo visual y efectos de iluminación que antes solo eran posibles con software de renderizado tradicional mucho más lento y costoso. Esto supone un cambio muy significativo a nivel de costes, ya que el renderizado de una película de animación 3D, aunque varía, puede suponer fácilmente el 25% del presupuesto total.

2.1.4. Agencias de publicidad

Las agencias de publicidad a menudo trabajan con presupuestos y plazos ajustados. Herramientas que permiten una visualización y modificación rápidas y eficientes de los activos digitales pueden ser extremadamente valiosas, permitiendo múltiples iteraciones sin costos prohibitivos. Además, con el creciente interés en campañas publicitarias que incorporan AR y VR, las agencias necesitan herramientas que puedan crear y manejar estos contenidos. Las mismas tecnologías de renderizado en tiempo real y creación de entornos inmersivos utilizadas en el desarrollo de videojuegos y la producción cinematográfica son aplicables aquí para crear experiencias de usuario novedosas y atractivas.

2.2. Fichas de usuario

Estudio de Desarrollo de Videojuegos

Nombre de la Empresa: Ice Games Studios.

- **Industria**: Desarrollo de Videojuegos.
- **Dimensión**: Grande.
- **Facturación anual**: Alta.
- **Ubicación**: Londres, Reino Unido.
- **Disposición a invertir**: Baja.
- **Descripción**: Estudio líder en la industria de videojuegos, busca mantener su posición en el mercado a través del desarrollo y uso de tecnologías de última generación. Generalmente desarrolla su propias herramientas y tecnología.
- **Necesidades**: Desarrollo de escenarios masivos con gráficos de alta calidad para juegos AAA, mejorar la eficiencia de los juegos, e integración y compatibilidad con sus herramientas y flujo de trabajo.
- **Beneficios buscados**: Mejorar el resultado final de sus videojuegos, reduciendo el tiempo de desarrollo y mejorando la experiencia del jugador con gráficos de vanguardia.

Estudio de Desarrollo de Videojuegos Indie

Nombre de la Empresa: BlackHammer Games.

- **Industria**: Desarrollo de Videojuegos.
- **Dimensión**: Pequeña.
- **Facturación anual**: Baja.
- **Ubicación**: Barcelona, España.
- **Disposición a invertir**: Moderada.
- **Descripción**: Estudio Indie que busca entrar en el mercado de los videojuegos. Se centra en un enfoque creativo y con mecánicas innovadoras para diferenciarse de la competencia. Está dispuesto a invertir siempre que el coste-beneficio suponga un cambio significativo para el estudio.
- **Necesidades**: Software que facilite la creación rápida de prototipos y el desarrollo de juegos con gráficos de alta calidad y rendimiento optimizado en diversas plataformas.
- **Beneficios buscados**: Reducir los tiempos de desarrollo y los costos operativos para mantener la viabilidad financiera del estudio. Mejorar el resultado final de sus videojuegos, para competir en el mercado.

Estudio de Producción Cinematográfica

Nombre de la Empresa: Piranha Studios

- **Industria**: Producción Cinematográfica.
- **Dimensión**: Alta
- **Facturación anual**: Alta.
- **Ubicación**: Hollywood, California.
- **Disposición a invertir**: Media.
- **Descripción**: Estudio cinematográfico especializado en películas de gran presupuesto, valora la eficiencia y calidad en la creación de efectos visuales.

- **Necesidades**: Herramientas avanzadas para la creación y edición de efectos visuales, así como capacidades de previsualización en tiempo real que faciliten el proceso creativo y técnico durante la producción.
- **Beneficios buscados**: Optimización del proceso de producción para reducir tiempos muertos y costos, mientras se mejora la precisión en la creación de efectos especiales y se potencia la colaboración entre diferentes departamentos.

Estudio de Animación 3D

Nombre de la Empresa: Gusi Animation Studios

- **Industria**: Producción Cinematográfica de Animación 3D.
- **Dimensión**: Mediana
- **Facturación anual**: Mediana
- **Ubicación**: Valencia, España
- **Disposición a invertir**: Alta, con foco en adquisición de tecnología de vanguardia.
- **Descripción**: Estudio innovador de animación 3D que busca mantener su competitividad con tecnología de vanguardia.
- **Necesidades**: Paquete 3D que integre tecnologías de iluminación en tiempo real como *Ray Tracing* y *Path Tracing*.
- **Beneficios Buscados**: Reducir los tiempos de producción y los costes de renderizado, sin sufrir una pérdida de calidad, permitiendo de esta manera obtener una posición más competitiva en el mercado.

Agencia de Publicidad

Nombre de la Empresa: PullForward Ad Agency

- **Industria**: Publicidad y Marketing Digital
- **Dimensión**: Mediana
- **Facturación Anual**: Alta
- **Ubicación**: Madrid, España.
- **Disposición a invertir**: Baja
- **Descripción**: Líder en campañas innovadoras, la agencia busca integrar soluciones tecnológicas que aceleren los procesos de producción sin sacrificar la calidad, mejorando así su oferta a los clientes y optimizando los tiempos de entrega.
- **Necesidades**: Herramientas que faciliten el desarrollo y la iteración rápida de contenido digital, permitiendo realizar numerosas modificaciones en cortos periodos de tiempo para responder eficazmente a las demandas del cliente.
- **Beneficios buscados**: Aumentar la eficiencia operativa y reducir costos mediante la optimización de los flujos de trabajo de campañas publicitarias, lo que permite ofrecer servicios más ágiles y coste-eficientes.

2.3. Competencia/Antecedentes

En el desarrollo de escenarios 3D a gran escala, la generación procedural de elementos es esencial para alcanzar los resultados deseados sin una inversión desmedida de recursos humanos y computacionales. En la industria, diversos softwares abordan esta tarea, pero en este documento nos centraremos en Unreal Engine, que representa una competencia directa para la herramienta propuesta.

Unreal Engine se destaca no solo por su suite nativa de herramientas procedurales, sino también por su capacidad para integrar plugins externos, ofreciendo así una amplia gama de opciones a los usuarios. Estas herramientas permiten a diseñadores y artistas poblar vastos paisajes de manera eficiente y con gran detalle estético, ajustándose a diversas reglas y parámetros que garantizan una integración natural con el entorno virtual. Entre las herramientas más destacadas de Unreal Engine para la creación procedural, se pueden encontrar:

2.3.1. Herramientas nativas de Unreal Engine

Grass Tool
https://dev.epicgames.com/documentation/en-us/unreal-engine/grass-quick-start-in-unreal-engine
Esta herramienta permite la colocación automática de hierba y pequeñas plantas sobre la superficie del terreno en función de las texturas definidas en el material del escenario.

- **Pros:** Fácil integración con los materiales del terreno. Permite cubrir grandes áreas rápidamente. Su distribución se hace pintando con una herramienta "pincel" 3D.
- **Contras**: Control y variedad de plantas limitados. Parámetros básicos
- **Complejidad:** Baja. Su uso es bastante intuitivo y no requiere configuraciones complicadas más allá de la asignación inicial en el material del terreno.

Procedural Foliage Tool
https://dev.epicgames.com/documentation/en-us/unreal-engine/procedural-foliage-tool-in-unreal-engine
Esta herramienta permite la distribución de árboles, arbustos y plantas en grandes superficies. Utiliza un sistema basado en simulaciones para gestionar la posición, tamaño y estado de cada planta, a través de parámetros que definen la competencia por la luz, el espacio o la vida. De esta manera se crean configuraciones de ecosistemas que permiten a los usuarios obtener entornos realistas en cuanto a la distribución de la flora de un escenario.

- **Pros**: Permite crear ecosistemas dinámicos y realistas con interacciones complejas entre especies de plantas.
- **Contras**: Requiere de un gran número de recursos computacionales durante las simulaciones. Su distribución está delimitada a un área definida por un *spawner*, una caja 3D.

- **Complejidad**: Media. La creación de ecosistemas realistas implica dedicar tiempo significativo para configurar múltiples parámetros. Además, las simulaciones necesarias para observar los efectos de estos ajustes también requieren tiempo, lo que prolonga el proceso hasta alcanzar el resultado final deseado.

Foliage System Tools
https://dev.epicgames.com/documentation/en-us/unreal-engine/foliage-mode-in-unreal-engine
Conjunto de herramientas para la colocación manual y semiautomática de vegetación y elementos inorgánicos. Permite un control detallado sobre la colocación de cada objeto, incluyendo árboles, arbustos, y otros elementos como rocas o construcciones artificiales.

- **Pros**: Gran flexibilidad y precisión en la colocación de objetos, es ideal para detalles finales y ajustes específicos. Es compatible con las herramientas anteriores, permitiendo la edición y refinamiento de los sistemas procedurales de dichas herramientas.
- **Contras**: Puede ser muy laborioso para grandes áreas sin la ayuda de automatización adicional.
- **Complejidad**: Media-alta, dependiendo del nivel de detalle y la precisión requerida en el proyecto.

Procedural Content Generation (PCG)
https://dev.epicgames.com/documentation/es-es/unreal-engine/procedural-content-generation-overview
Conjunto de herramientas para la creación de contenidos y herramientas procedimentales propias dentro de Unreal Engine. Permite generar todo tipo de elementos, desde terrenos, vegetación, ciudades, texturas e incluso niveles enteros, de manera dinámica y en tiempo real.

- **Pros**: Puede generar una amplia variedad de contenido diferente. Permite optimizar muy eficientemente el espacio de almacenamiento y el rendimiento de las escenas. Tiene una gran escalabilidad permitiendo la expansión del contenido y los escenarios de manera ilimitada.
- **Contras**: Requiere un alto nivel de habilidad en programación y comprensión de algoritmos complejos, lo cual puede ser un obstáculo para equipos pequeños o poco experimentados. Las simulaciones y generaciones complejas pueden consumir recursos significativos del sistema, potencialmente afectando al rendimiento, especialmente en dispositivos con capacidades limitadas.
- **Complejidad**: Alta. La correcta implementación y optimización de PCG necesita de habilidades y conocimientos avanzados en Unreal Engine y programación.

2.3.2.Plugins externos

Hierarchical Scatter Tool
https://www.unrealengine.com/marketplace/en-US/product/hierarchical-scatter-tool
Plugin desarrollado para Unreal Engine que permite dispersar mallas estáticas aleatoriamente dentro de un área específica. Posee controles que permiten un buen grado de control artístico para conseguir un *scatter* realista de rocas, follaje, accesorios, etc. en poco tiempo.

- **Pros**: Esta herramienta ofrece una interfaz intuitiva y clara, mejor que las opciones nativas de Unreal Engine. Una característica destacada es su capacidad para mover los *scatters* a través del terreno, ajustándose automáticamente a las irregularidades y variaciones de la superficie.
- **Contras**: Al ser un plugin externo, implica un costo adicional y existe el riesgo de perder soporte técnico y actualizaciones en el futuro. Está limitado a sus funcionalidades específicas, no admite geometría animada y no está optimizado para grandes escenarios.
- **Complejidad**: Baja. Uso sencillo e intuitivo con la posibilidad de consultar un video explicativo del uso de la herramienta lo que facilita un rápido aprendizaje de la misma.

Procedural Instance Spawner
https://www.unrealengine.com/marketplace/en-US/product/procedural-instance-spawner
Plugin desarrollado para Unreal Engine, está diseñado para crear grandes entornos 3D mediante la generación procedimental. Tiene un gran número de controles que permiten optimizar y obtener resultados artísticos de gran complejidad.

- **Pros**: La interfaz ofrece numerosas opciones para ajustes detallados y una serie de configuraciones preestablecidas que simplifican el proceso. El software está optimizado para realizar cálculos en segundo plano, facilitando la gestión de entornos extensos. Incluye una función de comprobación de proximidad avanzada para evitar intersecciones no deseadas.
- **Contras**: Actualmente no soporta geometría animada y, siendo un plugin externo, conlleva costes adicionales y el riesgo de perder soporte técnico y actualizaciones futuras.
- **Complejidad**: Media. Aunque la interfaz es rica en opciones, esto puede complicar el aprendizaje de su uso completo. Además, ciertas operaciones requieren modificaciones en las configuraciones nativas de Unreal Engine, aumentando la complejidad.

2.3.3.Tabla comparativa

Herramienta	Pros	Contras	Complejidad	Versatilidad	Coste	Nativa
Grass Tool	Integración fácil con materiales de terrenos. Rápida cobertura de áreas grandes.	Limitada en control y variedad de vegetación. Parámetros básicos.	Baja	Baja	Gratuito	Sí
Procedural Foliage Tool	Ecosistemas dinámicos y realistas.	Alta demanda de recursos. Limitada a un área definida por un *spawner*.	Media	Media	Gratuito	Sí
Foliage System Tools	Alta flexibilidad y precisión. Indicada para detalles finales.	Laborioso para grandes áreas sin automatización.	Media-Alta	Media	Gratuito	Sí
Procedural Content Generation (PCG)	Extrema versatilidad en generación de contenido.	Requiere conocimientos avanzados en Unreal Engine y de programación.	Alta	Alta	Gratuito	Sí
Hierarchical Scatter Tool	Interfaz intuitiva. Adapta *scatters* automáticamente al terreno.	Costo adicional. Riesgo de perder soporte. No para grandes escenarios.	Baja	Media	Bajo	No
Procedural Instance Spawner	Numerosas opciones de ajuste. Optimizado para entornos extensos.	No admite geometría animada. Costo adicional. Riesgo de perder soporte.	Media	Alta	Alto	No

Tabla 7: Comparativa competencia

2.4. Análisis DAFO

FORTALEZAS	DEBILIDADES
• Uso de mapas procedurales para la generación de *scatters*. • Gran número de opciones de control y personalización. • Gran versatilidad y escalabilidad por su desarrollo en Houdini Engine. • Amplia experiencia con Houdini.	• Primera herramienta desarrollada para el sector. Necesidad de clientela. • Falta de conocimientos de Unreal Engine. • Dificultad inicial para implementar todas las opciones. Necesario I+D. • Dependencia de Houdini Engine. Implica gasto adicional
OPORTUNIDADES	**AMENAZAS**
• Sector al alza con crecimiento constante. • Poca competencia directa en el nicho específico de mercado. • Mercado internacional.	• Competencia fuerte de herramientas nativas y plugins externos. • Competencia consolidada. • Cambios en el software nativo. • Riesgo de piratería.

Tabla 8: Análisis DAFO

3. Propuesta

La propuesta consiste en el desarrollo de una herramienta diseñada para facilitar la creación de extensos escenarios naturales de forma procedural, ofreciendo un control técnico y artístico preciso durante todo el proceso. Desarrollada con Houdini y operable en Unreal Engine mediante Houdini Engine, el elemento diferenciador de esta herramienta respecto a otras opciones nativas y otros plugins externos es su capacidad para ajustar meticulosamente el *scatter* a través del uso de mapas de información, así como la posibilidad de reusar la configuración para crear nuevos escenarios gracias a la proceduralidad. Este enfoque proporciona una versatilidad y control sin igual en la configuración de entornos complejos y dinámicos dentro de Unreal Engine.

3.1. Definición de objetivos y especificaciones del producto

Se describen a continuación las características principales y secundarias de la herramienta. Los objetivos principales del TFG se centrarán en desarrollar las características principales de la herramienta, y, de ser posible, también se buscará implementar las funcionalidades correspondientes a los objetivos secundarios, ya sea en la fase actual o en desarrollos futuros.

3.1.1. Objetivos y características principales

- **Generación basada en mapas de información**: Su funcionalidad más característica y diferenciadora es generar *scatters* en función de diferentes mapas de información. Estos mapas de información sirven para definir, principalmente la densidad, pero también pueden usarse para definir la escala, rotación, el objeto a instanciar, etc. lo cual permite tener un control absoluto a través de los diferentes mapas.

- **Mapas de información versátiles**: Los mapas de información se pueden generar de manera manual mediante herramientas de pincel o de manera procedural, a través de mapas de ruido procedurales, análisis de altitud y mapas de oclusión, etc. Los mapas son combinables entre sí, añadiendo o restando información.

- **Control detallado del *scatter***: Una vez realizado el *scatter* de los diferentes elementos, la herramienta permite ajustar dicho *scatter* a través de las opciones de configuración, añadiendo más o menos densidad, modificando los valores de rotación, posición, escala, eligiendo los elementos a instanciar, así como su proporción, etc.

3.1.2.Objetivos y características secundarias

- **Interactividad con geometrías**: Capacidad para integrar y reconocer otros elementos 3D, como curvas y figuras geométricas. Dichos elementos se utilizan para refinar el *scatter* delimitando, añadiendo o eliminando zonas de densidad.

