Institución:
Universidad Autónoma de Aguascalientes
Centro:
Ciencias Básicas
Departamento:
Ciencias de la Computación
Posgrado:
Maestría en Ciencias con Opciones a la Computación, Matemáticas Aplicadas
Tesis Para la Obtención del Grado de Maestría:
Optimización con aprendizaje de máquina de método para obtención de nubes de puntos y su representación en realidad virtual
Estudiante:
Ing. Misael Alejandro Rivas Juárez (ID: 538387)
Tutores:
Dr. Hermilo Sánchez Cruz (Director)
Dr. Osvaldo Arturo Tapia Dueñas (Co-Director)
Asesor:
Dr. Julio César Ponce Gallegos

Aguascalientes, Ags., 15/06/2026

Mtro. Guillermo Domínguez Aguilar
DECANO (A) DEL CENTRO **DE CIENCIAS BÁSICAS**

P R E S E N T E

Por medio del presente como ***DIRECTOR*** designado del estudiante ***MISAEL ALEJANDRO RIVAS JUAREZ*** con ID *538387* quien realizó *la tesis* titulada: ***OPTIMIZACIÓN CON APRENDIZAJE DE MÁQUINA DE MÉTODO PARA OBTENCIÓN DE NUBES DE PUNTOS Y SU REPRESENTACIÓN EN REALIDAD VIRTUAL***, un trabajo propio, innovador, relevante e inédito y con fundamento en la facción IX del Artículo 43 del Reglamento General de Posgrados, doy mi consentimiento de que la versión final del documento ha sido revisada y las correcciones se han incorporado apropiadamente, por lo que me permito emitir el **VOTO APROBATORIO**, para que *él* pueda continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

A T E N T A M E N T E
"Se Lumen Proferre"
Aguascalientes, Ags., a *12* de *junio* de *2026*

Dr. Hermilo Sánchez Cruz
Director de *tesis*

c.c.p.- Interesado
c.c.p.- Coordinación del Programa de Posgrado

Mtro. Guillermo Domínguez Aguilar
DECANO (A) DEL CENTRO **DE CIENCIAS BÁSICAS**

_____
P R E S E N T E

Por medio del presente como ***CODIRECTOR*** designado del estudiante ***MISAEL ALEJANDRO RIVAS JUAREZ*** con ID *538387* quien realizó *la tesis* titulada: ***OPTIMIZACIÓN CON APRENDIZAJE DE MÁQUINA DE MÉTODO PARA OBTENCIÓN DE NUBES DE PUNTOS Y SU REPRESENTACIÓN EN REALIDAD VIRTUAL***, un trabajo propio, innovador, relevante e inédito y con fundamento en la facción IX del Artículo 43 del Reglamento General de Posgrados, doy mi consentimiento de que la versión final del documento ha sido revisada y las correcciones se han incorporado apropiadamente, por lo que me permito emitir el **VOTO APROBATORIO**, para que *él* pueda continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

A T E N T A M E N T E
"Se Lumen Proferre"
Aguascalientes, Ags., a *12* de *junio* de *2026*.

Dr. Osvaldo Arturo Tapia Dueñas
Co director de *tesis*

c.c.p.- Interesado
c.c.p.- Coordinación del Programa de Posgrado

Mtro. Guillermo Domínguez Aguilar
DECANO (A) DEL CENTRO **DE CIENCIAS BÁSICAS**

_____
P R E S E N T E

Por medio del presente como ***ASESOR*** designado del estudiante ***MISAEL ALEJANDRO RIVAS JUAREZ*** con ID *538387* quien realizó *la tesis* titulada: ***OPTIMIZACIÓN CON APRENDIZAJE DE MÁQUINA DE MÉTODO PARA OBTENCIÓN DE NUBES DE PUNTOS Y SU REPRESENTACIÓN EN REALIDAD VIRTUAL***, un trabajo propio, innovador, relevante e inédito y con fundamento en la facción IX del Artículo 43 del Reglamento General de Posgrados, doy mi consentimiento de que la versión final del documento ha sido revisada y las correcciones se han incorporado apropiadamente, por lo que me permito emitir el **VOTO APROBATORIO**, para que *él* pueda continuar con el procedimiento administrativo para la obtención del grado.

Pongo lo anterior a su digna consideración y sin otro particular por el momento, me permito enviarle un cordial saludo.

A T E N T A M E N T E
"Se Lumen Proferre"
Aguascalientes, Ags., a *12* de *junio* de *2026*

Dr. Julio César Ponce Gallegos
Asesor de *tesis*

c.c.p.- Interesado
c.c.p.- Coordinación del Programa de Posgrado

UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES

DICTAMEN DE LIBERACIÓN ACADÉMICA
PARA INICIAR LOS TRÁMITES DEL EXAMEN DE GRADO

POSGRADOS UAA

Fecha de dictaminación (dd/mm/aaaa): 12/06/2026

NOMBRE: MISAEL ALEJANDRO RIVAS JUAREZ **ID** 538387

PROGRAMA: MAESTRÍA EN CIENCIAS CON OPCIONES A LA COMPUTACIÓN, MATEMÁTICAS APLICADAS **LGAC (del posgrado):** Inteligencia Artificial

MODALIDAD DEL PROYECTO DE GRADO: Tesis Tradicional (X) *Tesis por artículos científicos ( ) **Tesis por Patente ( ) Trabajo Práctico ( )

TITULO: OPTIMIZACIÓN CON APRENDIZAJE DE MÁQUINA DE MÉTODO PARA OBTENCIÓN DE NUBES DE PUNTOS Y SU REPRESENTACIÓN EN REALIDAD VIRTUAL

IMPACTO SOCIAL (señalar el impacto logrado): USO DE NUBES DE PUNTOS OPTIMIZADAS EN REALIDAD VIRTUAL

INDICAR SEGÚN CORRESPONDA: **SI, NO, NA** (No Aplica)