- **Resolución de intersecciones**: Detecta y corrige las intersecciones entre los objetos instanciados para evitar solapamientos no deseados.

- **Edición post-*scatter***: Permite ajustes finos post-*scatter* de manera manual, permitiendo al usuario modificar los elementos instanciados, pudiendo eliminar o cambiar la posición, escala o rotación de instancias concretas.

- **Soporte para elementos animados:** La herramienta incluye funcionalidades para instanciar tanto elementos estáticos como animados, lo que enriquece los escenarios con un realismo adicional mediante movimientos sutiles, como el balanceo de hojas y ramas de los árboles.

3.1.3.Especificaciones técnicas

- **Desarrollo en Houdini:** Reconocido por sus potentes herramientas procedurales, Houdini permite la creación y manipulación compleja de datos y geometrías. La interoperabilidad con Unreal a través de Houdini Engine facilita la integración de procesos complejos, y su eficiencia en la simulación y manejo de grandes volúmenes de datos asegura un rendimiento óptimo para proyectos de gran escala.

- **Implementación en Unreal Engine:** La implementación de la herramienta en Unreal Engine aprovecha su avanzado sistema de renderizado y amplia aceptación en la industria, proporcionando un entorno óptimo para la visualización y manipulación en tiempo real de entornos complejos. Su robusta plataforma facilita las pruebas directas y la interactividad, permitiendo a los desarrolladores optimizar los escenarios de manera eficiente y asegurando compatibilidad y soporte constante para futuras actualizaciones.

- **Compatibilidad de formatos**: La herramienta ofrece soporte para formatos tradicionales como Alembic, OBJ y FBX, así como para el moderno formato USD y el uso de *blueprints* de Unreal Engine. Esto asegura una amplia compatibilidad y flexibilidad, facilitando la integración en diversos flujos de trabajo y proyectos.

- **Optimización del rendimiento**: Diseñada para ser eficiente en el uso de recursos computacionales, facilitando la creación de grandes entornos sin comprometer el rendimiento general de Unreal Engine.

3.2. Modelo de negocio

El modelo de negocio se basa en la venta de licencias de la herramienta a través de internet, utilizando Gumroad como la plataforma principal de distribución, la cual ofrece un acceso fácil y directo al mercado global. Se ofrecerán dos tipos de licencias: una "Licencia Indie" para usuarios individuales y pequeñas empresas, y una "Licencia *Commercial*" para grandes empresas. Esta estructura asegura que tanto desarrolladores independientes como empresas puedan acceder a la herramienta según sus necesidades.

Tipos de Licencias a la venta:

- **Licencia Indie:** Dirigida a usuarios individuales o pequeñas empresas, con un costo de 250€ por licencia. Esta opción facilita el acceso a la herramienta para desarrolladores independientes y pequeños estudios.
- **Licencia *Commercial*:** Diseñada para empresas, con un costo de 600€ por licencia y ordenador donde se use. Este modelo permite a las empresas utilizar la herramienta de manera más extensiva y profesional.

Gastos Asociados:

- **Desarrollo inicial:** El costo total estimado del desarrollo inicial asciende a 18.900€.
- **Actualizaciones anuales:** La herramienta requiere actualizaciones periódicas para asegurar su compatibilidad con las nuevas versiones de Unreal Engine y Houdini, además de la incorporación de nuevas funcionalidades y la corrección de errores. Se estima un gasto anual de 2.400€.
- **Marketing:** Se suscribirá a LinkedIn Premium a un costo de 25€ mensuales, lo que representa un gasto anual de 300€.
- **Plataforma de venta:** Gumroad aplica una comisión fija del 10% sobre cada venta.

Beneficios Económicos Esperados:

El éxito económico del proyecto dependerá de la aceptación y adopción de la herramienta por parte de la comunidad de desarrolladores de videojuegos y efectos visuales. Una vez alcanzada una base de usuarios significativa, la herramienta comenzará a generar beneficios de manera pasiva debido a sus bajos costos de mantenimiento. Además, el plan de negocio no se enfoca únicamente en esta herramienta, sino que se contempla el desarrollo continuo de otras herramientas, muchas de las cuales serán asociadas y complementarias a esta. Esta estrategia permitirá diversificar la oferta y maximizar los beneficios a largo plazo.

3.3. Estrategia de marketing

La estrategia de marketing para la herramienta se centrará en una fuerte presencia en redes sociales y campañas de promoción dirigida al público profesional.

Promoción de la herramienta:

- **Redes Sociales:** LinkedIn será la plataforma principal debido a su capacidad para llegar a profesionales de la industria de desarrollo de videojuegos y efectos visuales. Se crearán publicaciones regulares que muestren las capacidades de la herramienta, casos de uso y testimonios de usuarios.
- **Demostraciones en Video:** Se crearán tutoriales y demostraciones en video que se compartirán en YouTube, Instagram y LinkedIn para mostrar la funcionalidad y las ventajas de la herramienta.

Estrategia de Venta:

- **Precios Competitivos:** Se establecerán precios accesibles para desarrolladores independientes y competitivos para empresas.
- **Descuentos y Ofertas Especiales:** Se ofrecerán descuentos por lanzamiento y promociones durante eventos relevantes para impulsar las ventas.

4. Diseño

En este capítulo, se describen los aspectos fundamentales del diseño de la herramienta desarrollada para la creación procedural de elementos en grandes superficies. Esta herramienta se enfoca principalmente en la generación de vegetación, pero su capacidad se extiende a otros tipos de elementos. Utilizando Houdini para el desarrollo y Houdini Engine para su ejecución en Unreal Engine, a continuación, se detalla la arquitectura general del sistema, la arquitectura de la información, el diseño gráfico e interfaces, y los lenguajes de programación y APIs utilizados.

4.1. Arquitectura general de la aplicación/sistema/servicio

Este apartado proporciona una descripción detallada de los componentes que integran el sistema desarrollado y la forma en que estos interactúan en la ejecución de la herramienta.

El sistema está compuesto por tres componentes principales que trabajan en conjunto para generar y visualizar elementos procedurales:

- **Houdini**: Houdini es un paquete 3D completo desarrollado por SideFX. Una de sus principales características es su sistema basado en nodos, que permite la creación de modelos y escenas complejas de manera procedural. En este proyecto, Houdini se utiliza para diseñar y desarrollar los algoritmos necesarios que definen los *scatters* de los elementos a instanciar, determinando su posición, rotación, escala y tipo de instancia a utilizar.

- **Houdini Engine:** Actúa como middleware, permitiendo que los algoritmos y nodos desarrollados en Houdini sean utilizados y ejecutados dentro de Unreal Engine. Este componente traduce las operaciones de Houdini en tareas que Unreal puede entender y procesar. Recibe los *inputs* de Unreal, ejecuta la herramienta con la configuración asignada y devuelve los resultados a Unreal para su visualización.

- **Unreal Engine:** Unreal Engine es el motor de videojuegos donde se ejecutan y visualizan los resultados generados por la herramienta. Este programa también es responsable de suministrar los *inputs* necesarios a la herramienta. Los *inputs* incluyen la “Geometría Base” sobre la cual se realizarán los *scatters*, los “Elementos a instanciar” que serán distribuidos procedimentalmente, y “Otros elementos” que se utilizarán para crear y ajustar las máscaras que definirán la densidad y distribución de los elementos instanciados.

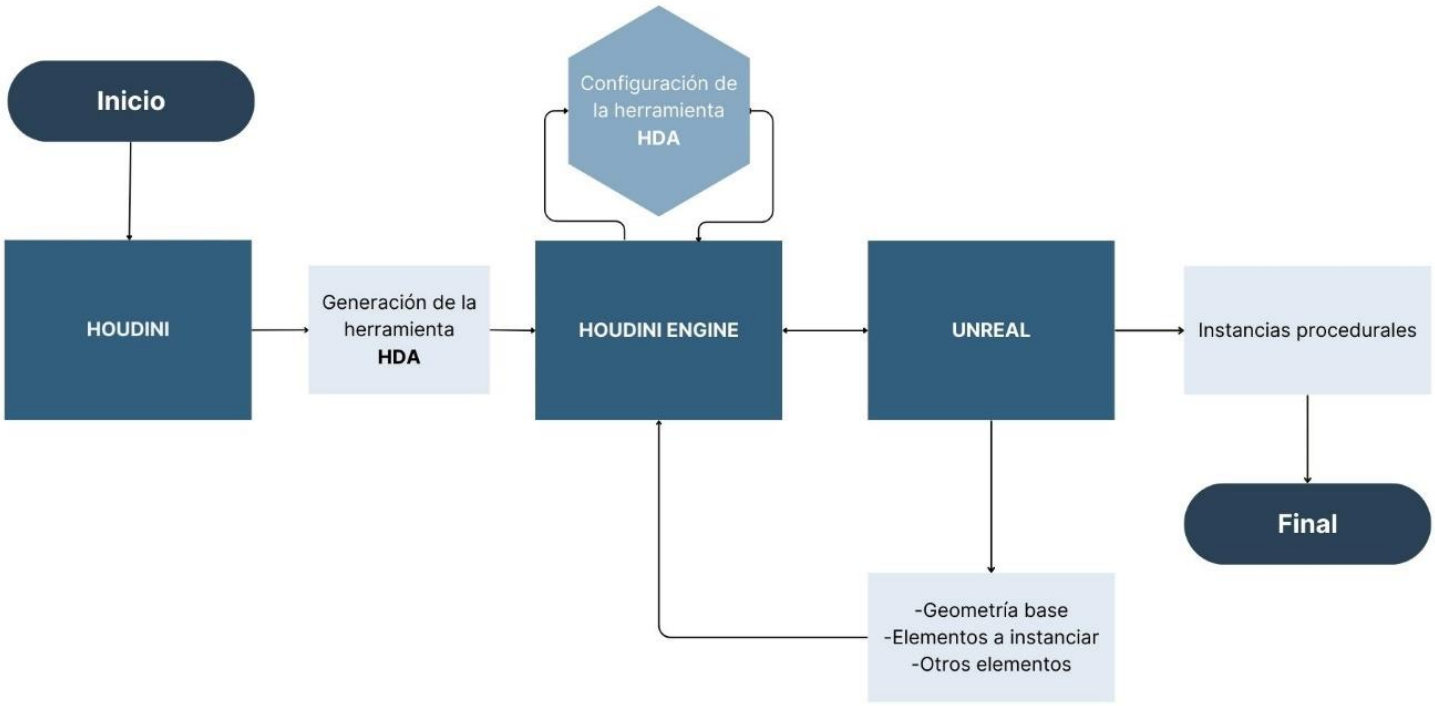


Figura 3: Diagrama general de funcionamiento de la herramienta

4.2. Arquitectura de la información y diagramas de navegación

Este apartado describe detalladamente los elementos principales que componen la aplicación mediante diagramas, explicando cómo se organizan y cómo interactúan.

4.2.1. Creación y configuración de máscaras.

1. **Inicio: Configuración de la herramienta**
   - **UNREAL INPUT 1: Geometría con "vertex color" rojo**
     - El proceso comienza con la configuración de la geometría base en Unreal Engine, donde se aplica un color rojo a los vértices de las áreas que se desean utilizar para el *scatter*.
2. **Generación de Máscaras Extra**
   - Si se requiere, se pueden generar máscaras adicionales basadas en diferentes parámetros:
     - **NOISE:** Añade una máscara basada en ruido.
     - **DIRECTION:** Añade una máscara basada en la dirección.
     - **NORMAL:** Añade una máscara basada en las normales.
     - **AMBIENT OCCLUSION:** Añade una máscara basada en la oclusión ambiental.
     - **HEIGHT:** Añade una máscara basada en la altura.
     - **POINTS:** Añade una máscara basada en puntos específicos.
     - **GEOMETRY:** Añade una máscara basada en la geometría.
3. **Inversión de Máscara**
   - **Inversión de máscara (Sí/No):** Opción para invertir la máscara generada.
4. **Remapeo de Máscara**
   - **Remapear máscara (Sí/No):** Opción para remapear los valores de la máscara para ajustarlos según sea necesario.
5. **Suavización de Máscara**
   - **Suavizar máscara (Sí/No):** Opción para suavizar la máscara para obtener transiciones más suaves.
6. **Mezcla de Máscaras**
   - **Mezcla de máscaras (Sí/No):** Opción para combinar varias máscaras generadas. Si se elige combinar máscaras, se presentan diferentes opciones para la operación de mezcla:
     - **ADD:** Suma las máscaras.
     - **SUBTRACT:** Resta una máscara de otra.
     - **MULTIPLY:** Multiplica las máscaras.
     - **DIVIDE:** Divide una máscara entre otra.
     - **MAXIMUM:** Toma el valor máximo de las máscaras.
     - **MINIMUM:** Toma el valor mínimo de las máscaras.
     - **BLEND:** Mezcla las máscaras de acuerdo a un factor de mezcla.
7. **Output: Máscara Final**
   - El resultado final es la máscara compuesta, que se utiliza para definir la densidad y distribución de los elementos instanciados en la escena.

El siguiente diagrama de flujo muestra las diferentes opciones de la herramienta descritas anteriormente para añadir y modificar nuevas máscaras y utilizarlas como información de densidad en *el scatter* final.

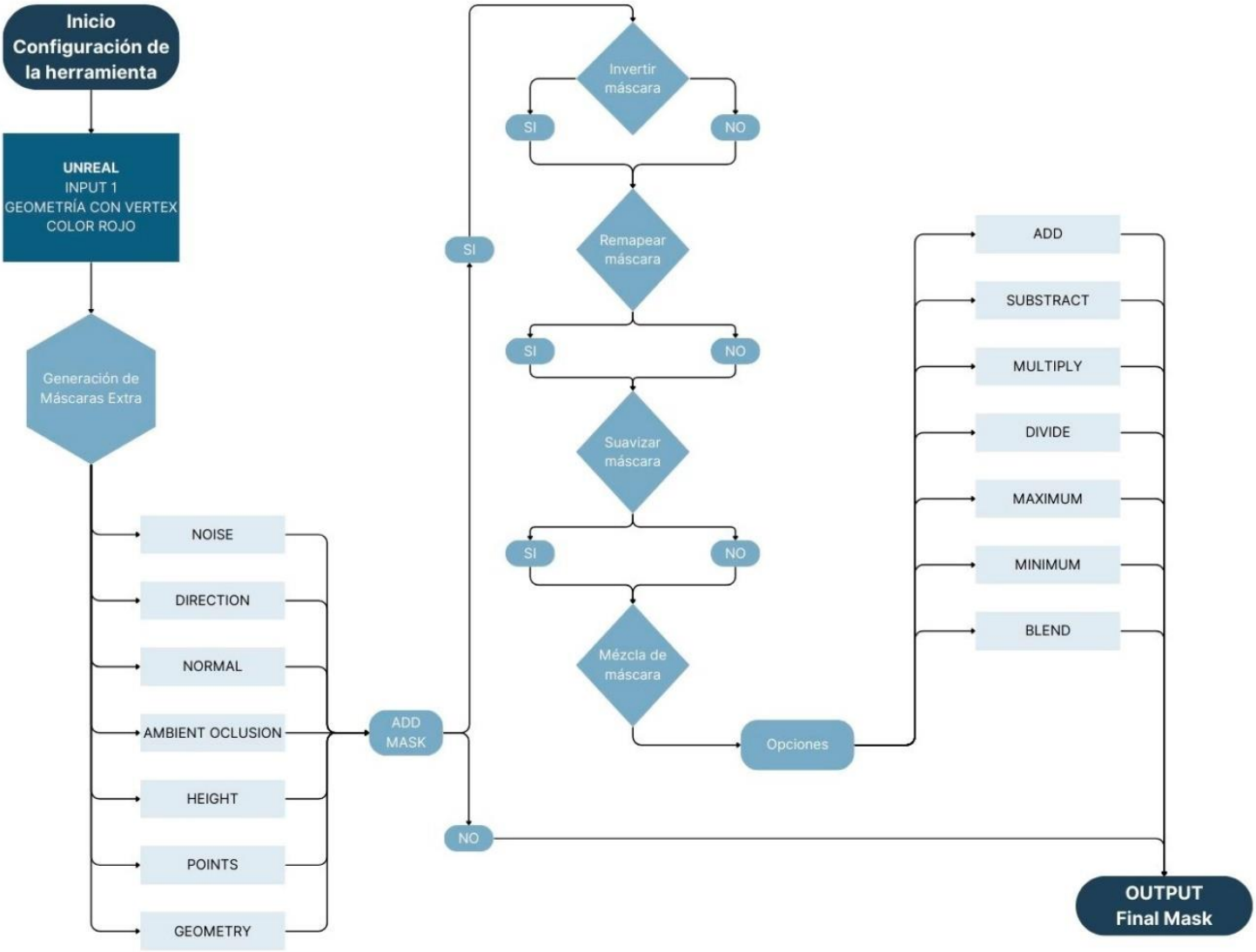


Figura 4: Diagrama del funcionamiento de la herramienta. Sección máscaras

4.2.2.Creación y configuración del *scatter*.