	Elementos para la revisión académica del trabajo de tesis o trabajo práctico:
SI	El trabajo es congruente con las LGAC del programa de posgrado
SI	La problemática fue abordada desde un enfoque multidisciplinario
SI	Existe coherencia, continuidad y orden lógico del tema central con cada apartado
SI	Los resultados del trabajo dan respuesta a las preguntas de investigación o a la problemática que aborda
SI	Los resultados presentados en el trabajo son de gran relevancia científica, tecnologíca o profesional según el área
SI	El trabajo demuestra más de una aportación original al conocimiento de su área
SI	Las aportaciones responden a los problemas prioritarios del país
SI	Generó transferencia del conocimiento o tecnológica
SI	Cumple con la ética para la investigación (reporte de la herramienta antiplagio)
	El egresado cumple con lo siguiente:
SI	Cumple con lo señalado por el Reglamento General de Posgrados
SI	Cumple con los requisitos señalados en el plan de estudios (créditos curriculares, optativos, actividades complementarias, estancia, predoctoral, etc.)
SI	Cuenta con los votos aprobatorios del comité tutorial
NA	Cuenta con la carta de satisfacción del Usuario (En caso de que corresponda)
SI	Coincide con el título y objetivo registrado
SI	Tiene congruencia con cuerpos académicos
SI	Tiene el CVU de la SECIHTI actualizado
NA	Tiene el o los artículos aceptados o publicados y cumple con los requisitos institucionales (en caso de que proceda)
	****En caso de Tesis por artículos científicos publicados*** *(completar solo si la tesis fue por artículos)*
NA	Aceptación o Publicación de los artículos en revistas indexadas de alto impacto según el nivel del programa
NA	El (la) estudiante es el primer autor(a)
NA	El (la) autor(a) de correspondencia es el Director (a) del Núcleo Académico
NA	En los artículos se ven reflejados los objetivos de la tesis, ya que son producto de este trabajo de investigación.
NA	Los artículos integran los capítulos de la tesis y se presentan en el idioma en que fueron publicados
	*****En caso de Tesis por Patente***
NA	Cuenta con la evidencia de solicitud de patente en el Departamento de Investigación (anexarla al presente formato)

Con base en estos criterios, se autoriza continuar con los trámites de titulación y programación del examen de grado:

Sí X

No

FIRMAS

Elaboró:

*NOMBRE Y FIRMA DEL(LA) CONSEJERO(A) SEGÚN LA LGAC DE ADSCRIPCION: Dr. Rogelio Salinas Gutierres

** En caso de conflicto de intereses, firmará un revisor miembro del NA de la LGAC correspondiente distinto al director o miembro del comité tutorial, asignado por el Decano.*

NOMBRE Y FIRMA DEL COORDINADOR DE POSGRADO: Dra. Mariana Alfaro Gómez

Revisó:

NOMBRE Y FIRMA DEL SECRETARIO DE INVESTIGACIÓN Y POSGRADO: Dra. Iliana Ernestina Medina Ramirez

Autorizó:

NOMBRE Y FIRMA DEL DECANO: Mtro. Guillermo Domínguez Aguilar

Nota: procede el trámite para el Depto. de Apoyo al Posgrado

En cumplimiento con el Art. 24 fracción V del Reglamento General de Posgrado, que a la letra señala entre las funciones del Consejo Académico: Proponer criterios y mecanismos de selección, permanencia, egreso y titulación de estudiantes para asegurar la eficiencia terminal y la titulación y el Art. 28 fracción IX, atender, asesorar y dar el seguimiento del estudiantado desde su ingreso hasta su titulación.

Agradecimientos

En la presente, agradezco primeramente a Alejandro Rivas Anaya, María Eugenia Juárez Espinosa, Jennifer Andrei Rivas Juárez y Eivar Ulises Rivas Juárez, mi familia, por el amor y el apoyo incondicional que he tenido de su parte en todos los caminos que he recorrido durante mi vida. Sin tales elementos en mi recorrido, no habría llegado a la conclusión de una maestría en el área de la computación, por lo cual dedico la culminación de esta etapa a nombre de ellos.

Así mismo, agradezco a Hermilo Sánchez Cruz, a Osvaldo Tapia Dueñas y a Julio César Ponce Gallegos por haber formado parte de este proyecto de tesis que ha logrado llegar a una conclusión. Gracias a los aportes en conocimiento derivados de la experiencia de cada uno de ellos fue posible concluir con esta tesis.

Además, agradezco a todo aquel personal docente del área de posgrados que ha dedicado su vida y esfuerzo en la formación de las próximas generaciones tanto de profesionales como de generadores de conocimiento. De manera especial, agradezco a todo aquel docente que impartió sus conocimientos conmigo en el transcurso de mi maestría, tanto en las clases que cursé durante mis cuatro semestres como en las asesorías personales que recibí de ellos.

Por último, agradezco a la vida por las oportunidades que se presentaron a lo largo de mis veintisiete años de trayecto que me trajeron a vivir la dicha y el privilegio de haber podido llegar a este punto en la vida.

Contenidos

Índice de Figuras

Índice de Tablas

Resumen

En este manuscrito se presenta un procedimiento nuevo para la optimización de nubes de puntos basado en la reducción de la distancia de Hausdorff (métrica de error) entre la nube y el objeto 3D que representa, con la finalidad de conseguir una representación más fidedigna del contorno del objeto. Además, también se propone un entorno de realidad virtual en el cual se manipulan las nubes optimizadas. Para lograr la optimización de las nubes de puntos, se utilizan dos nubes de diferente densidad de puntos de un mismo objeto 3D, una con alta densidad (nube de puntos original) y la otra con baja densidad (nube simplificada de puntos); esto se hace para considerar las posiciones que hay en la nube original al momento de reordenar todos los puntos de la nube simplificada. La metodología para la optimización consiste en, primeramente, escalar ambas nubes de puntos para que ambas estén en el mismo rango sin perder su forma original, luego se posicionan ambas nubes en el mismo espacio para formar vecindarios de puntos en la nube original a los que se puedan ligar uno o varios puntos de la nube simplificada. Posteriormente, se reordena cada punto de la nube simplificada considerando el vecindario que le corresponde de la nube original. Para el diseño del entorno de realidad virtual, se utilizó un motor de desarrollo de videojuegos (Unreal Engine 5) en el cual se implementaron las nubes de puntos resultantes para poder ser trasladadas, rotadas, escaladas e incluso utilizadas como un objeto sólido con físicas de movimiento; todo esto con el objetivo de explorar las opciones que el motor ofrece para manipular este tipo de objetos.

Como resultado, el proceso de optimización demostró ser efectivo en la reducción del error al dar como resultado nubes de puntos optimizadas que presentan tanto una mejora visual como una reducción en la distancia de Hausdorff con respecto a las nubes simplificadas. Además, el acercamiento a Unreal Engine 5 permitió tanto entender la manera en la que funciona el motor, como ver las posibles áreas de mejoras en él al momento de trabajar con nubes de puntos; lo cual, podría derivar en propuestas de herramientas futuras para dicho motor.

Abstract

This work presents a new method for optimizing point clouds based on reducing the Hausdorff distance (error metric) between the point cloud and the 3D object it represents, with the aim of achieving a more accurate representation of the object's shape. In addition, a virtual reality environment in which the optimized point clouds can be manipulated is also proposed. To optimize point clouds, two point clouds of different densities representing the same 3D object are used: one with high density (the original cloud) and the other with low density (the simplified cloud). This is done by considering the positions in the original cloud while reordering all the points in the simplified cloud. The optimization methodology consists of, first, scaling both point clouds so that they are within the same range without losing their original shape; then, both point clouds are positioned in the same space to make neighborhoods of points in the original cloud that can be linked to one or more points in the simplified cloud. Next, each point in the simplified cloud is reordered based on its corresponding neighborhood in the original cloud. To design the virtual reality environment, a video game development engine (Unreal Engine 5) was used, in which the resulting point clouds were implemented so they could be moved, rotated, scaled, and even used as solid objects with physics-based movement; all of this was done with the goal of exploring the options the engine offers for manipulating this type of object.

As a result, the optimization process proved effective in reducing error, yielding optimized point clouds that show both visual improvement and a reduction in Hausdorff distance compared to the simplified clouds. Furthermore, the exploration of Unreal Engine 5 allowed us to understand how the engine works and to identify potential areas for improvement when working with point clouds; which could lead to proposals for future tools for that development engine.

Capítulo 1: Introducción

Desde los inicios de la computación, uno de los objetivos que se han perseguido hasta la actualidad ha sido lograr que la computadora sea capaz de replicar aquello que puede hacer el cerebro humano. Por esta razón, muchas de las áreas en el campo de la informática están destinadas a intentar replicar funcionamientos específicos que son hechos por nuestro órgano más complejo. Es aquí donde podemos encontrar dos áreas de interés en particular para este trabajo: el aprendizaje de máquina y la visión por computadora.

El aprendizaje de máquina es el campo de la informática y la inteligencia artificial que busca dotar a las computadoras de la capacidad de poder realizar tareas que son realizadas por el intelecto humano. Las aplicaciones más comunes de las herramientas de este campo son para predecir, agrupar, clasificar datos e incluso tomar decisiones a partir de datos específicos que se requieran utilizar para realizar alguna tarea específica; tal como lo haría un humano.

Por su parte, la visión por computadora es el campo de la informática que se enfoca en el procesamiento de imágenes del mundo real para extraer información de ellas, que puede ser utilizada en la toma de decisiones de un sistema informático o robótico, por ejemplo. Como tal, busca hacer que los sistemas computacionales sean capaces de distinguir formas y patrones visualmente, tal como lo haría un humano.

Cabe mencionar que estos dos campos han sido combinados para lograr tareas que anteriormente solo podían ser hechas por personas. Un ejemplo bastante conocido es el de las funciones de las redes sociales de etiquetar automáticamente a las personas que aparecen en una fotografía, o el de las funciones que incluyen las cámaras de seguridad de un negocio para reconocer las caras de las personas que trabajan en él y detectar cuando alguien no autorizado entra a una zona restringida, entre muchos otros ejemplos más.

Aunque cada uno de estos campos propone herramientas que permiten a la computadora realizar funciones que son ejecutadas por nuestro cerebro humano e incluso son combinadas para lograr actividades más avanzadas, en la actualidad el avance y el enfoque de lo que se puede y se quiere hacer con la computación ha ido evolucionando; ahora no solamente se enfoca en replicar lo que hace el cerebro humano, sino que también se está enfocando en expandir lo que nuestro cerebro puede percibir mediante estímulos creados con la misma tecnología computacional actual. Es así como llegamos a la tercera área de interés de este trabajo: la realidad virtual.

La realidad virtual se apoya en la tecnología que simula entornos tridimensionales (3D) inmersivos, en los que una persona, a través del hardware adecuado (gafas de realidad virtual, controles, entre otros), puede tanto visualizar como interactuar con entornos y objetos 3D percibiendo a través de la vista, el oído e incluso el tacto estímulos que le hacen creer al cerebro que tal realidad está ocurriendo alrededor de él. Dicha tecnología ha tenido tal desarrollo en los últimos años que ya sus aplicaciones pueden verse implementadas en varios aspectos de la vida

cotidiana, como en el consumo de videojuegos, el cine, la navegación por internet, entre otros más.

Sin embargo, la realidad virtual no se libra de la cuestión inherente que hay en todos los ámbitos en los que se transmiten datos en grandes cantidades: siempre es posible, e incluso necesario, optimizar tanto la transmisión como el almacenamiento y el procesamiento de los datos. Esto cobra más fuerza al hablar de un campo donde los datos son tanto objetos 3D como las interacciones que hay de los usuarios con los objetos (y también entre ellos). Por esta cuestión de una necesidad de optimización de los datos, las dos áreas descritas anteriormente (aprendizaje de máquina y visión por computadora) pueden encajar a la perfección en el campo de la realidad virtual para cumplir con el cometido de lograr tal optimización.

Por tal motivo, y por el hecho de que la investigación en el campo de la realidad virtual no se ha hecho esperar, es que podemos encontrar en la literatura actual propuestas que utilizan tanto herramientas del aprendizaje de máquina como herramientas del campo de la visión por computadora para poder optimizar todos estos aspectos de los entornos de la realidad virtual. Algunas de esas propuestas se basan en el uso de las nubes de puntos 3D.

Con las nubes de puntos se pueden representar desde objetos simples hasta entornos completos. Esto ya supone una ventaja en su implementación para entornos virtuales porque, como menciona Kharroubi [1], utilizar datos 3D no estructurados, como las nubes de puntos, puede conducir a un flujo de trabajo eficiente sin la necesidad de largos procesos de optimización. Por esa razón, incluso podemos ver que las nubes de puntos han tenido aplicación en el rubro de la animación [2], video [3], [4], [5], diseños de sistemas CAD [6], [7], [8] y en la creación de entornos de realidad virtual [1], [9].

Ahora, es importante mencionar que la tarea de representar un objeto 3D presenta un reto que, a la fecha de publicación de este trabajo, sigue siendo investigado, tanto por las diferentes maneras en las que puede abordarse ese tema, como por las diferentes consideraciones sobre qué es lo que se quiere priorizar. Esto es lo que encontramos al intentar responder la cuestión: ¿cuántos puntos tendrían que ser utilizados para representar un objeto 3D con una nube de puntos?

Evidentemente, a mayor densidad de puntos, habrá una representación más fidedigna de un objeto en cuestión; aunque también habrá una mayor demanda en el almacenamiento y el procesamiento del objeto representado. Pero si se utiliza una densidad de puntos muy baja, se corre el riesgo de perder características importantes del objeto representado. Por lo que decidir el número exacto de puntos para lograr un equilibrio entre una densidad baja y una representación que no comprometa la forma del contorno de un objeto, es una tarea tanto vigente como relevante.