1. **Inicio: Configuración de la herramienta**
   - **HOUDINI ENGINE INPUT 1: *Final Mask***
     - El proceso comienza con la máscara final generada en Houdini Engine, la cual se utilizará para definir la densidad y distribución de los puntos del *scatter*.

2. **Generación de los Puntos de *Scatter* (DENSITY)**
   - ***Force total count*:** Opción para forzar un número total de puntos.
   - ***Density scale*:** Ajusta la escala de densidad de los puntos.
   - ***Seed*:** Genera un valor de semilla para el scatter.
   - ***Relax Points*:** Relaja los puntos del *scatter* para una distribución más uniforme.

3. **Generación de las Orientaciones (ORIENTATION)**
   - ***Orient to Normal*:** Orienta los puntos según las normales.
   - ***Orient to World*:** Orienta los puntos según las coordenadas del mundo.
   - ***General Rotation*:** Ajusta la rotación general en los ejes X, Y y Z.
     - ***Rotation X***
     - ***Rotation Y***
     - ***Rotation Z***
   - ***Random Rotation*:** Ajusta la rotación aleatoria dentro de un rango mínimo y máximo en los ejes X, Y y Z.
     - ***Rotation X min/máx.***
     - ***Rotation Y min/máx.***
     - ***Rotation Z min/máx.***
   - ***Seed***: Genera un valor de semilla para la rotación.

4. **Generación de la Escala (SCALE)**
   - ***General Scale*:** Ajusta la escala general de los puntos.
   - ***Random Scale*:** Ajusta la escala aleatoria dentro de un rango mínimo y máximo.
     - ***Scale min/máx.***
   - ***Seed*:** Genera un valor de semilla para la escala.

5. **Generación de Offset de Posición (POSITION OFFSET)**
   - ***General offset*:** Ajusta el desplazamiento general en los ejes X, Y y Z.
     - ***Offset X***
     - ***Offset Y***
     - ***Offset Z***
   - ***Random offset*:** Ajusta el desplazamiento aleatorio dentro de un rango mínimo y máximo en los ejes X, Y y Z.
     - ***Offset X min/máx.***
     - ***Offset Y min/máx.***
     - ***Offset Z min/máx.***
   - ***Seed*:** Genera un valor de semilla para el desplazamiento.

6. **Copia de *Assets* a Puntos**
   - **UNREAL INPUT 2*: Assets***
     - Se configuran los *assets* a instanciar en los puntos del scatter generado.
   - **Número de Assets:** Determina el número de elementos a copiar.
   - ***Asset Name*:** Especifica el nombre del elemento.
   - ***Asset Weight*:** Asigna un peso al elemento para la distribución.
   - ***Seed*:** Genera un valor de semilla para la distribución de elementos.
7. **Output: *Scatter Final***
   - El resultado final es el *scatter* compuesto, que se utiliza para instanciar los elementos en la escena según la configuración especificada.

El siguiente diagrama de flujo muestra el proceso detallado de configuración y generación del *scatter* en la herramienta. Al final de este proceso se obtiene el *scatter* final de los elementos.

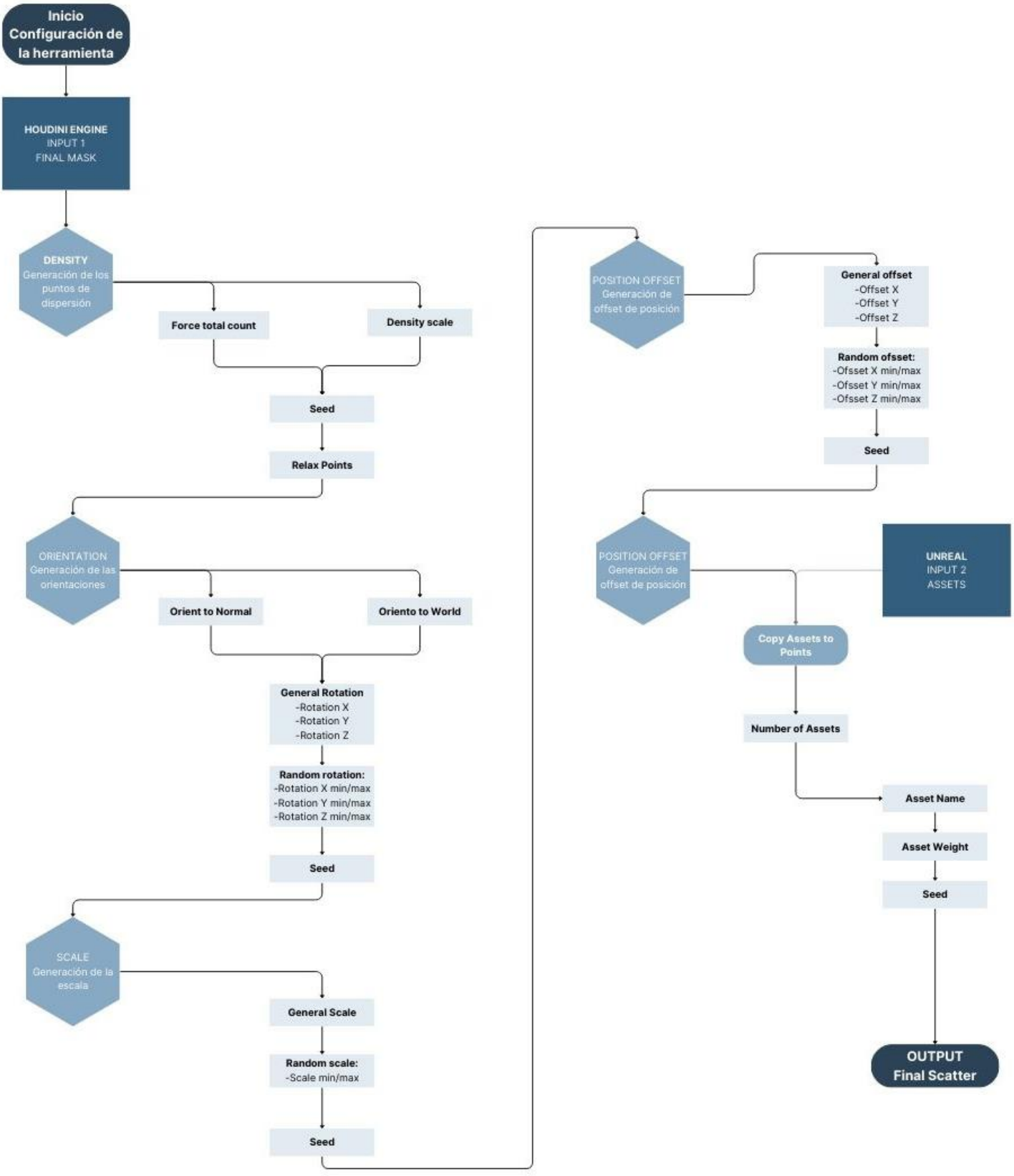


Figura 5: Diagrama del funcionamiento de la herramienta. Sección *scatter*.

4.3. Diseño gráfico e interfaces

Aunque el diseño gráfico está limitado a las opciones de interfaz disponibles en Unreal Engine y Houdini, la interfaz de la herramienta ha sido estructurada en diferentes paneles y elementos gráficos que siguen un orden lógico de configuración. Esta organización permite a los usuarios navegar y utilizar la herramienta de forma eficiente, asegurando una experiencia de usuario optimizada.

1. **Estructura general:**
   La estructura general presenta una serie de paneles dentro de Unreal Engine, donde se configuran y ajustan los parámetros necesarios para la generación procedural.

   Los paneles están organizados de manera jerárquica y lógica, agrupando las opciones relacionadas bajo tres categorías principales "*Mask Options*", "*Scatter Options*" y "*Assets Options*".

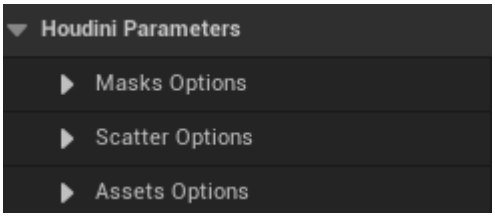


Figura 6: Interfaz general de la herramienta

2. **Elementos gráficos de la interfaz:**
   A pesar de las limitaciones, la interfaz contiene diversos elementos gráficos que facilitan la configuración de todas las opciones de la herramienta de manera fácil e intuitiva.
   - ***Checkboxes* (Casillas de Verificación):** A lo largo de toda la herramienta se encuentran diferentes *checkboxes* para activar o desactivar diferentes funcionalidades.

Figura 7: Elemento interfaz - Checkboxes

   - **Menús desplegables:** Permiten alternar entre diferentes opciones de la herramienta.

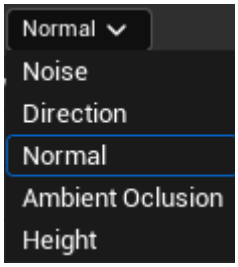


Figura 8: Elemento interfaz - Menú desplegable

- **Curvas de Remapeo:** Gráfica interactiva que permite remapear los valores de las máscaras.

Figura 9: Elemento interfaz - Curva de remapeo

- ***Sliders* (Deslizadores):** Permite seleccionar el valor deseado deslizando el control a través de una ventana gráfica delimitada por un rango especifico.

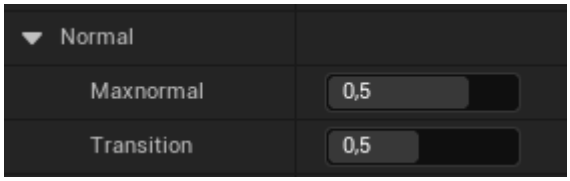


Figura 10: Elemento interfaz - Deslizadores

- **Campos de Entrada Numéricos:** Ingresa directamente valores numéricos para controlar diferentes opciones.

Figura 11: Elemento interfaz - Campos de entrada numéricos

- **Botones:** Proporcionan diferentes funciones, como agregar o eliminar elementos, resetear, etc.

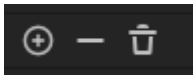


Figura 12: Elemento interfaz - Botones

3. Elementos Gráficos del viewport

A través de los controles de la herramienta se puede configurar diferentes opciones de visualización.

- **Display Ground:** Muestra en viewport la geometría base sobre la que se está realizando el *scatter*.

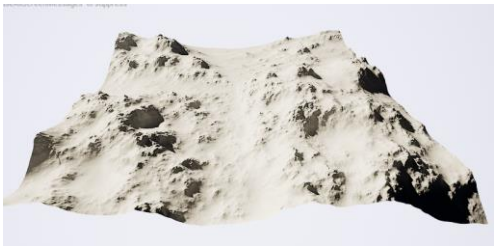


Figura 13: Visualización de la geometría base

- ***Display Final/This Mask*:** Esta opción permite visualizar en el viewport de Unreal Engine las áreas activas de la máscara, facilitando la creación y ajuste de las diferentes máscaras de manera visual.

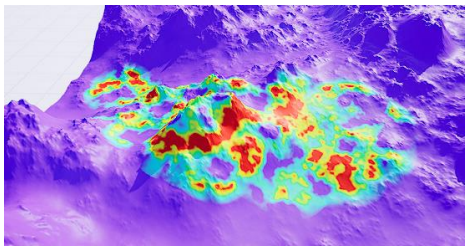


Figura 14: Visualización de máscara en viewport

- ***Display Instances*:** Opción por defecto, sirve para visualizar los elementos instanciados en nuestro *scatter*.

Figura 15: Visualización de las instancias

- ***Display Scatter*:** Si no se ha asignado un elemento específico para instanciar al realizar un *scatter*, por defecto se visualizará una caja en cada punto del *scatter* con los atributos de transformación aplicados, como escala, rotación y movimiento.

Figura 16: Visualización del scatter

4.3.1. Usabilidad /UX

La usabilidad de la herramienta se centra en proporcionar una experiencia intuitiva y eficiente para los usuarios, permitiéndoles configurar y ajustar los parámetros del *scatter* de manera rápida y precisa, sin necesidad de tener conocimientos avanzados en el programa. En este sentido, incluso un usuario con muy poca experiencia puede lograr *scatters* avanzados con facilidad.

La interfaz está diseñada de manera lógica, con paneles y controles organizados jerárquicamente para facilitar la navegación y configuración de parámetros. Las opciones de "*Display Final Mask / This Mask*" proporcionan un *feedback* visual en tiempo real, permitiendo a los usuarios ver directamente en el visor de Unreal Engine dónde aparecerán los elementos dispersados, lo que mejora significativamente la precisión de los ajustes.

Los usuarios pueden personalizar una amplia gama de parámetros, como densidad, escala, orientación y desplazamiento de los puntos del *scatter*, utilizando controles claros y accesibles. Esta capacidad de personalización permite ajustar cada aspecto del *scatter* para satisfacer las necesidades específicas del proyecto.

Además, cualquier cambio realizado en los parámetros se refleja de inmediato en la vista de Unreal Engine, lo que permite a los usuarios ver el impacto de sus ajustes sin demora. Este *feedback* visual inmediato es crucial para iterar rápidamente y obtener resultados precisos, mejorando la eficiencia del flujo de trabajo y la experiencia general del usuario.

4.4. Lenguajes de programación y APIs utilizados

La elección de las herramientas utilizadas para el desarrollo de esta herramienta se fundamenta en dos razones principales. En primer lugar, Houdini y Unreal Engine son softwares líderes en sus respectivos campos, ampliamente utilizados en la industria del entretenimiento y la creación de videojuegos por su capacidad avanzada en la generación procedural y visualización en tiempo real. En segundo lugar, los conocimientos previos en el manejo de estas herramientas han sido determinantes. La experiencia previa en Houdini y HScript facilita y acelera el proceso de desarrollo al trabajar con herramientas ya conocidas.

Recursos tecnológicos utilizados

1. **Software:**
   - **Houdini SideFX 20.0.653:** Utilizado para el desarrollo principal de la herramienta, incluyendo la creación de sistemas procedurales y la configuración de los algoritmos necesarios para realizar el *scatter* de elementos.
   - **Unreal Engine 5.3.2:** Utilizado para ejecutar y visualizar los resultados de los algoritmos procedurales desarrollados en Houdini.
   - **Windows 11:** Sistema operativo utilizado para el desarrollo y ejecución de las herramientas.

2. **Lenguaje de programación:**
   - **HScript:** Principal lenguaje de programación utilizado en Houdini para la creación de scripts y herramientas procedurales. HScript es un lenguaje de *scripting* que se asemeja a Java en su sintaxis y estructura, lo que facilita su aprendizaje y uso para aquellos familiarizados con Java.

3. **Complementos:**
   - **Houdini Engine API:** *Middleware* que facilita la integración de Houdini en Unreal Engine, permitiendo la ejecución de la herramienta desarrollada en Houdini dentro de Unreal Engine.

4. **Hardware:**
   - **Estación de trabajo personal:**
     - **Procesador:** AMD Ryzen 9 5950X 16-Core Processor a 3.40 GHz.
     - **Memoria RAM:** 128 GB DDR4 3.200 Mhz.
     - **Tarjeta gráfica:** NVIDIA GeForce RTX 3070.

4.5. Desarrollo de la herramienta en Houdini SideFx

El desarrollo principal de la herramienta se lleva a cabo en Houdini de SideFX. Houdini es un software de animación y efectos 3D altamente reconocido por su sistema basado en nodos y su notable flexibilidad. Este enfoque, conocido como programación nodal, permite a los usuarios crear y personalizar sus propias herramientas de manera intuitiva y visual. Una de sus características más destacadas es el concepto de "*sandbox*", que permite a los usuarios crear y personalizar sus propias herramientas según sus necesidades específicas.

Además, Houdini utiliza "puntos" (*points*) que sirven para almacenar y manipular información. Estos puntos son fundamentales para una amplia variedad de operaciones, desde la modificación de los vértices de la geometría hasta la creación de efectos complejos como fuego y agua, entre otros. La capacidad de manejar estos puntos y editar su información es clave para realizar cualquier tarea que se necesite.