En la literatura, podemos encontrar algunos ejemplos de diferentes enfoques que se han utilizado para llevar a cabo la tarea de simplificar nubes de puntos. Algunos de ellos, fueron propuestos por El Sayed [10] que propuso un método a partir del uso de grafos ponderados para detectar

regiones características en la nube de puntos y eliminar los vértices redundantes, E. Leal [11] que calcula vectores de características en los puntos en las nubes para identificar radios de clusterización dinámicos, y Tapia-Dueñas [12] que aplicó el proceso de voxelizar un objeto 3D para poder obtener un código de cadena que describe el contorno de cada nivel del objeto voxelizado y, con ese código obtenido, detectar los puntos clave que conformaban al nivel.

El enfoque que se propone en este trabajo considera los dos aspectos mencionados: buscar una densidad baja de puntos mientras se logra mantener e incluso mejorar la fidelidad que se hace con la nube de puntos sobre el objeto 3D que representa. Para ello, se parte de las nubes simplificadas obtenidas por el método de Tapia-Dueñas [12] para reordenar los puntos que la conforman buscando el lugar que, estratégicamente, podría usar cada punto de manera que se logre representar mejor el objeto 3D original en el que se basan. Es importante mencionar que se hace uso del método del agrupamiento jerárquico para lograr este cometido. En la literatura podemos encontrar aportes que aplican este método de aprendizaje de máquina sobre nubes de puntos de alta densidad para unirlas e incluso segmentarlas [13], [14]. En este trabajo, se aplica tal método de aprendizaje de máquina para segmentar pequeñas cantidades de puntos.

Posteriormente, para explorar la representación y manipulación de las nubes de puntos en un entorno de realidad virtual, se crea un entorno de realidad virtual en el motor de desarrollo de videojuegos Unreal Engine 5 (UE 5), que es ampliamente utilizado en el sector del entretenimiento y también ha empezado a ser utilizado en el sector de la investigación, tanto para el rubro de las nubes de puntos como el rubro de la realidad virtual. Por ejemplo, se ha utilizado para la clasificación e integración de nubes de puntos que son representadas en realidad virtual [1], en la representación de entornos urbanos realistas con nubes de puntos [15] e incluso en la evaluación de los distintos formatos de nubes de puntos que el motor puede utilizar [16].

1.1: Estructura de la tesis

Los capítulos próximos de este trabajo siguen la estructura detallada a continuación:

- **Capítulo 2: Preliminares**
  En este capítulo se detallan los fundamentos teóricos que son utilizados a lo largo de este trabajo, mencionando los puntos relevantes que abarcan las áreas de interés de este trabajo: nubes de puntos, aprendizaje de máquina y realidad virtual.
- **Capítulo 3: Optimización de nubes de puntos**
  En este capítulo se detalla la metodología que se propone para el reacomodo de los puntos en una nube de puntos de baja densidad (simplificada), partiendo de los enfoques en los que se deciden cuáles puntos de la nube simplificada serán ligados a una sección (vecindario) específica de la nube de alta densidad (original). Luego, se analiza cuál es la mejor posición que cada punto de la nube simplificada puede ocupar para volverse una

nube optimizada considerando los integrantes que conforman el vecindario de la nube original al que fue ligado.

- **Capítulo 4: Representación y manipulación de nubes de puntos en realidad virtual**
  En este capítulo se detallan tanto las herramientas utilizadas en software y en hardware, así como la manera en la que se creó, configuró y programó un proyecto de realidad virtual sobre el que se representan y manipulan las nubes de puntos. Además de ofrecer una comparación de las diferentes maneras que UE 5 permite emplear nubes de puntos 3D en un proyecto de realidad virtual, señalando tanto los beneficios como las limitantes que cada manera ofrece.
- **Capítulo 6: Conclusiones**
  En este capítulo final se mencionan tanto los aprendizajes finales del presente trabajo, como las ideas a desarrollar en un futuro próximo.

1.2 Aportaciones principales de la tesis

A lo largo de este trabajo se pueden encontrar las siguientes aportaciones principales:

- Una metodología nueva para la optimización de nubes de puntos de baja densidad basada en el análisis y el entendimiento del objetivo de reducir la distancia de Hausdorff entre dos nubes de diferente densidad de puntos. La cual derivó en la publicación de un artículo de congreso que recibió el reconocimiento del "Best Student Paper Award", así como una invitación para publicar un artículo de revista indexada.
- Una estrategia para el uso de Unreal Engine 5 en la representación y manipulación de nubes de puntos en un entorno de realidad virtual.
- Un análisis comparativo entre utilizar nubes de puntos con las herramientas nativas de Unreal Engine 5 y utilizar nubes de puntos procesadas con un software de modelado 3D (Blender).

Capítulo 2: Preliminares

A continuación, se detallan los conceptos importantes sobre los que se desenvuelve este trabajo para dar un acercamiento a los términos que se utilizan en la metodología utilizada para alcanzar los objetivos propuestos. Se parte desde las definiciones matemáticas que se utilizan, pasando a la explicación de los puntos clave en cada una de las áreas de interés en las que se enfoca este trabajo.

2.1: Definiciones y conceptos

Definición 2.1.1 La distancia euclidiana entre dos puntos 3D puede ser definida de la siguiente manera:

$$d(p_1, p_2) = \|p_1, p_2\| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad (1)$$

Donde

- p_1 y p_2 son dos puntos 3D ubicados en posiciones diferentes
- x_n, y_n y z_n son los valores de las coordenadas de cada punto.

Definición 2.1.2 La Distancia de Hausdorff entre puntos de dos nubes de puntos N_1 y N_2 puede ser definida de la siguiente manera:

$$Hau(N_1, N_2) = \max\left\{\max_{p_1 \in N_1}\left\{\min_{p_2 \in N_2}\{\|p_1, p_2\|\}\right\}, \max_{p_2 \in N_2}\left\{\min_{p_1 \in N_1}\{\|p_2, p_1\|\}\right\}\right\} \quad (2)$$

Donde $\|p_1, p_2\|$ denota la distancia euclidiana entre los puntos p_1 y p_2.

Definición 2.1.3 La distancia máxima de error entre dos nubes de puntos N_1 y N_2 puede ser definida de la siguiente manera:

$$\Delta_{max}(N_1, N_2) = \max_{p_1 \in N_1}\left\{\min_{p_2 \in N_2}\{\|p_1, p_2\|\}\right\} \quad (3)$$

Donde $\|p_1, p_2\|$ denota la distancia euclidiana entre los puntos p_1 y p_2.

Definición 2.1.4 La distancia promedio de error entre dos nubes de puntos N_1 y N_2 puede ser definida de la siguiente manera:

$$\Delta_{avg}(N_1, N_n) = \frac{1}{\#N_1} \sum_{p_1 \in N_1} \|p_1, N_n\| \quad (4)$$

Donde $\|p_1, p_2\|$ denota la distancia euclidiana entre los puntos p_1 y p_2, y $\#N_1$ denota la cantidad de puntos que conforman la nube N_1.