4.5.1. Árbol de nodos

El funcionamiento interno de la herramienta, consiste en la manipulación de la información de los puntos de una geometría a través de un árbol de nodos de la siguiente manera:

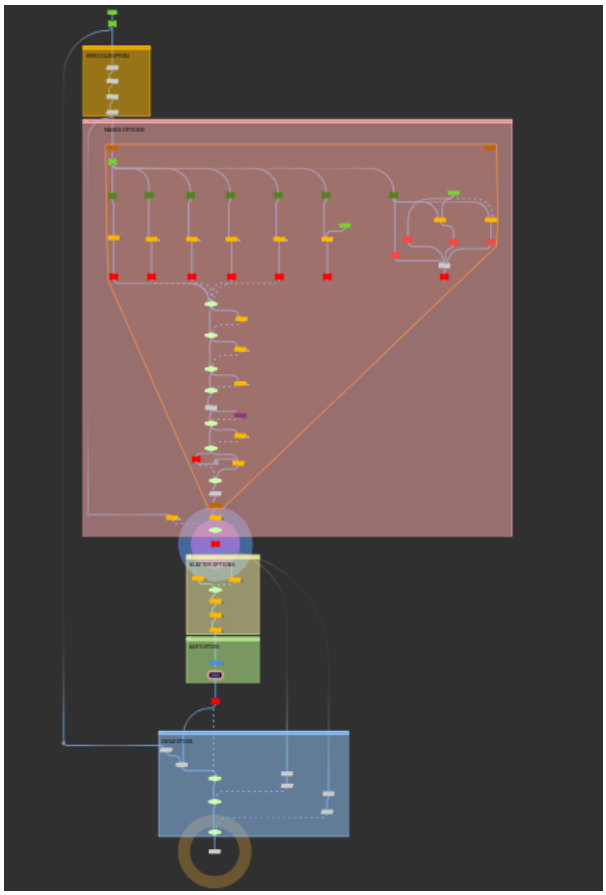


Figura 17: Houdini Setup – Árbol de nodos

Vertex Color Options

- La herramienta recibe una geometría con un atributo de color aplicado a sus vértices (*vertex color*) y transfiere esa información a los puntos para poder trabajar con ella.
- Se lee el atributo de color rojo (Cd.r) y se transforma en un atributo personalizado de tipo *float* llamado "*baseMask*". Este atributo almacena la información que define la densidad de puntos del *scatter*: a mayor intensidad de color rojo, mayor será la densidad de puntos; a menor intensidad de color rojo, menor será la densidad de puntos.

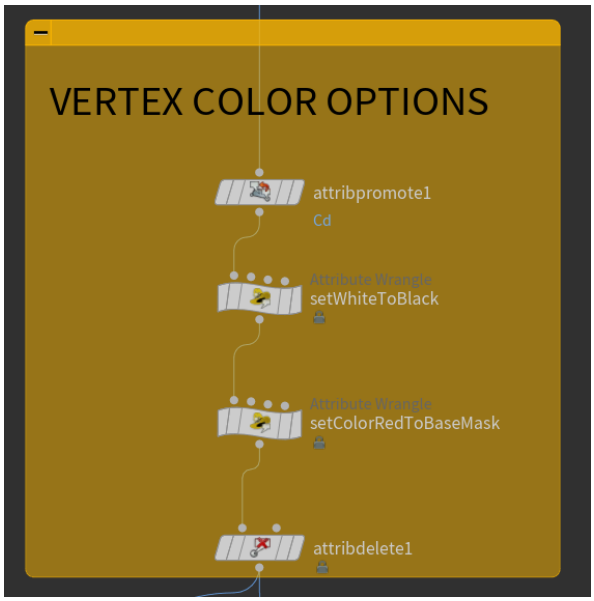


Figura 18: Árbol de nodos. Sección vertex color

Mask Options

- La información de los puntos entra en un bucle "*for each*" que se retroalimenta y añade una nueva iteración por cada una de las máscaras añadidas, con cada nueva máscara la información de *baseMask* se modifica y enriquece hasta obtener un nuevo atributo llamado "*finalMask*".

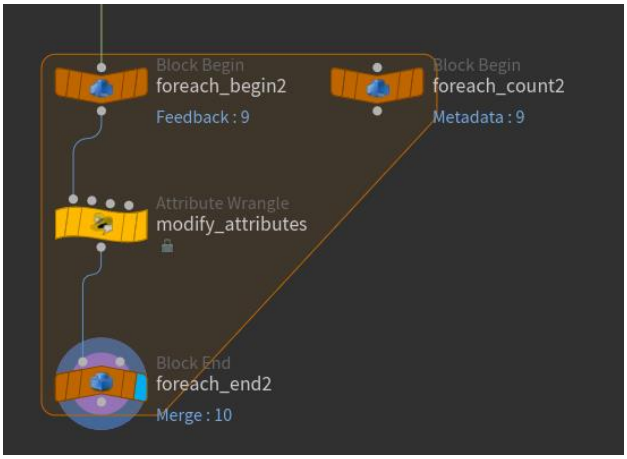


Figura 19: Árbol de nodos. Ejemplo *for each* loop

- En esta sección encontramos diferentes configuraciones para crear diferentes máscaras, en función de la máscara seleccionada el bucle pasará por una de las ramas gracias a un “*swich*” que desviará el bucle hasta la opción escogida.

Figura 20: Árbol de nodos. Sección tipos de máscaras

- Para finalizar, la información pasa por una serie de códigos que sirven para invertir, suavizar o remapear la información, así como la manera en cómo se agrega esta información a la máscara anterior.

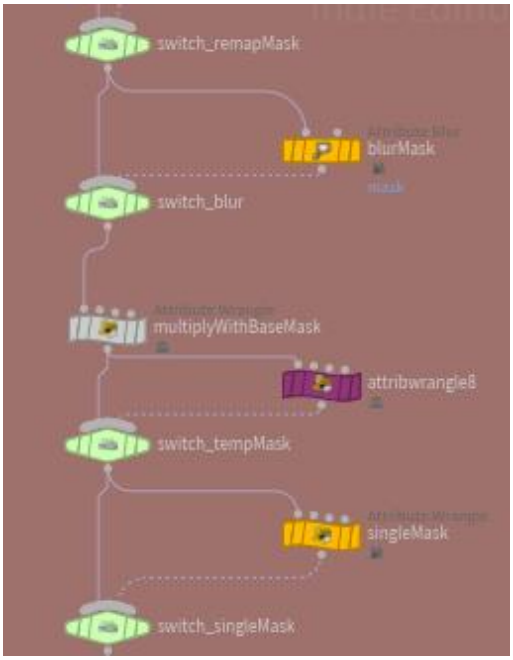


Figura 21: Árbol de nodos. Sección modificar máscara

Scatter Options

- En estos nodos se crean y editan los atributos que contienen la información de transformación de cada una de las instancias.
- Se utilizan nodos programados con código "HScript" para crear controles que permiten modificar la información de transformación de cada punto. Estas transformaciones se basan principalmente en operaciones matemáticas con vectores y quaterniones.

```
VEXpression
// Lee los parámetros de offset general
float offsetX = ch("offset_X");
float offsetY = ch("offset_Y");
float offsetZ = ch("offset_Z");

// Lee el parámetro de semilla para la aleatoriedad
int seed = chi("seed");

// Lee los parámetros de desplazamiento aleatorio (offset) mínimo y máximo para cada eje
float minOffsetX = ch("offset_min_X");
float maxOffsetX = ch("offset_max_X");
float minOffsetY = ch("offset_min_Y");
float maxOffsetY = ch("offset_max_Y");
float minOffsetZ = ch("offset_min_Z");
float maxOffsetZ = ch("offset_max_Z");

// Genera valores aleatorios dentro del rango especificado usando la semilla
float randomOffsetX = fit(rand(@ptnum + seed + 0.1), 0, 1, minOffsetX, maxOffsetX);
float randomOffsetY = fit(rand(@ptnum + seed + 0.2), 0, 1, minOffsetY, maxOffsetY);
float randomOffsetZ = fit(rand(@ptnum + seed + 0.3), 0, 1, minOffsetZ, maxOffsetZ);

// Calcula la posición final sumando el offset general y el aleatorio
vector finalOffset = set(offsetX + randomOffsetX, offsetY + randomOffsetY, offsetZ + randomOffsetZ);

// Aplica la posición final a cada punto
@P += finalOffset;

```

Figura 22: Código HScript. Offset position

- En este proceso ajusta y modifica la información de escala, posición y rotación de cada punto, determinando cómo se aplicarán estas transformaciones a las copias instanciada.

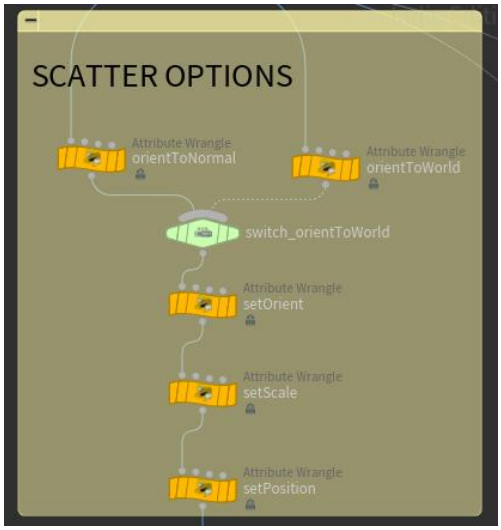


Figura 23: Árbol de nodos. Sección scatter

Assets Options

- En esta etapa, se especifica el *blueprint*/elemento que se copiará en cada punto. Hay opciones para agregar más elementos y ajustar su proporción relativa respecto a los demás.
- Finalmente, se definen las opciones de densidad, que determinan el número de copias basándose en el mapa de densidad creado anteriormente.

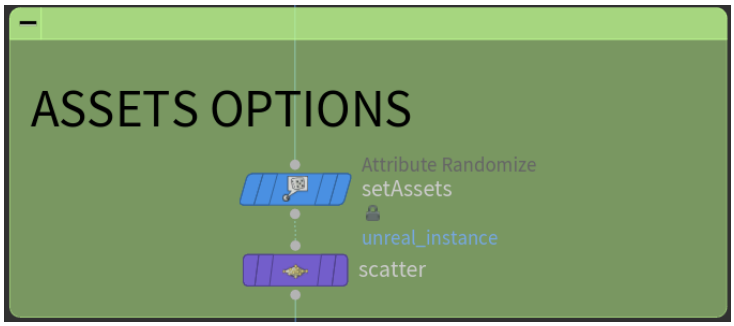


Figura 24: Árbol de nodos. Sección assets

Display Options

- Estos nodos sirven como métodos auxiliares que permiten visualizar la información y los diferentes elementos en el viewport mientras se trabaja.

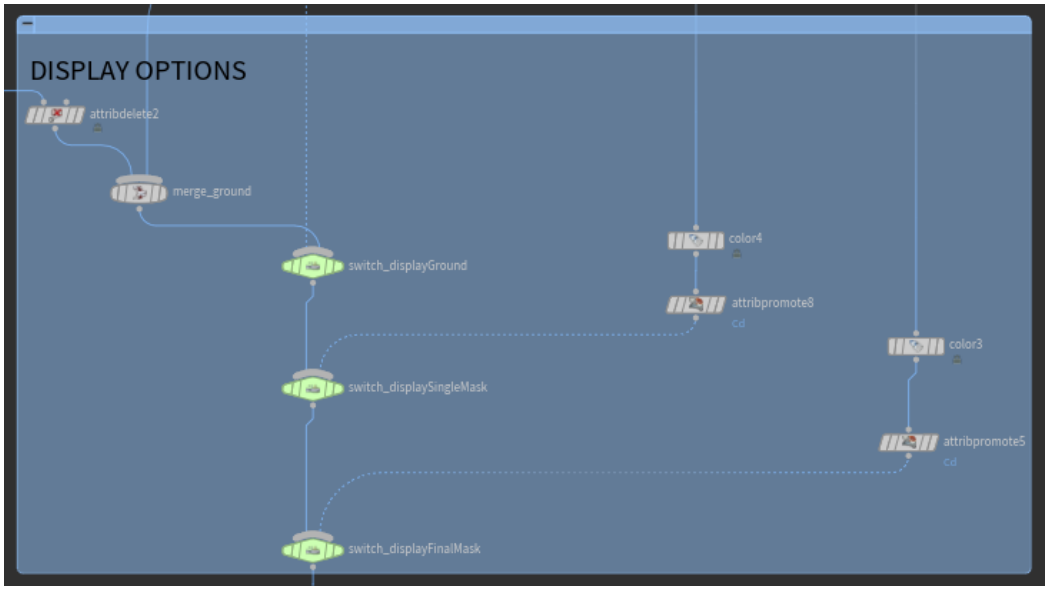


Figura 25: Árbol de nodos. Sección display options

4.5.2.Interfaz de usuario

Para facilitar el manejo de la herramienta, se crea una interfaz que permite a los usuarios interactuar con la herramienta de manera intuitiva, ajustando los diferentes parámetros de manera sencilla para conseguir el resultado deseado.

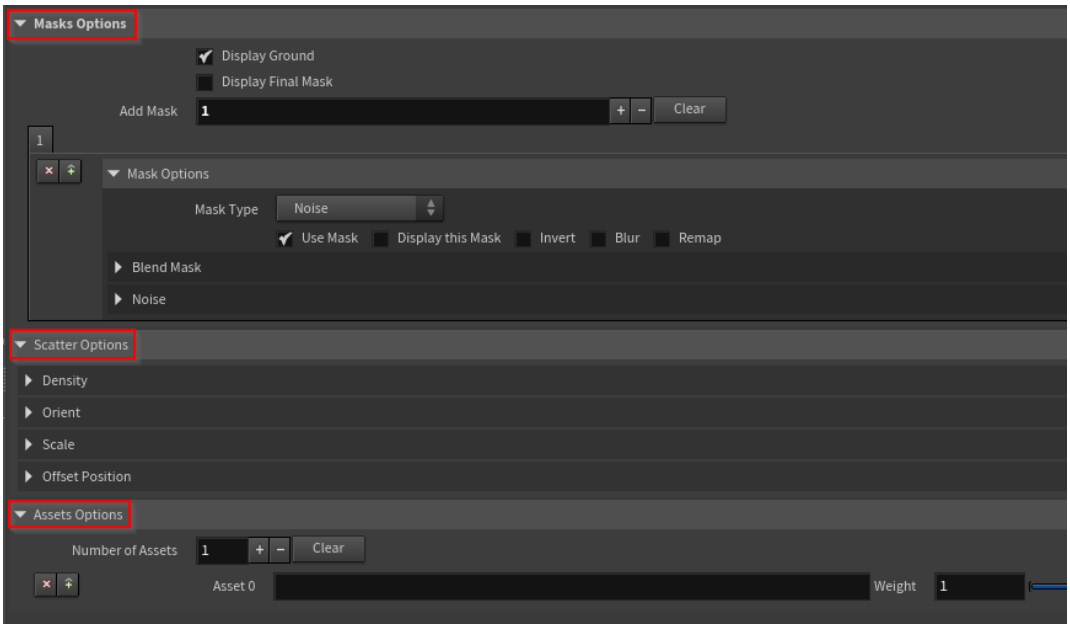


Figura 26: Houdini Setup – Interfaz de usuario

Para configurar esta interfaz se tienen que crear y añadir los parámetros deseados al nodo principal.

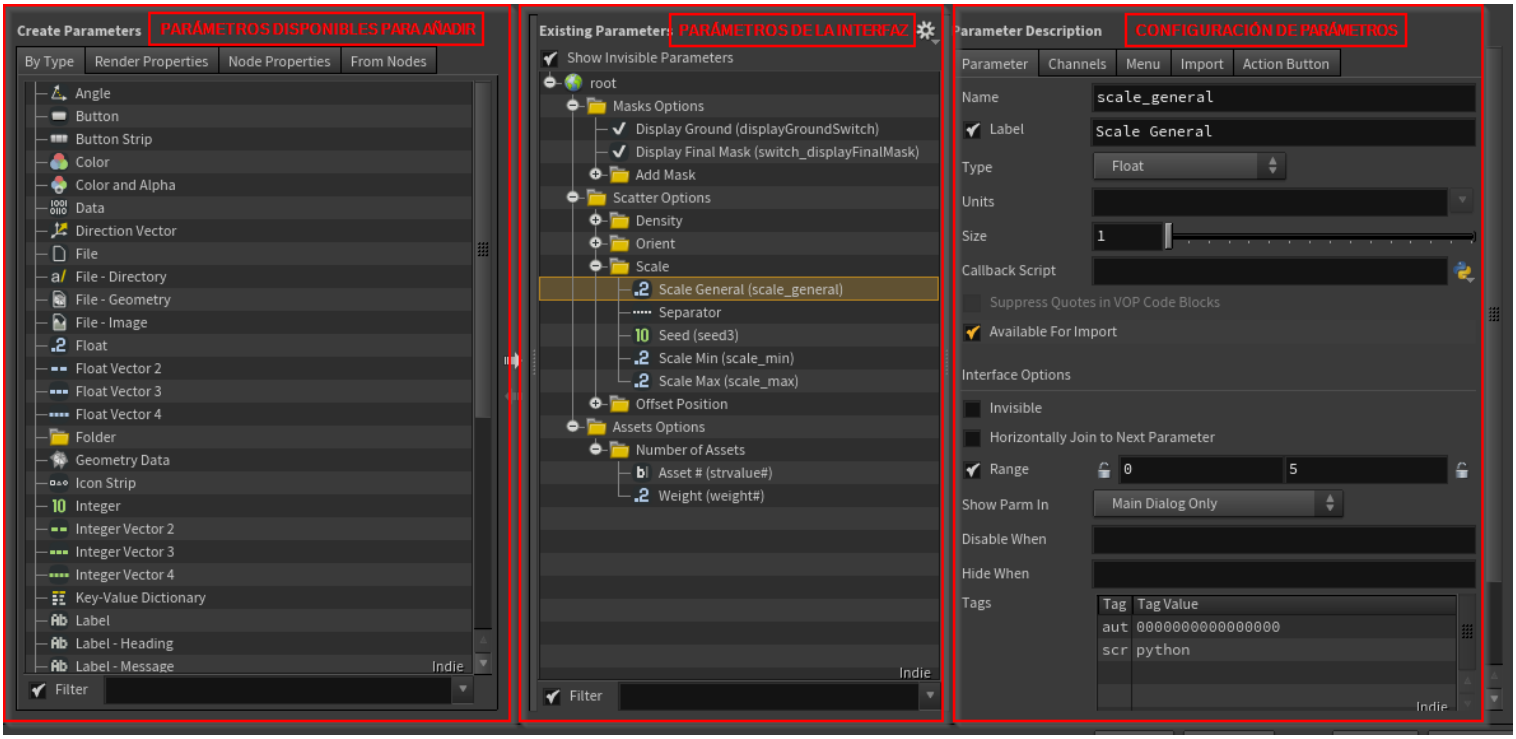


Figura 27: Houdini Setup – Configuración de Interfaz de usuario

Una vez creados y configurados los parámetros de la interfaz, es necesario vincularlos con los parámetros del árbol de nodos. Esto se logra utilizando expresiones de código basadas en el lenguaje nativo de Houdini.

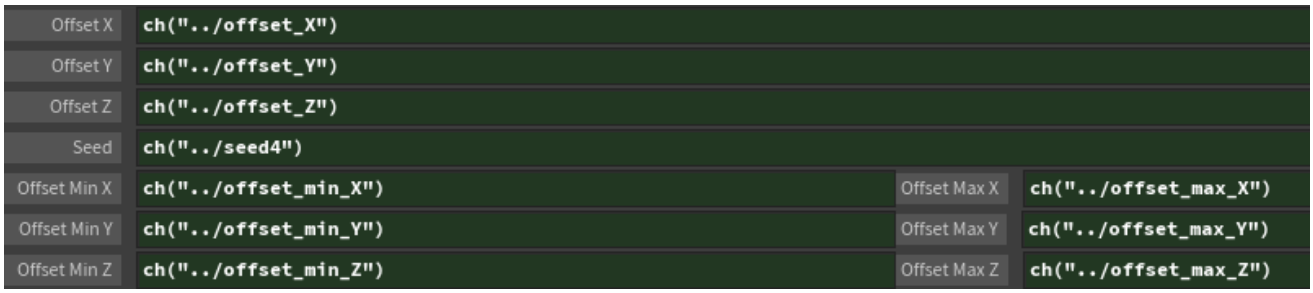


Figura 28: Houdini Setup – Vinculación de parámetros

5. Implementación

5.1. Requisitos de instalación

A continuación, se detallan los requisitos mínimos necesarios para la instalación y uso de la herramienta:

- **Software:**
  - **Houdini Engine for Unreal**: Necesario para ejecutar y manipular la herramienta. La licencia se obtiene de manera gratuita junto con la compra de una licencia Indie/Core/Fx de Houdini SideFx. Versión utilizada: 20.0.653.
  - **Unreal Engine 5**: Motor gráfico donde se ejecutará la herramienta y donde se desarrollará el proyecto en el que se usará la herramienta. Versión utilizada: 5.3.2.

- **Hardware**
  - **Estación de trabajo de altas prestaciones**. Mínimo recomendable:
    - **Procesador:** AMD Ryzen 7 5800X / Intel Core i7-11700K
    - **Memoria RAM:** 32 GB DDR4 3200 MHz
    - **Tarjeta Gráfica:** NVIDIA GeForce RTX 3060 12GB

- **Formación/Conocimientos**
  - **Conocimientos en Unreal Engine**: Experiencia básica en la utilización de Unreal Engine. No es necesario ningún tipo de conocimiento en Houdini ni conocimientos avanzados en Unreal Engine.

- **Otros requisitos**
  - ***Blueprints / Assets*:** Conjunto de elementos que desean ser instanciados. El usuario deberá disponer de los elementos que desea instanciar para proporcionárselos a la herramienta como entrada. Además, también deberá disponer del elemento sobre el que se realizarán las instancias, generalmente será la geometría que hará la función de suelo.

5.2. Instrucciones de instalación

En el siguiente enlace se encuentra la información oficial de SideFx sobre cómo realizar la instalación de Houdini Engine for Unreal:

https://www.sidefx.com/docs/houdini/unreal/install_houdiniengine.html

Para instalar la herramienta en Unreal Engine tan solo hay que **arrastrarla a cualquier carpeta de nuestro proyecto**. En este caso se ha creado una nueva carpeta llamada HDA (Houdini Digital Asset).

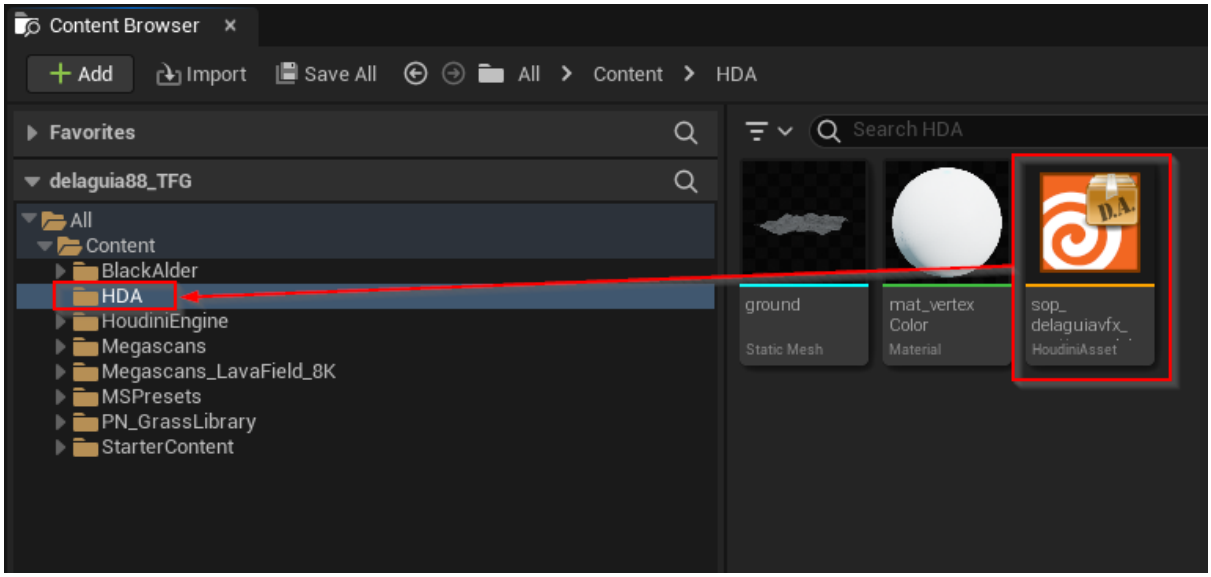


Figura 29: Unreal – Importación de la herramienta

6. Demostración

6.1. Instrucciones de uso

A continuación, se detallan los pasos previos necesarios para el uso de la herramienta, así como la explicación de las diferentes secciones de la herramienta y sus funcionalidades.

6.1.1. Preparación previa

Para usar la herramienta, es necesario asegurarse de que ciertos elementos estén presentes en el proyecto de Unreal Engine y seguir algunos pasos previos muy sencillos.

Elementos necesarios:

1. **Geometría del suelo**: Debe incluirse en el proyecto el elemento o geometría que servirá como suelo, donde se realizará el instanciado.

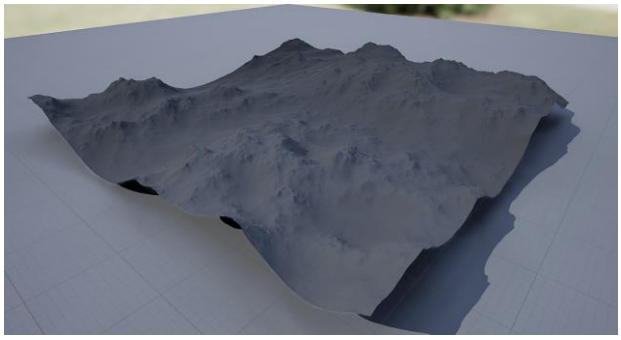


Figura 30: Geometría base

2. ***Blueprints*/Elementos para Instanciar**: Deben estar disponibles los *blueprints* o elementos específicos que se desean instanciar en el proyecto.

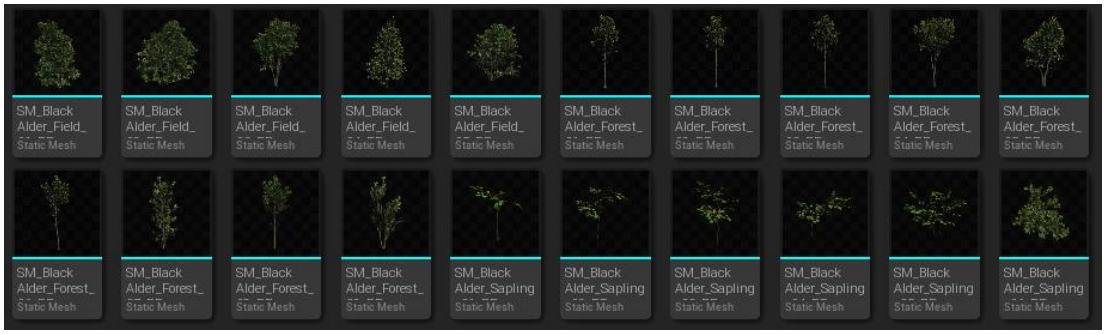


Figura 31: Blueprints para instanciar

Pasos previos:

1. **Material *vertex color*:** Para una mejor visualización, se recomienda crear y configurar un material que muestre en el *viewport* la información del *vertex color*.

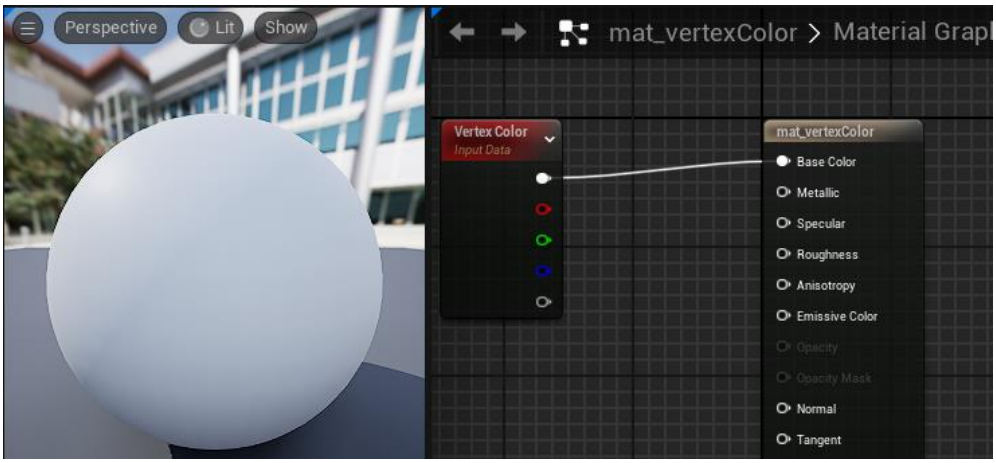


Figura 32: Creación de material vertex color

2. **Duplicar la geometría "suelo":** Aunque no es indispensable, sí que es recomendable duplicar la geometría sobre la que haremos el *scatter* ya que tendremos que manipularla.

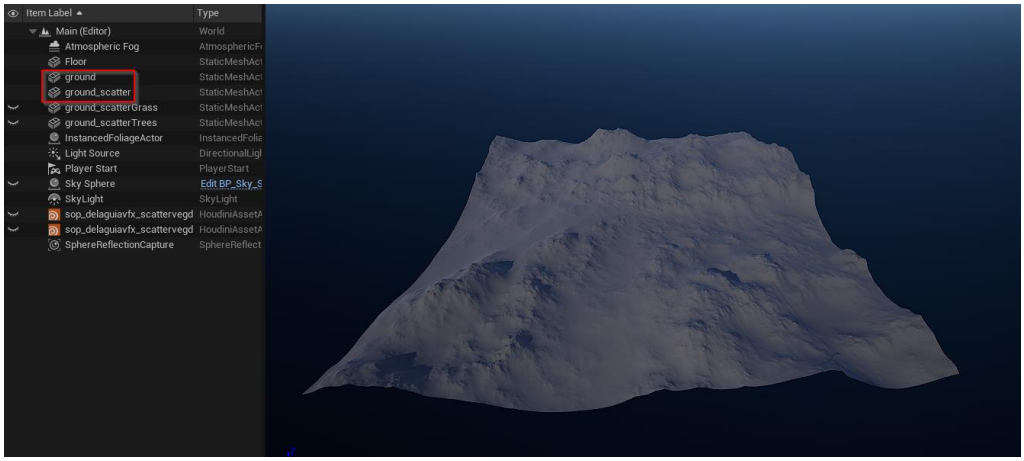


Figura 33: Duplicado de geometría base

3. **Aplicar el material *vertex color* a la geometría duplicada:** Este material nos servirá para visualizar correctamente la máscara que pintaremos en el siguiente paso.

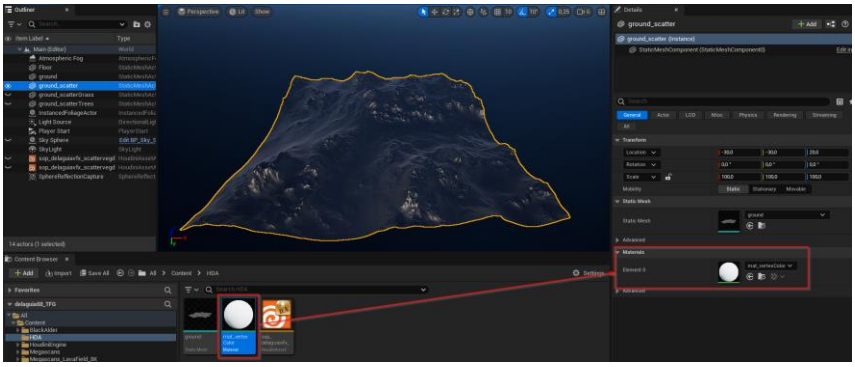


Figura 34: Aplicación de material vertex color

4. **Pintar de rojo el área deseada:** En este paso, se debe pintar la geometría base de un color neutro (negro o blanco) y, utilizando la herramienta de pincel, crear una primera máscara general del área donde se realizará el instanciado. No es necesario que esta máscara sea muy detallada, ya que se enriquecerá posteriormente con la herramienta.