Definición 2.1.5 El escalamiento min-max es una técnica que permite procesar datos numéricos de manera que permite ajustar el rango en el que se encuentran los mismos. Este escalamiento se define de la siguiente manera:

$$V_{escalado} = \left[\left(\frac{V - V_{min}}{V_{max} - V_{min}}\right)(max - min)\right] + min \quad (5)$$

Donde:

- $V_{escalado}$ es el valor resultante del escalamiento
- V es el valor que se quiere escalar
- V_{min} y V_{max} son los valores mínimos y máximos del conjunto de datos que se quieren escalar
- min y max son los valores a los cuales se quiere escalar el conjunto de datos original

Definición 2.1.6 La media poblacional de un conjunto de datos es un estadístico poblacional que puede ser definido de la siguiente manera:

$$\mu = \frac{\sum_{i=1}^{N} x_i}{N} \quad (6)$$

Donde:

- N es el número de elementos totales en el conjunto de datos
- x es el valor del conjunto de datos en la posición i

Definición 2.1.7 La desviación estándar poblacional de un conjunto de datos es un estadístico poblacional que puede ser definido de la siguiente manera:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N}} \quad (7)$$

Donde:

- N es el número de elementos totales en el conjunto de datos
- x es el valor del conjunto de datos en la posición i
- μ es la media poblacional del conjunto de datos

Definición 2.1.8 El escalamiento por estandarización es una técnica que permite procesar datos numéricos de manera que son ajustados para que sigan una distribución normal entre ellos, con una media (μ) de 0 y una desviación estándar (σ) de 1, al aplicar la siguiente fórmula:

$$V_{escalado} = \frac{V - \mu}{\sigma} \quad (8)$$

Donde:

- $V_{escalado}$ es el valor resultante del escalamiento
- V es el valor que se quiere escalar
- μ es la media poblacional del conjunto de datos que se quieren escalar
- σ es la desviación estándar poblacional del conjunto de datos que se quieren escalar

2.2: Nubes de puntos 3D

La representación de un objeto 3D mediante una nube de puntos consiste en utilizar un conjunto de puntos 3D ordenados de tal manera que se busque recrear el contorno de dicho objeto 3D. Al tratarse de objetos discretos que representan el contorno de un volumen continuo, se tiene que tomar en cuenta la fidelidad que puede estar manteniéndose, o perdiéndose, al utilizar una nube de puntos con respecto al objeto que intenta preservar su forma. Como se ha mencionado anteriormente, esto depende principalmente de la cantidad de puntos que se emplee en ella.

Para ello es conveniente utilizar métricas de error que indiquen cuán alejada puede estar la representación del objeto 3D que se está haciendo con la nube de puntos. Al momento de calcular estas métricas de error se puede trabajar directamente con el objeto 3D en cuestión o también con una nube de puntos del mismo objeto con una alta densidad de puntos en la que no se pierda alguna característica geométrica del mismo.

Aunque también debe considerarse que la medición de una métrica de error es susceptible al rango en el que están las nubes de puntos; si se utiliza una nube de alta densidad para saber el error que tiene una nube de baja densidad y ambas están en un rango que difiere, el error que se intenta calcular en la nube de baja densidad será mayor al que realmente es. Por ello, se deben de considerar las diferentes maneras que hay para transformar una nube de puntos sin alterar su forma al escalar su tamaño a un rango específico.

A continuación, se detallan las métricas de error utilizadas al trabajar con dos nubes de puntos (alta densidad y baja densidad) y se hace una comparación de las dos transformaciones que se pueden hacer para escalarlas.

Métricas de error

- **Distancia de Hausdorff**

Teniendo en cuenta la fórmula 2, se puede apreciar que lo primero en ser realizado para obtener el valor de la distancia de Hausdorff es obtener el conjunto de las distancias euclidianas mínimas de cada punto de la nube de puntos N_1 con respecto a los puntos que hay en la nube N_2, para elegir la distancia máxima que se encuentre en ese conjunto. Luego, se obtiene el conjunto de las distancias euclidianas mínimas de cada punto de la nube N_2 con respecto a los puntos en la nube N_1, para elegir la distancia máxima de ese segundo conjunto. La distancia de Hausdorff está determinada por el valor mayor entre las distancias máximas obtenidas entre ambos conjuntos.

Visto de una manera visual, se representa con un ejemplo bidimensional (2D) de las figuras 1-3. En ellas, los puntos de la nube N_1 son representados con círculos azules, los puntos de la nube N_2 son representados con círculos verdes y la distancia mínima de cada punto en N_1 con respecto a los puntos de N_2 está representada con una flecha roja. La figura 1 representa las distancias mínimas encontradas de los puntos en la nube N_1 a la nube N_2, en la figura 2 se representa lo mismo, pero ahora de los puntos de la nube N_2 hacia los puntos de la nube N_1. La figura 3 muestra las distancias máximas encontradas de ambas búsquedas de distancias mínimas, en ella se puede apreciar que, si se mueve alguno de los puntos que conforman esas distancias máximas, la distancia de Hausdorff cambiará, pero existe la posibilidad de que ahora otros puntos conformen esas distancias máximas.

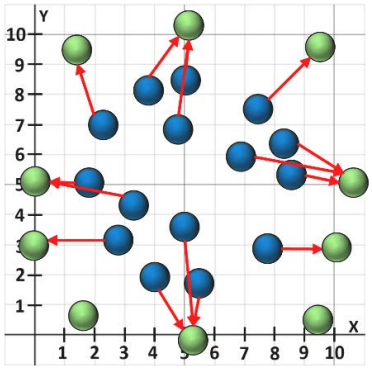


Figura 1 – Distancias mínimas de los puntos en N_1 hacia N_2

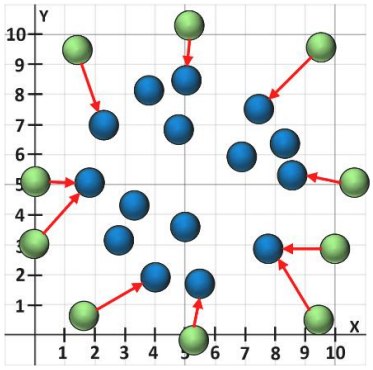


Figura 2 – Distancias mínimas de los puntos en N_2 hacia N_1

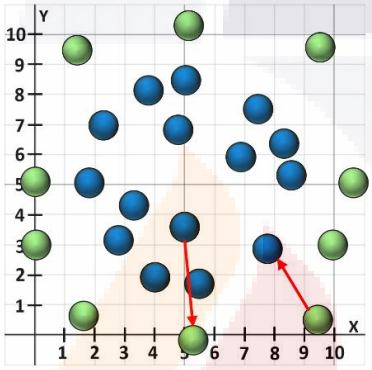


Figura 3 – Distancias máximas entre los puntos de ambas nubes

Esta métrica nos puede ayudar a visualizar el error que hay entre dos nubes de puntos ya que, entre mayor sea la distancia de Hausdorff, mayor será la diferencia que haya entre una nube de puntos y otra.

- **Distancia máxima y distancia promedio de error**

Como se puede apreciar en la fórmula 3 (distancia máxima) y fórmula 4 (distancia promedio), el cálculo de estas métricas solo va a considerar las distancias mínimas que se pueden obtener de los puntos de la nube N_1 hacia los puntos de la nube N_2; visto de forma visual, las distancias que se representan con flechas rojas en la figura 1.

En el caso de la distancia máxima, la distancia mayor que haya en esas distancias de los puntos en N_1 hacia los puntos de N_2 será la que determine su valor. En el caso de la distancia promedio, la suma de todas esas distancias de los puntos en N_1 hacia los puntos de N_2 dividida sobre el número de puntos en N_1 será lo que determine su valor.

Cabe mencionar que ambas métricas pueden encontrarse en algunos trabajos de la literatura actual, como en [10] y [11]; sin embargo, la distancia máxima del error también es referida como la distancia de Hausdorff en estos mismos trabajos.

Ambas métricas pueden ayudar a ver cuán alejada puede estar la representación de una nube de puntos con respecto a otra; sin embargo, es un poco sesgado solo utilizar una dirección (N_1 a N_2) para considerar cuán alejados están dos conjuntos de puntos, ya que siempre existe la posibilidad de que estos errores puedan ser más grandes si se considera la dirección opuesta (N_2 a N_1).

Transformaciones sobre nubes de puntos (escalamiento)

- **Escalamiento min-max**

Como se puede apreciar en la fórmula 5, hay dos rangos de datos con los que se trabaja para poder llevar a cabo el escalamiento: el rango original de los datos y el rango al que se quieren escalar.

Un ejemplo de aplicación de este escalamiento es el siguiente. Supóngase que se necesita escalar el rango de los puntos mostrados en la figura 4 para que estén dentro de un rango de [0:2] en ambos ejes.

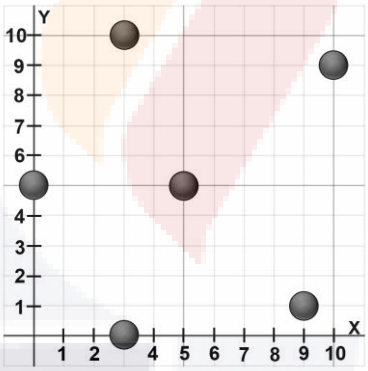


Figura 4 - Ejemplo de puntos a escalar con min-max

De los puntos mostrados en la figura 4, se extraen los siguientes conjuntos con los valores de cada eje en las coordenadas de cada punto; de manera que el elemento X_1 y Y_1 pertenecen a las coordenadas del punto ubicado en (0,5) y los demás elementos de los conjuntos forman las coordenadas de los demás puntos.

$$X = \{0, 3, 3, 5, 9, 10\}, \qquad Y = \{5, 0, 10, 5, 1, 9\}$$

Ahora, nótese que el rango en el que se encuentran los datos del conjunto X es de $[0:10]$, por lo que, al momento de sustituir valores en la fórmula 5, V_{min} adopta el valor de 0 y V_{max} el valor de 10. Además, nótese que el conjunto Y también está en este rango, por lo que de igual manera considera los mismos valores para V_{min} y V_{max}, esto es importante considerarlo ya que cada escalamiento debe considerar los rangos originales de cada conjunto de datos para determinar adecuadamente los valores a utilizar para min y max en la fórmula 5. En este caso, al tener los mismos valores en los rangos de ambos conjuntos de datos originales, se puede emplear los valores 0 y 2 para min y max respectivamente en el escalamiento de ambos conjuntos.

De esta manera, y sustituyendo los valores en la fórmula 4 considerando los valores de X_1 y Y_1 para V y así obtener sus valores escalados Xe_1 y Ye_1, se tiene que:

$$Xe_1 = \left[\left(\frac{0-0}{10-0}\right)(2-0)\right] + 0 = [(0)(2)] + 0 = 0$$

$$Ye_1 = \left[\left(\frac{5-0}{10-0}\right)(2-0)\right] + 0 = [(0.5)(2)] + 0 = 1$$

Repitiendo el mismo procedimiento por cada coordenada de cada punto, se obtiene que los conjuntos de valores escalados X_e y Y_e quedan de la siguiente manera:

$$X_e = \{0,0.6,0.6,1,1.8,2\}, \qquad Y_e = \{1,0,2,1,0.2,1.8\}$$

El resultado visual del escalamiento se muestra en la figura 5.

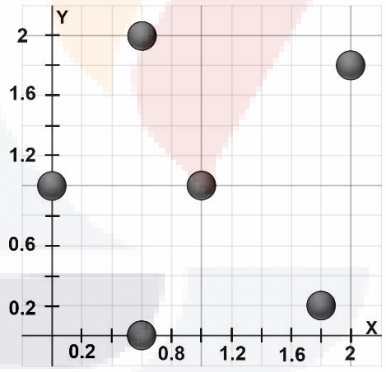


Figura 5- Resultado del primer escalamiento min-max

Sin embargo, en este ejemplo de escalamiento ambos conjuntos de coordenadas originales tenían el mismo rango, por lo que se puede aplicar el mismo rango objetivo de escalamiento sin alterar su naturaleza. Es importante considerar que, si cada conjunto de datos posee un rango distinto al de los demás, cada conjunto de datos deberá tener un rango diferente de escalamiento que sea proporcional a su rango original para mantener la naturaleza en los datos.

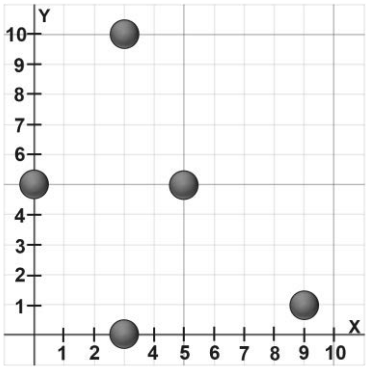


Figura 6 - Segundo ejemplo de escalamiento min-max

Para visualizarlo, si se toma al ejemplo de la figura 4 y se elimina un punto con la coordenada $(10,9)$, como se muestra en la figura 6, de manera que el rango del conjunto X queda comprendido de $[0:9]$ y el rango del conjunto Y se mantiene igual de $[0:10]$, pero se requiere que los puntos de igual manera estén en el rango $[0:2]$ y se considera los valores 0 y 2 para min y max en los escalamientos de los conjuntos X y Y, la naturaleza de los datos se verá afectada. Si bien, los valores del conjunto escalado Y_e se mantendrán igual, pero los valores del conjunto X_e mostrarán un ligero cambio:

$$X_e = \{0, 0.667, 0.667, 1.11, 2\}$$

Lo cual hace que los puntos resultantes sufran un ligero desplazamiento en el eje X, lo que equivale a alterar la naturaleza original de los datos (figura 7).