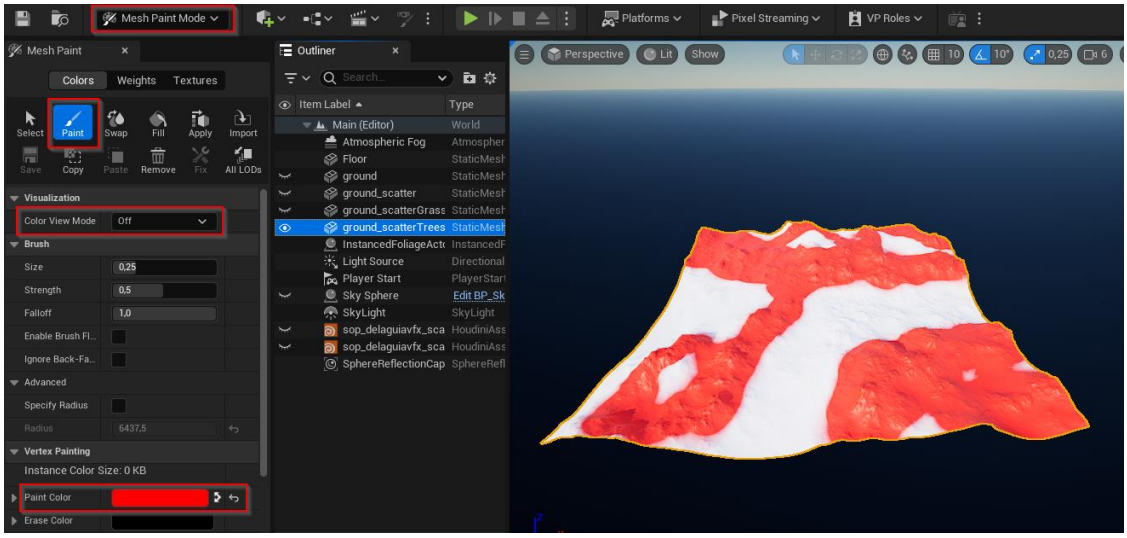


Figura 35: Creación de máscara manual

Si el *scatter* va a ser aplicado sobre todo el área en general, como puede ser en el caso de hierba, se debe pintar toda la geometría de color rojo.

5. **Inicializar la herramienta:** Seleccionarla desde el “Content Browser” y arrastrarla a cualquier lugar del *viewport*.

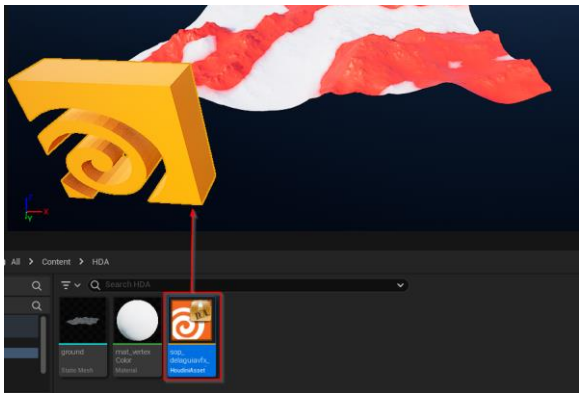


Figura 36: Inicialización de la herramienta

6. **Asignar la geometría base:** Con la herramienta seleccionada en el *outliner*, acceder al menú “Houdini *Inputs*” y añadir la geometría sobre la que se quiere realizar el *scatter* seleccionándola desde “World Input”.

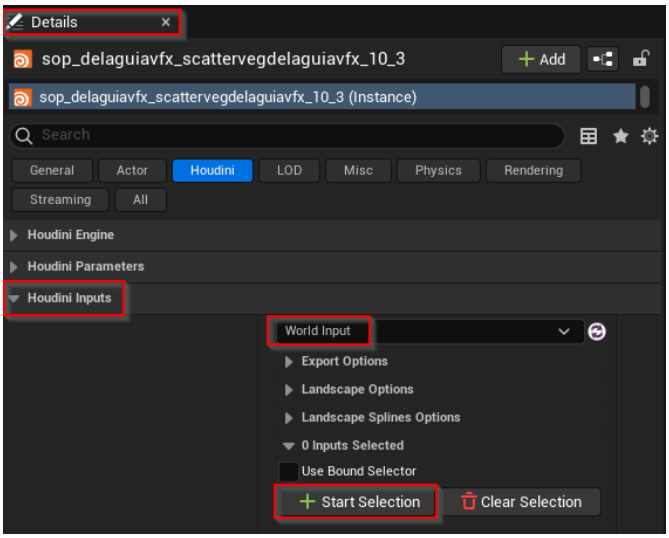


Figura 37: Asignación a la geometría base

Una vez añadida correctamente, la geometría aparecerá en el listado de *inputs*.

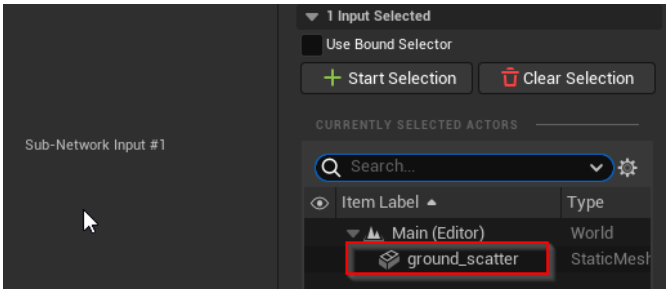


Figura 38: Geometría base añadida

Una vez asignada la geometría base con la máscara correspondiente, todo estará listo para usar la herramienta. Si se desean aplicar diferentes *scatters* en distintas zonas, se recomienda duplicar la geometría del suelo tantas veces como sea necesario y pintar las máscaras correspondientes para cada *scatter*. El flujo de trabajo habitual consiste en separar los elementos según sus características. Por ejemplo, en un proyecto complejo, se puede tener un *scatter* para árboles, otro para arbustos, uno para césped, otro para rocas grandes y otro para piedras pequeñas.

6.1.2.Houdini *Parameters*

Para acceder a los parámetros de la herramienta, ir al menú “Houdini Parameters”. La herramienta se divide en tres secciones claramente diferenciadas: *Masks Options, Scatter Options* y *Assets Options*.

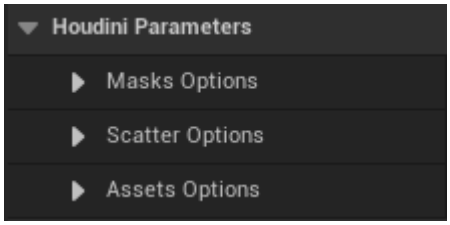


Figura 39: Secciones herramienta

6.1.3.Masks Options

En esta sección se configuran las diferentes máscaras que se utilizarán para realizar el *scatter*. Aquí se pueden añadir máscaras procedurales que se combinarán entre sí sumándose, restándose, multiplicándose, etc. para producir una máscara final de gran riqueza. Estas máscaras permiten obtener en el *scatter* un aspecto natural y lograr fácilmente el resultado deseado.

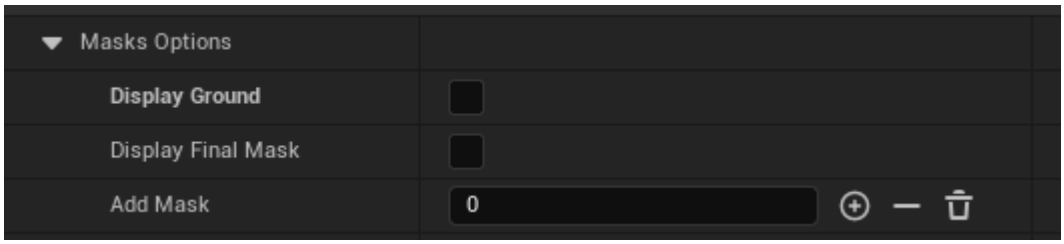


Figura 40: Opciones de máscaras

La visualización de todas las máscaras seguirá el siguiente esquema de color, donde el color púrpura representará el valor mínimo (0) y el color rojo representará el valor máximo (1).

Figura 41: Rampa de color utilizada en la visualización de las máscaras

En esta sección tenemos las siguientes opciones:

A. ***Display Ground*:** Opción de visualización para ocultar y desocultar una copia de la geometría base en el *viewport*. Se deberá desactivar al terminar el *scatter*.

B. ***Display Final Mask*:** Opción que muestra el resultado final de la combinación de todas las máscaras.

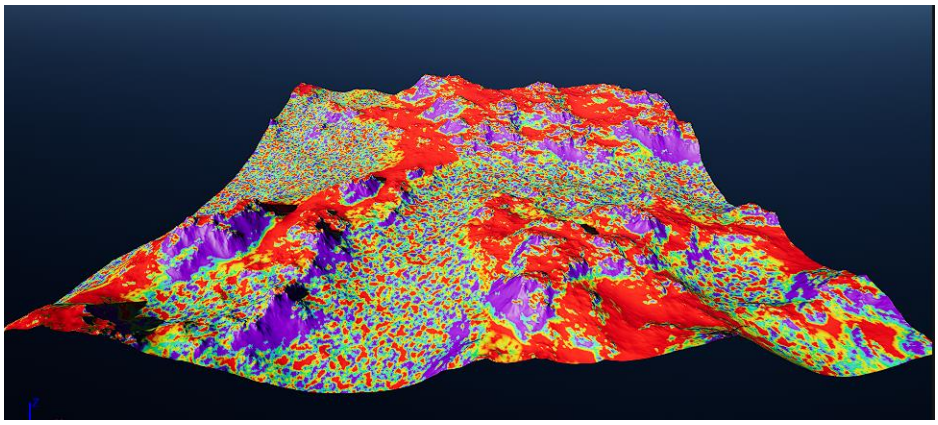


Figura 42: Visualización de la máscara final

C. ***Add Mask*:** Opción que añade una nueva máscara. Con esta opción se irán creando todas y cada una de las máscaras deseadas.

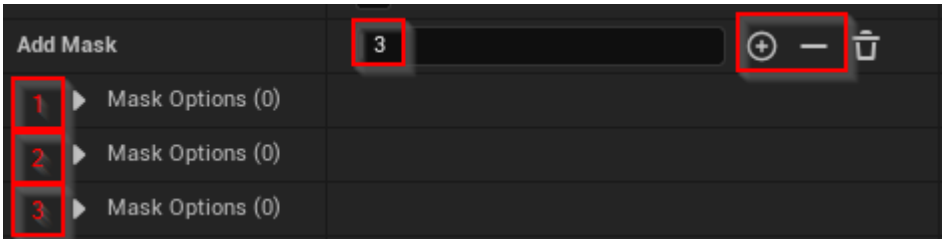


Figura 43: Añadir nueva máscara

D. ***Mask Options*:** Al añadir una nueva máscara se mostrará un menú desplegable con las opciones para configurar cada una de las máscaras de manera independiente. Cada tipo de máscara tiene opciones diferentes.

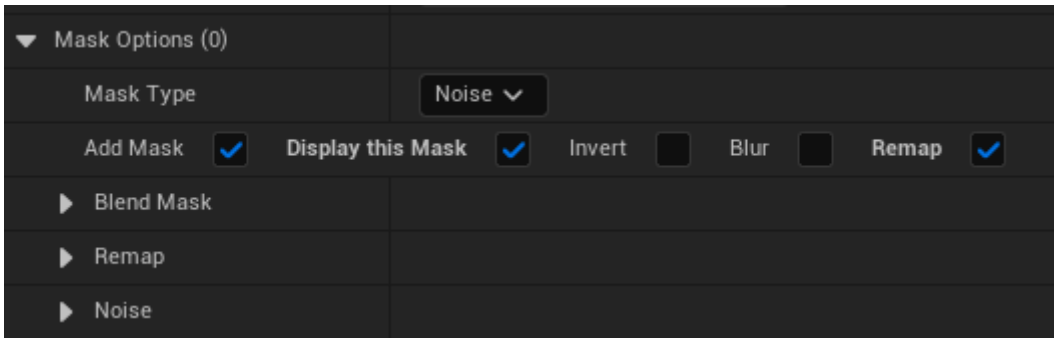


Figura 44: Opciones de las máscaras

E. ***Use Mask*:** En ocasiones el usuario desea desactivar una máscara para comparar el resultado con ella y sin ella o sencillamente quiere no utilizarla en ese momento, pero no desea eliminarla del todo. Con esta opción puede añadir o eliminar la máscara del resultado final sin eliminarla del todo. Por defecto estará activada

F. ***Display this Mask*:** Opción que permite visualizar solamente la máscara seleccionada en lugar de la combinación de todas las máscaras. Muy útil para ver como realmente afecta una máscara cuando se está configurando.

G. ***Invert*:** Permite invertir los valores de una máscara.

H. ***Blur*:** Añade opciones para desenfocar la máscara y obtener transiciones más suaves.

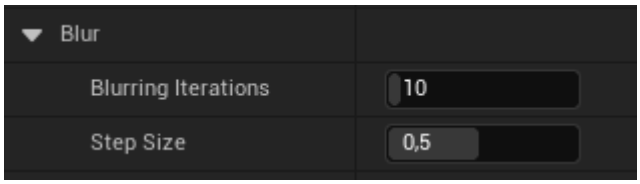


Figura 45: Desenfocar máscara

I. **Remap:** Agrega una curva de remapeo para remapear los valores de la máscara. Muy útil para “endurecer” los valores.

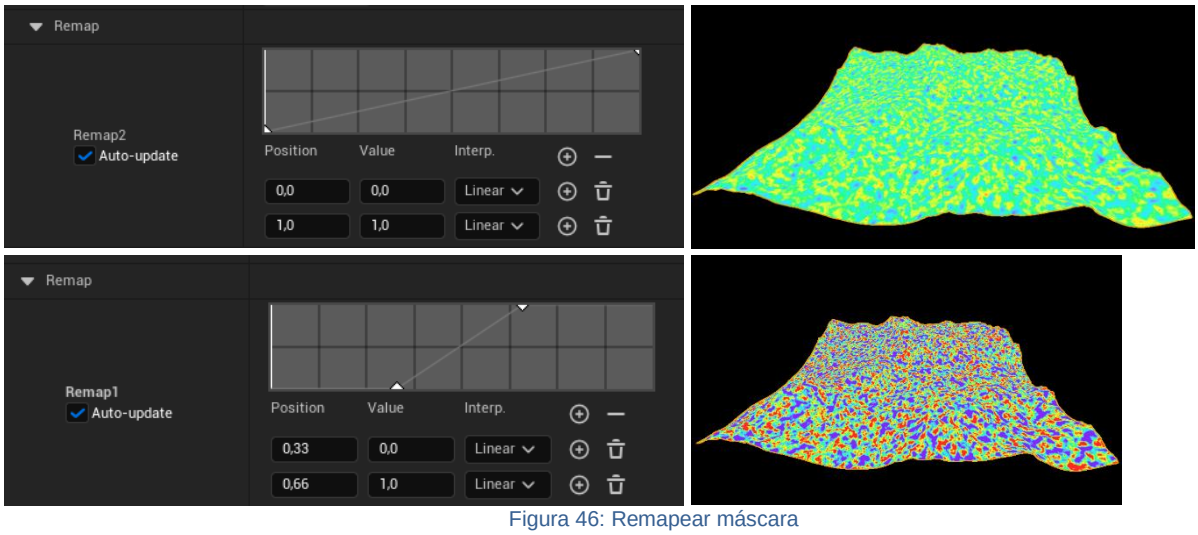


Figura 46: Remapear máscara

J. ***Blend Mask*:** Define la forma en la que se integrará la máscara con las máscaras anteriores. Se pueden realizar operaciones como sumar, restar, multiplicar, dividir, seleccionar el valor máximo, seleccionar el valor mínimo y mezclar. De esta manera, se combinarán las diferentes máscaras para generar el resultado final.

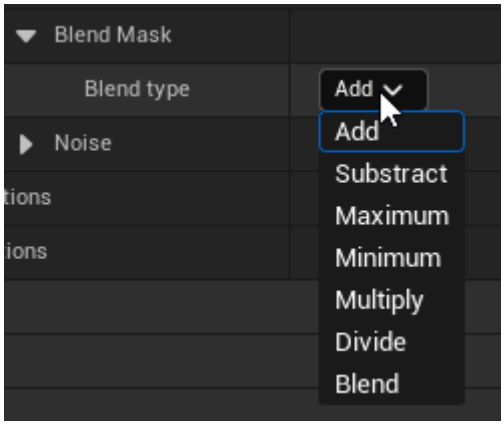


Figura 47: Opciones de combinación

K. ***Mask type*:** Permite seleccionar entre distintos tipos de máscaras: *Noise, Direction, Normal, Ambient Oclusion* y *Height*.

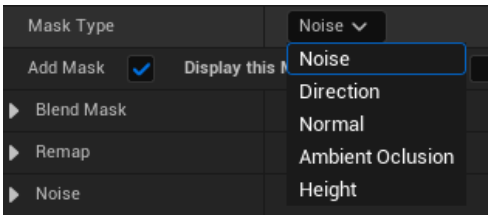


Figura 48: Tipos de máscarass

L. ***Noise*:** Genera un ruido procedural que servirá como máscara. Ofrece diferentes opciones para controlar la escala, la frecuencia y la intensidad del ruido.

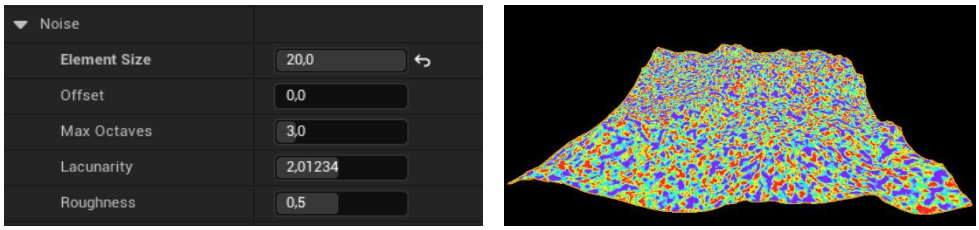


Figura 49: Máscara noise

M. ***Direction*:** Máscara generada proyectando un vector desde uno de los ejes (X, Y, Z) y utilizando el ángulo máximo entre el vector y la normal de la superficie. Un ejemplo de uso es enmascarar la zona "Norte" (-Y) de un terreno para simular una zona más húmeda y con mayor vegetación.

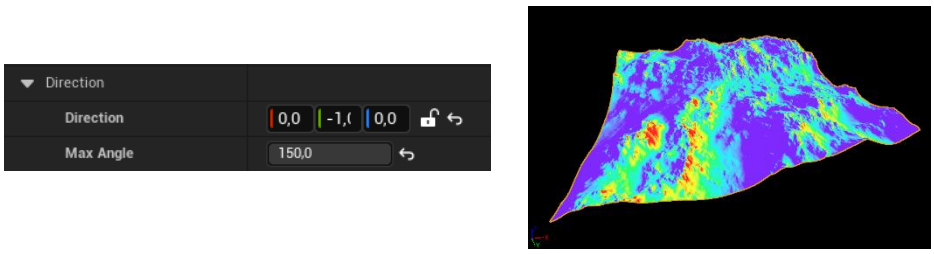


Figura 50: Máscara direction

N. **Normal:** Máscara basada en la componente 'Y' de las normales de la geometría, con un rango de -1 a 1. Incorpora un parámetro para suavizar la transición de la máscara. De gran utilidad para seleccionar todas las zonas horizontales o verticales del terreno, por ejemplo, para no poner árboles o vegetación en paredes verticales

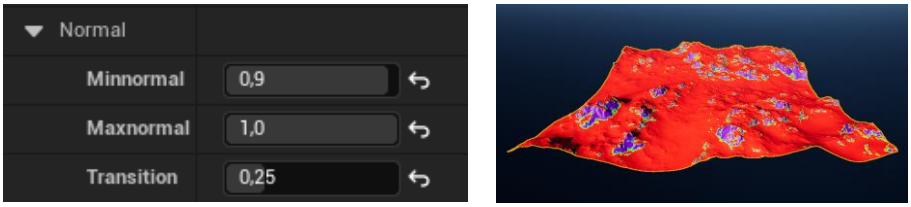


Figura 51: Máscara normal

O. ***Ambient Oclusión*:** Máscara basada en la oclusión ambiental de la geometría. Es muy situacional y útil, por ejemplo, para enmascarar grietas en el terreno.

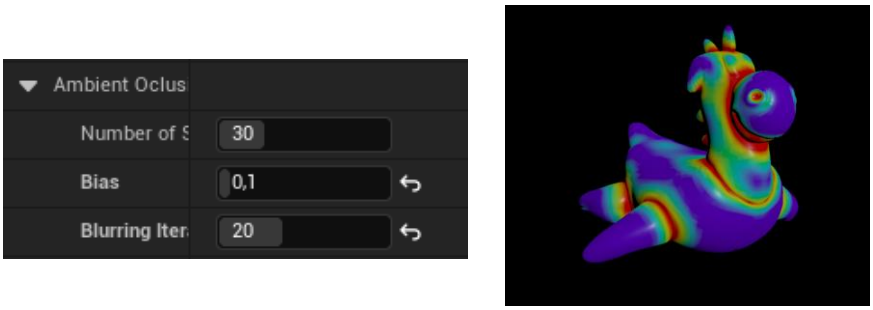


Figura 52: Máscara Ambient Oclusion

P. **Height:** Máscara basada en la altura de la geometría, ya sea respecto al mundo o a la propia geometría. Es muy útil para diferenciar zonas en altura, por ejemplo, las cimas de las montañas donde puede que no se desee vegetación o se quiera un tipo de vegetación diferente.

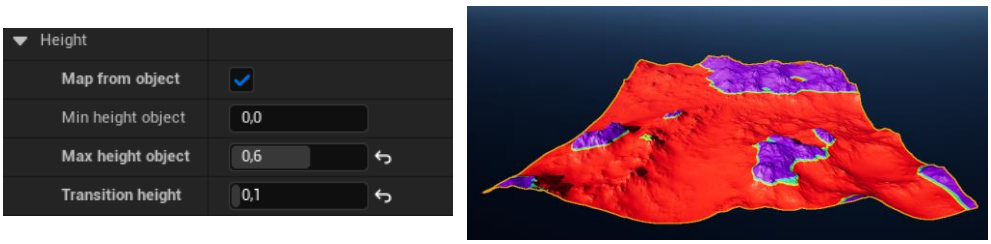


Figura 53: Máscara height

6.1.4.Scatter Options

En esta sección se configuran los parámetros relacionados con el *scatter* de los elementos instanciados. Aquí se determina la cantidad de copias que se crearán y se definen las diversas transformaciones, tanto conjuntas como individuales, que se aplicarán a estas copias. Estas opciones permiten personalizar la distribución y apariencia de los elementos en el entorno, asegurando un resultado final coherente y visualmente atractivo.

Figura 54: Opciones de scatter

1. **Density:** Esta opción permite configurar la cantidad de copias a instanciar y su distribución. Incluye ajustes para controlar la densidad y la relajación, lo que permite distribuir los puntos de manera más uniforme o más aleatoria, según se desee.

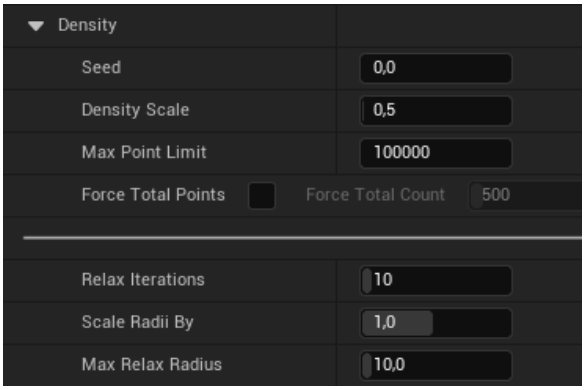


Figura 55: Opciones de densidad

2. ***Orient*:** En esta sección se configuran los parámetros relacionados con la orientación de las copias instanciadas. Se pueden ajustar las rotaciones en los ejes X, Y y Z de manera general para todas las instancias, así como agregar una rotación aleatoria a cada copia. Además, se puede optar por orientar las instancias en relación al mundo o en relación a la dirección de la normal de la geometría base. Por ejemplo, los árboles generalmente se orientan en relación al mundo.

Figura 56: Opciones de orientación

3. ***Scale*:** Permite configurar cómo se escalarán las copias instanciadas. Se puede ajustar una escala general para todas las instancias y definir un rango mínimo y máximo para introducir variaciones aleatorias en la escala de cada copia.

Figura 57: Opciones de escala

4. ***Ofsset Position*:** Configura el desplazamiento de las copias instanciadas en los ejes X, Y y Z. Se pueden establecer valores generales para el desplazamiento y definir rangos mínimos y máximos para introducir variaciones aleatorias en la posición de cada copia.

Figura 58: Opciones de posición

6.1.5.Assets Options

En este apartado se configura el número de elementos a instanciar y sus propiedades individuales. Se puede especificar la cantidad de *assets* y ajustar el peso de cada uno para controlar su distribución en el *scatter*. Además, se pueden agregar o eliminar *assets* según sea necesario para lograr la combinación deseada.

Figura 59: Opciones de assets

1. **Number of Assets:** Define la cantidad de elementos que se van a instanciar en el *scatter*. Permite ajustar el número total de diferentes tipos de *assets* que se incluirán en la distribución.

2. **Asset:** Se refiere a cada uno de los elementos individuales que se van a instanciar. Aquí se arrastrará el *asset* específico que se desea utilizar.

3. **Weight:** Establece la ponderación de cada *asset* en el *scatter*. Este valor determina la frecuencia con la que un *asset* particular aparece en comparación con otros *assets*, permitiendo un control preciso sobre la distribución final. Por ejemplo, si se desea que un *asset* aparezca solo en unas pocas copias a lo largo del *scatter*, se puede disminuir el valor de peso hasta lograr la cantidad adecuada.

Ejemplo:

Figura 60: Ejemplo assets

6.2. Prototipos

Durante el desarrollo de la herramienta se crearon tres prototipos principales, cada uno correspondiente a la implementación de las funcionalidades clave. Además, a lo largo del proceso se generaron diversas versiones intermedias para iterar sobre las funcionalidades y corregir errores. Estas iteraciones permitieron un desarrollo ágil, abordando y resolviendo problemas de manera continua y asegurando que cada funcionalidad se implementara de manera robusta y efectiva antes de proceder a la siguiente etapa.

Prototipo 1: Funcionalidades de las Máscaras

- El primer prototipo se centró en implementar y probar las funcionalidades básicas de las máscaras.
- Este prototipo permitió definir y ajustar las diferentes opciones de creación y combinación de máscaras, así como los tipos de máscaras disponibles.
- Fue crucial para asegurarse de que las máscaras se aplicaran correctamente, ya que son la base de los *scatters*.

Prototipo 2: Funcionalidades del *Scatter*

- El segundo prototipo incorporó las funcionalidades del *scatter*, permitiendo controlar la cantidad de instancias y su espaciado, así como modificar las transformaciones de las copias instanciadas.
- Este prototipo permitió evaluar y refinar el *scatter* de los elementos, asegurando que las transformaciones se aplicaran correctamente.

Prototipo 3: Implementación de Blueprints de Unreal

- El prototipo final integró la herramienta con los *blueprints* de Unreal Engine, completando así todas las funcionalidades previstas.
- En esta fase, se implementó la capacidad de leer correctamente los *blueprints* de Unreal Engine y utilizarlos como elementos a instanciar en el *scatter*.
- Esta integración fue fundamental para que la herramienta funcionara dentro del entorno de Unreal Engine según los objetivos del proyecto.

6.3. Tests

En este apartado se detallan las pruebas de funcionamiento, rendimiento y usabilidad realizadas para asegurar la calidad y eficacia de la herramienta.

6.3.1. Pruebas de Funcionamiento

Se han llevado a cabo exhaustivas pruebas de funcionamiento para todas las funciones de la herramienta. Durante estas pruebas, se han detectado y corregido diversos problemas hasta lograr un funcionamiento correcto y estable. Cada función ha sido testeada de manera individual y en conjunto, asegurando que todas las características operen según lo esperado.

El proceso de prueba incluyó la verificación de la correcta aplicación de las máscaras, el *scatter* de los elementos instanciados y la interacción con los parámetros de orientación, escala y desplazamiento.

6.3.2. Pruebas de Rendimiento

Para evaluar el rendimiento de la herramienta, se han creado escenarios muy grandes con millones de copias instanciadas. Los resultados han sido en su mayoría satisfactorios, mostrando que la herramienta puede manejar grandes volúmenes de datos y realizar *scatters* a gran escala de manera eficiente.

Sin embargo, en las pruebas más exigentes se han observado algunos problemas, como errores de visualización y fallo del programa. Estos problemas han sido atribuidos principalmente a la falta de memoria RAM en la tarjeta gráfica, lo que indica la necesidad de contar con hardware adecuado para manejar escenarios extremadamente grandes.

6.3.3. Pruebas de Usabilidad

Se han realizado pruebas de usabilidad con un grupo muy reducido de personas. Los resultados de estas pruebas han sido muy positivos, ya que los usuarios han logrado realizar *scatters* complejos con facilidad, a pesar de su limitada experiencia con Unreal Engine.

La interfaz de usuario y los controles intuitivos han permitido que incluso los usuarios novatos puedan utilizar la herramienta de manera efectiva, lo que demuestra su accesibilidad y facilidad de uso.

6.4. Ejemplos de uso del producto

A continuación, se presentan una serie de imágenes que muestran los resultados obtenidos con la utilización de la herramienta desarrollada. Para la realización de los scatters, se ha empleado la herramienta creada, utilizando elementos gratuitos provenientes del Marketplace de Unreal, los cuales han sido instanciados.

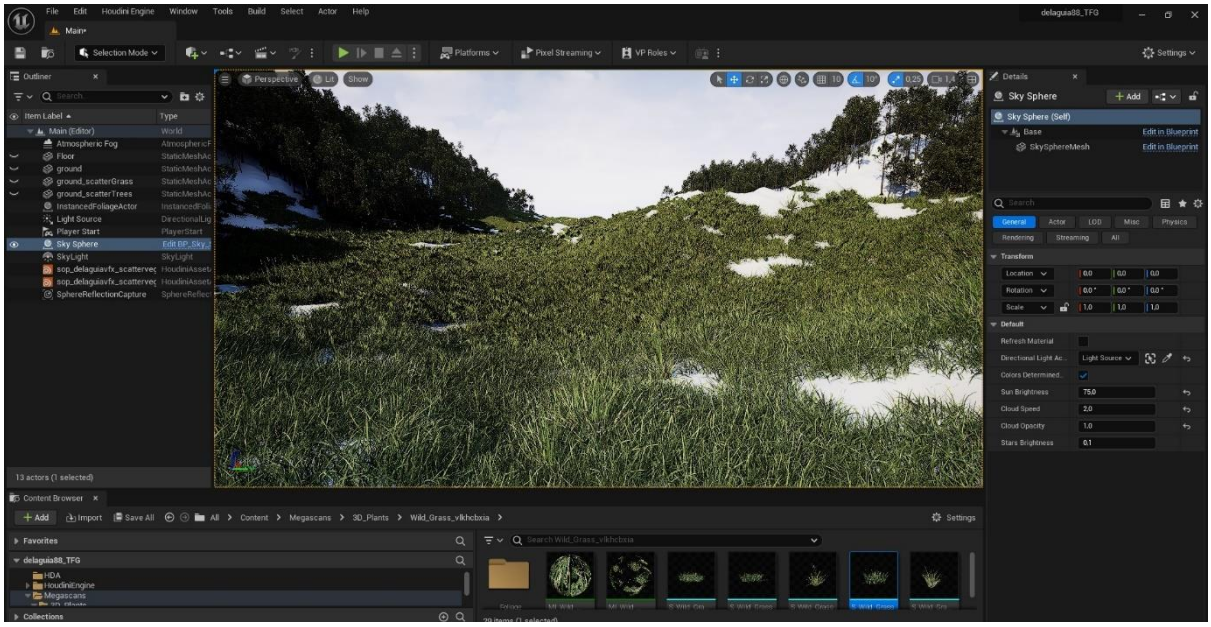


Figura 61: Ejemplos de uso. Interfaz

En los ejemplos 1 y 2 se ha utilizado la misma base, es decir, el mismo terreno y los mismos atributos de máscaras y de scatter. La única diferencia reside en los elementos a instanciar. Gracias a la proceduralidad, con una única configuración se pueden generar diferentes escenarios cambiando únicamente los elementos instanciados, de manera muy rápida.

En el ejemplo 3, se han utilizado los mismos elementos a instanciar, pero con diferentes geometrías base. Este ejemplo refleja nuevamente la proceduralidad del sistema, ya que se adapta perfectamente a cualquier tipo de geometría base, lo cual permite crear una configuración que se puede repetir en distintos terrenos.

Ejemplo 1: Misma geometría base, diferentes elementos.

Figura 62: Ejemplos de uso. Diferentes elementos 1

Ejemplo 2: Misma geometría base, diferentes elementos.

Figura 63: Ejemplos de uso. Diferentes elementos 2

Ejemplo 3: Diferente geometría base, mismos elementos.