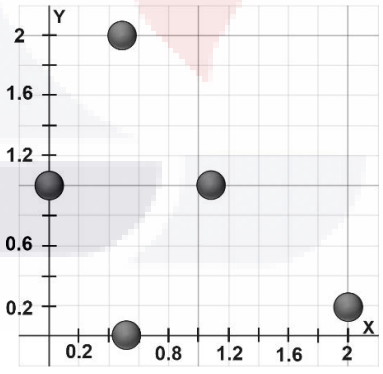


Figura 7 - Resultado de escalamiento sin aplicar rangos proporcionales

Ahora, lo que se debe de hacer en estos casos es buscar rangos objetivos de escalamiento que mantengan las proporciones de cada conjunto de datos. Una manera de lograrlo es considerar el eje con el mayor desplazamiento para ver cómo será la proporción respecto al rango que tiene y

el que tendrá. En este ejemplo, el rango del conjunto Y tiene una longitud de 10, ya que está comprendido en $[0, 10]$, y el rango del conjunto X tiene una longitud de 9; por lo que se partirá con el análisis del conjunto Y. De este modo, se puede asumir que reducirá 5 veces la longitud del rango de Y; ya que, al aplicar con el nuevo rango de [0:2], la longitud final será de 2. Por lo tanto, si se reduce 5 veces la longitud de recorrido de X, que es de 9, para mantener la naturaleza de los datos, entonces la nueva longitud deberá de ser 1.8; dicho de otras palabras, del rango $[0: 1.8]$. De esta manera, los valores obtenidos para el nuevo conjunto X_e serían los siguientes:

$$X_e = \{0,0.6,0.6,1,1.8\}$$

Como se puede apreciar en figura 8, al fijar adecuadamente el rango proporcional para cada conjunto de datos, se logra respetar la naturaleza original de los mismos.

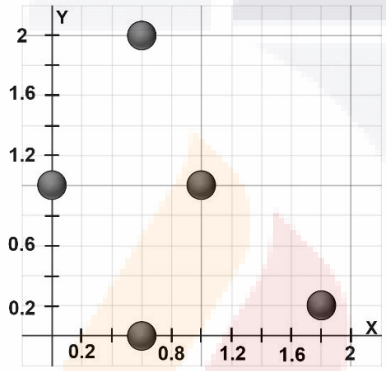


Figura 8 - Resultado de escalamiento aplicando rangos proporcionales

- **Escalamiento por estandarización**

Como se puede apreciar en la fórmula 8, para trabajar con este tipo de escalamiento, se deben considerar los descriptores estadísticos conocidos como la media poblacional y la desviación estándar poblacional de los datos que se están escalando por estandarización. Retomando el ejemplo propuesto en la figura 4, se obtiene que los descriptores estadísticos de esos datos son:

$$\mu_x = 5, \qquad \sigma_x = 3.5119$$

$$\mu_y = 5, \qquad \sigma_y = 3.6968$$

recordando que de los conjuntos de datos X y Y se obtienen los descriptores estadísticos descritos en las fórmulas 6 y 7. Cabe mencionar que se utilizan los estadísticos poblacionales dado que, en la naturaleza de estos casos, solo se considera como población los puntos utilizados.

De esta manera, y sustituyendo los valores en la fórmula 8 considerando los valores de X_1 y Y_1 para V y así obtener sus valores escalados Xe_1 y Ye_1, se tiene que:

$$Xe_1 = \frac{0-5}{3.5119} = -1.424$$

$$Ye_1 = \frac{5-5}{3.6968} = 0$$

Repitiendo el mismo procedimiento por cada coordenada de cada punto, se obtiene que los conjuntos de valores escalados X_e y Y_e quedan de la siguiente manera:

$$X_e = \{-1.424, -0.569, -0.569, 0, 1.139, 1.424\}$$

$$Y_e = \{0, -1.353, 1.353, 0, -1.082, 1.082\}$$

El resultado visual del escalamiento se muestra en la figura 9.

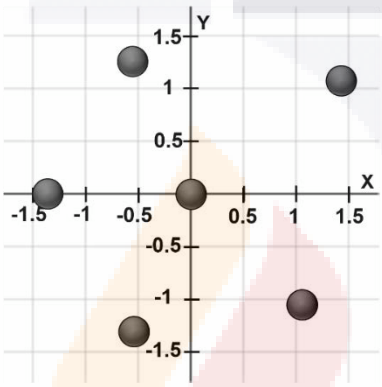


Figura 9 – Resultado del escalamiento por estandarización

Como puede apreciarse en la figura 9, los puntos no conservan la misma naturaleza que antes del escalamiento en sus nuevas coordenadas. Lo cual se puede corroborar analizando los datos obtenidos en los conjuntos X_e y Y_e, ya que en ellos se puede apreciar que la longitud en el rango de los nuevos valores del eje x es de 2.847 y que la longitud en los nuevos valores del eje y es de 2.435, lo cual, deja ver que los datos ya no tienen la misma longitud como antes del escalamiento.

2.3: Aprendizaje de máquina

Aunque el aprendizaje de máquina tiene una gran variedad de métodos, solo se hace énfasis en las técnicas de agrupamiento no supervisado ya que esas son las más adecuadas para poder proporcionar una segmentación sobre datos que no han sido clasificados ni seccionados con anterioridad, como lo son las nubes de puntos 3D. Además, solo se consideran aquellas que no

consideran aspectos específicos como probabilidades, distribuciones o densidades en los datos, ya que las nubes de puntos no los consideran.

Técnicas de agrupamiento no supervisado

- **K-means**

Como se menciona en [17], esta técnica de agrupamiento categoriza los datos, o en este caso los puntos en una nube de puntos, de manera que los divide en *K* clústeres considerando la distancia, generalmente euclidiana, que hay entre los puntos y los centroides de los clústeres donde fueron asignados. Para lograr esto, se siguen dos pasos principales:

I. Inicializar *K* centroides: Mediante una selección aleatoria se inicializan los k centroides en los que se segmentarán los puntos de una nube.
II. Asignación de centroides: Se asignan los puntos de una nube a su centroide más cercano en función de la distancia, generalmente euclidiana. Luego se calcula la media de todos los puntos para cada clúster y se reasigna el centroide del clúster. Este paso se repite hasta que se haya alcanzado el número máximo de iteraciones, o hasta que las posiciones de cada centroide hayan alcanzado una convergencia.