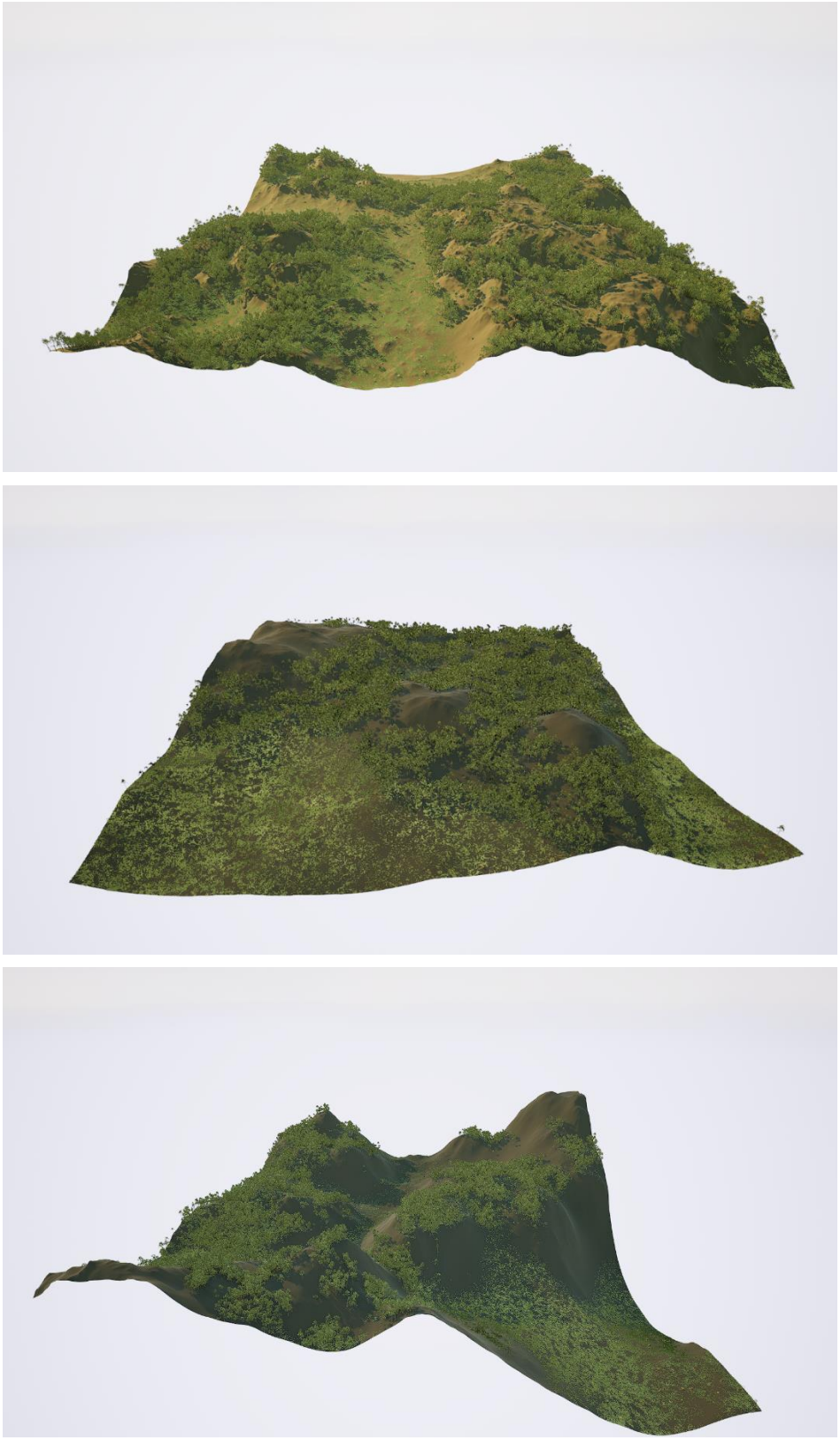


Figura 64: Ejemplos de uso. Diferentes geometrías base

Conclusiones y líneas de futuro

6.5. Conclusiones

Realizar este proyecto ha sido una experiencia desafiante y gratificante. El mayor reto ha sido integrar la herramienta con Unreal Engine, un motor gráfico que nunca antes había utilizado. A lo largo del proceso, he adquirido un amplio conocimiento sobre Unreal, descubriendo la potencia de los *blueprints* y la relativa simplicidad inicial de conectar Houdini con Unreal. Sin embargo, la verdadera complejidad radicó en lograr una conexión bidireccional, permitiendo que la herramienta en Houdini recibiera *inputs* de Unreal y los interpretara correctamente. Además, he comprobado por qué Unreal Engine es el estándar en la industria, al sorprenderme en mi primer test masivo instanciando millones de árboles y moviéndome libremente por la escena en tiempo real con una iluminación realista, algo impensable para un software 3D tradicional.

En cuanto a los objetivos planteados inicialmente, he logrado cumplir todos los principales. El resultado final ha sido una herramienta que facilita la creación de *scatters* procedurales masivos de manera sencilla y sin requerir conocimientos avanzados. Aunque el objetivo original era desarrollar una serie de herramientas con distintas funcionalidades, he optado por combinarlas en una sola herramienta, ya que resultaba más práctico.

Respecto a los objetivos secundarios, he implementado el soporte para elementos animados satisfactoriamente. Sin embargo, no he alcanzado completamente la interactividad con geometrías y la resolución de intersecciones. En Houdini, estas funcionalidades funcionaban correctamente, pero al trasladarlas a Unreal no he conseguido que la herramienta interpretara adecuadamente los *inputs* de geometría. Esto ha resaltado la dificultad de transformar los *inputs* de Unreal para que Houdini los interprete correctamente.

En cuanto a la planificación y la metodología, he tratado de seguir la planificación lo más rigurosamente posible y, en general, creo que lo he logrado, en parte porque estaba alineada con los requisitos para las entregas de las PEC. No obstante, en algunas ocasiones no he podido cumplir con los objetivos planificados debido a la falta de tiempo, ya que he tenido que conciliar mis responsabilidades laborales, familiares y académicas. A pesar de estas dificultades, considero que la metodología y la planificación han sido adecuadas para este tipo de proyecto y, probablemente, volvería a abordarlo de la misma manera.

En resumen, este proyecto ha representado un importante avance tanto en mis habilidades técnicas como en mi comprensión de las herramientas de desarrollo de videojuegos. A pesar de los desafíos y limitaciones encontradas, he conseguido desarrollar una herramienta funcional y práctica que cumple con los objetivos principales y proporciona una base sólida para futuras mejoras y desarrollos adicionales. Esta experiencia ha reafirmado la importancia de una planificación meticulosa y una metodología adaptable, demostrando que la combinación de esfuerzo, aprendizaje continuo y adaptación puede llevar a resultados exitosos y satisfactorios.

6.6. Líneas de futuro

6.6.1. Interactividad con Geometrías y Creación de Máscaras

Una de las mejoras más importantes es lograr una interactividad completa con geometrías y la creación avanzada de máscaras.

Esto se puede dividir en varias subáreas clave:

- **Creación de Máscaras a través de Elementos de Unreal:** Una futura mejora consistiría en permitir que elementos de Unreal Engine, como geometrías, curvas e incluso otros *scatters*, actúen como *inputs* para generar máscaras. Estos elementos se utilizarán para crear máscaras que reflejen el espacio que ocupan en la escena, y dichas máscaras se podrán sumar o restar del resultado final.

- **Implementación de Máscaras de Sombras:** Integrar máscaras de sombras que proyecten rayos para identificar las zonas de sombra en la escena. Un ejemplo práctico de esta funcionalidad sería la creación automática de vegetación que solo crece en áreas sombreadas bajo los árboles, mejorando la realismo y detalle de los entornos.

- **Control de Parámetros a través de Mapas de Información:** Añadir la opción de utilizar los mapas de información creados no solo para la distribución de elementos, sino también para controlar parámetros como la escala, rotación y tipo de *asset* instanciado. Esto permitiría una mayor personalización y precisión en la configuración del *scatter*.

6.6.2. Resolución de Intersecciones y Mejoras en la Usabilidad

- **Detección y resolución de Intersecciones:** Implementar una funcionalidad que detecte intersecciones durante el proceso de *scatter* para asegurar que las geometrías instanciadas no se intersecten entre sí o con algún elemento seleccionado.

- ***Presets* y Biomas:** Desarrollar una funcionalidad que permita guardar y cargar *presets*, facilitando la reutilización de configuraciones de scatter. Además, incluir *presets* predefinidos para la creación rápida de diferentes biomas, agilizando el proceso de diseño y mejorando la eficiencia del flujo de trabajo.

Estas mejoras no solo aumentarían la funcionalidad y versatilidad de la herramienta, sino que también optimizarían su uso y adaptabilidad a diversas necesidades de diseño, asegurando que pueda satisfacer las demandas crecientes de la industria de los videojuegos.

Bibliografía

- **Epic Games (2024) *Unreal Engine 5.4 Documentation*. Disponible en: https://dev.epicgames.com/documentation/es-es/unreal-engine/unreal-engine-5-4-documentation?application_version=5.4 (Consultado del 10 de abril de 2024 al 15 de mayo de 2024)**

- **Epic Games (2024) *Procedural Instance Spawner*. Unreal Engine Marketplace. Disponible en: https://www.unrealengine.com/marketplace/en-US/product/procedural-instance-spawner (Consultado el 11 de marzo de 2024)**

- **Epic Games (2024) *Simple Scatter*. Unreal Engine Marketplace. Disponible en: https://www.unrealengine.com/marketplace/en-US/product/simple-scatter/questions (Consultado el 23 de marzo de 2024)**

- **Epic Games (2024) *Hierarchical Scatter Tool*. Unreal Engine Marketplace. Disponible en: https://www.unrealengine.com/marketplace/en-US/product/hierarchical-scatter-tool?sessionInvalidated=true (Consultado el 30 de abril de 2024)**

- **Epic Games (2024) *Creating a Scatter Tool*. Unreal Engine Marketplace. Disponible en: https://www.unrealengine.com/marketplace/en-US/product/creating-a-scatter-tool-course-files (Consultado el 19 de marzo de 2024)**

- **Epic Games (2024) *Houdini Attributes Data*. Unreal Engine Documentation. Disponible en: https://docs.unrealengine.com/4.26/en-US/BlueprintAPI/HoudiniAttributesData/GetColorValue/ (Consultado el 3 de mayo de 2024)**

- **One Minute VEX (2024) *Ray-Cast Ambient Occlusion Using Intersect*. Disponible en: https://1minutevex.com/books/OneMinuteVex.html#ray-castambientocclusionusingintersect (Consultado el 5 de mayo de 2024)**

- **Reddit (2024) *Unreal Houdini Engine is My Favourite Thing Ever*. Disponible en: https://www.reddit.com/r/unrealengine/comments/dfebqv/unreal_houdini_engine_is_my_favourite_thing_ever/ (Consultado el 14 de marzo de 2024)**

- **SideFX (2024) *Houdini Documentation*. Disponible en: https://www.sidefx.com/docs/houdini/index.html (Consultado del 7 de marzo de 2024 al 15 de mayo de 2024)**

- **SideFX (2024) *Houdini Forum*. Disponible en: https://www.sidefx.com/forum/ (Consultado del 7 de marzo de 2024 al 15 de mayo de 2024)**

- **SideFX (2024) *Foundations: Procedural Assets for Unreal*. Disponible en: https://www.sidefx.com/tutorials/foundations-procedural-assets-for-unreal/ (Consultado del 7 de marzo de 2024 al 15 de mayo de 2024)**

- **Peyton Varney (2022) *How to SETUP Vertex Painting in Unreal Engine 5*. YouTube. Disponible en: https://www.youtube.com/watch?v=86vm_m0_YbQ (Consultado el 29 de marzo de 2024)**

- **InfinaWorks (2024) *Procedural Instance Spawner plugin – Get started*. YouTube. Disponible en: https://www.youtube.com/watch?v=qmq-x-SIxfY (Consultado el 2 de abril de 2024)**

- **CG Dealers (2023) *Create Massive World Landscape Auto Material*. YouTube. Disponible en: https://www.youtube.com/watch?v=rw8qDmFGsRo (Consultado el 1 de mayo de 2024)**

- **Paul Ambrosiussen (2022) *Creating and Updating Blackboxed Assets in Houdini*. YouTube. Disponible en: https://www.youtube.com/watch?v=qcNN31BKGgA&t=182s (Consultado el 2 de mayo de 2024)**

- **Bubblepins (2024) *Beginner's Guide to Unreal Engine 5 with Houdini*. YouTube. Disponible en: https://www.youtube.com/watch?v=FFrBpbz3ym0 (Consultado el 12 de abril de 2024)**

- **Beyond Extent (2023) *Intro to Creating Houdini tools for Unreal Engine*. YouTube. Disponible en: https://www.youtube.com/watch?v=8R7JX-TFBfg (Consultado el 8 de marzo de 2024)**

- **Carlos Coronado (2021) *Tutorial Unreal Engine 5 para principiantes en Español*. YouTube. Disponible en: https://www.youtube.com/watch?v=OxzkkbpuxLA (Consultado el 15 de abril de 2024)**

- **Bubblepins (2020) *Introduction to HDA in Unreal Engine*. YouTube. Disponible en: https://www.youtube.com/watch?v=gAT8ueaKAvs&t=3243s (Consultado el 25 de marzo de 2024)**

- **Arise.Works (2020) *Using Attributes in Unreal Engine*. YouTube. Disponible en: https://www.youtube.com/watch?v=itsBu9G0gxY (Consultado el 9 de abril de 2024)**

- **Indie-Pixel (2024) *Houdini 18 – Python Projects*. YouTube. Disponible en: https://www.youtube.com/watch?v=ApC9b4bR2S4 (Consultado el 4 de abril de 2024)**

- **Surprised Monkey Studio (2024) *Scattering Procedural con Blueprints en Unreal Engine 5*. YouTube. Disponible en: https://www.youtube.com/watch?v=y4rXSO3vGr8 (Consultado el 8 de mayo de 2024)**

- **Tiny Talisman Games (2024) *Stylized Nature Pack*. Unreal Engine Marketplace. Disponible en: https://www.unrealengine.com/marketplace/en-US/product/stylized-pbr-nature-pack(Consultado el 01 de Junio de 2024)**

Anexos

Anexo A: Glosario

- ***Blueprints*:** Conjunto de nodos gráficos que permite crear funciones complejas dentro de Unreal Engine. En el contexto de este trabajo, se refiere a la utilización de assets completos que incluyen geometría, materiales, animación, rig, etc.
- **Houdini Digital *Asset* (HDA):** Herramienta creada en Houdini que encapsula una serie de nodos para realizar una funcionalidad específica. Esta herramienta es transportable a otros softwares, como Unreal Engine, gracias a Houdini Engine.
- **Instancia:** Copia de un elemento que referencia al objeto original. A diferencia de una copia normal, las instancias no duplican la información completa del objeto, lo que las hace muy eficientes para la creación de múltiples copias.
- **Procedural:** Método de creación de contenidos de manera automatizada a través de algoritmos, en lugar de hacerlo manualmente, permitiendo la generación de entornos y elementos de forma eficiente y rápida.
- ***Scatter*:** Técnica de distribución de elementos a lo largo de una escena de manera controlada y aleatoria según ciertos parámetros. La traducción al castellano sería "dispersión".
- ***Seeds*:** Valores iniciales utilizados en algoritmos procedurales para generar variaciones únicas de un contenido. Al cambiar la semilla, se pueden obtener diferentes resultados de un mismo proceso procedural. La traducción al castellano sería “semillas”.
- ***Vertex Color*:** Atributo de color aplicado a los vértices de una geometría, que puede ser utilizado para almacenar y manipular información adicional, como la densidad de un scatter.

Anexo B: Entregables del proyecto

- **Memoria del Trabajo:** Este documento, es la memoria del trabajo que recoge de manera exhaustiva todo el proceso de desarrollo del TFG.
- **HDA delaGuiaScatterTool:** El HDA es la herramienta desarrollada con Houdini que contiene todas las propiedades, configuraciones y funcionalidades del proyecto desarrollado.
- **Video Tráiler de la Herramienta:** Vídeo que presenta la herramienta desarrollada, destacando sus funcionalidades, sus beneficios y posibles aplicaciones.
- **Vídeo de Defensa del Trabajo:** Vídeo en el que se presenta y defiende el TFG explicando cómo ha sido el desarrollo de la herramienta, sus funcionalidades y los resultados obtenidos.
- **Documento de Cesión de Derechos:** Documento legal en el que se cede los derechos de para publicar y difundir el contenido del TFG a través de los canales y medios indicados.

Anexo C: Currículum Vitae

A continuación, se detallan diferentes sitios webs donde se encuentra la información relativa a mi currículum vitae y parte de mi portfolio, así como la información de contacto.

- **LinkedIn:** linkedin.com/in/albertodelaguia
- **Artstation:** artstation.com/albertodelaguia
- **Vimeo:** vimeo.com/albertodelaguia

Información de contacto:

- **Email:** albertovfx@gmail.com
- **Telf**: +34 676 135 692