En la figura 10 se presenta un ejemplo de puntos 2D que se busca segmentar en tres clústeres (*K=3*). En la figura 11 se muestra el resultado visual de aplicar este agrupamiento, en ella, una cruz representa la posición final del centroide de cada clúster y, a su vez, tanto los integrantes como el centroide de cada clúster son representados por un color diferente.

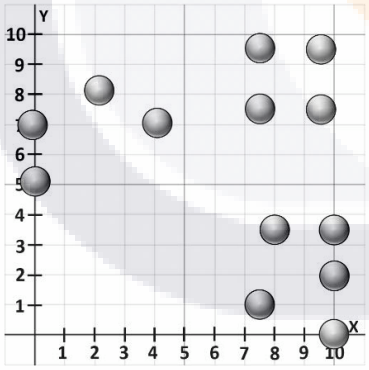


Figura 10 – Ejemplo de puntos a segmentar con K-means

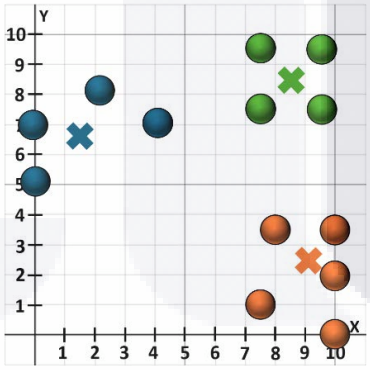


Figura 11 – Clústeres creados con K-means mostrando sus centroides

Cabe mencionar que, de igual manera mencionado en [17], este método de agrupamiento es fácilmente escalable con conjuntos de datos grandes ya que su funcionamiento está diseñado con un enfoque iterativo computacionalmente simple, aunque es sensible a valores atípicos.

En la literatura, se puede encontrar que esta técnica de agrupamiento se ha empleado con las nubes de puntos para trabajos como:

- Un sistema para comprender las formas de relieve usando filtrado de nubes de puntos basado en K-means para segmentar la vegetación del terreno digital [18].
- Un método para simplificación adaptativa de nubes de puntos densas obtenidas con dispositivos de escaneo que aplica K-means y desviaciones vectoriales para segmentar tales nubes [19].
- Un método para la extracción automatizada de discontinuidades aplicando una segmentación con K-means en nubes de puntos de rocas obtenidas con escaneo láser [20].
- Un método de segmentación de nubes de puntos de plantas de maíz basado en agrupamiento euclídeo y K-means [21].
- Un método para identificación de clases semánticas complejas en nubes de puntos de escenas 3D usando K-means para agrupar vectores en clústeres [22]

- **Agrupamiento jerárquico**

En [23] se define el agrupamiento jerárquico como una técnica de aprendizaje de máquina que agrupa objetos, o en este caso puntos, en una jerarquía de clústeres. Esto puede ser de dos modos: aglomerativo, en la que cada punto inicia siendo un clúster y se van uniendo aquellos elementos que son más cercanos para reducir el número de clústeres mientras se obtienen agrupaciones con más integrantes; y divisivo, en la que todos los puntos comienzan siendo un solo clúster y se va dividiendo los elementos más alejados en clústeres de manera que aumenta el número de clústeres mientras se obtienen agrupaciones con menos integrantes.

Una manera de visualizar el orden en el que los puntos se van uniendo, o dividiendo, es con el dendrograma, el cual es un diagrama en forma de árbol en donde se representan los elementos a clusterizar en el eje horizontal y las distancias que se manejan en el conjunto de datos en el eje vertical. En la figura 12 se muestra un conjunto de puntos que se pretende segmentar, y en la figura 13 se muestra el dendrograma que resulta al emplear el agrupamiento jerárquico.

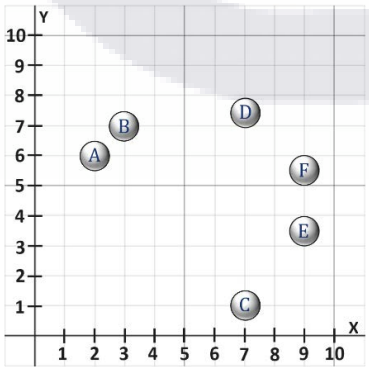


Figura 12 – Puntos a segmentar con agrupamiento jerárquico

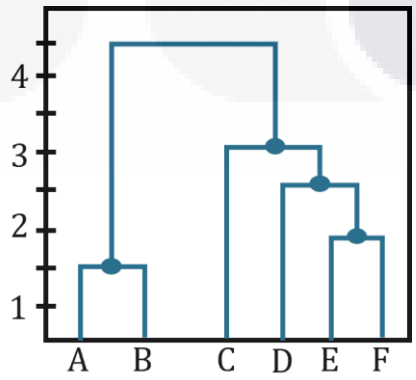


Figura 13 – Dendrograma de la agrupación

Viendo el dendrograma de la figura 13 y considerando que se aplica un agrupamiento aglomerativo para segmentar los puntos, se puede apreciar que la altura en la que está el enlace entre dos elementos representa la distancia a la que los mismos se encuentran para ser unidos. Por lo que, si analizamos el funcionamiento del agrupamiento, el algoritmo primero busca los elementos más cercanos, y se encuentra que los puntos *A* y *B* son los más cercanos con una distancia de 1.5, por lo que serán los primeros en ser unidos en un clúster; luego, los puntos *E* y *F* serían los próximos más cercanos con una distancia de 2, y formarán otro clúster; después, los elementos más cercanos serán el clúster *E-F* con el punto *D* a una distancia de 2.5, por lo que se unirá el punto *D* a este clúster, y así sucesivamente hasta llegar al punto en el que todos los puntos terminarían formando parte del mismo clúster.

Como se menciona en [24], hay diferentes estrategias para determinar cómo se considera la distancia entre los clústeres; a estas estrategias se les denomina "métodos de enlace" y los más comunes son los siguientes:

- **Enlace simple**: la distancia entre los clústeres es el mínimo de las distancias entre cualquier par de puntos del clúster 1 y el clúster 2.
- **Enlace completo**: la distancia entre dos clústeres es el máximo de las distancias entre cualquier par de puntos del clúster 1 y el clúster 2
- **Enlace promedio**: la distancia entre dos clústeres es el promedio de las distancias entre cualquier par de puntos del clúster 1 y el clúster 2.
- **Enlace del centroide**: la distancia entre dos clústeres es la distancia entre el centroide del clúster 1 y el centroide del clúster 2

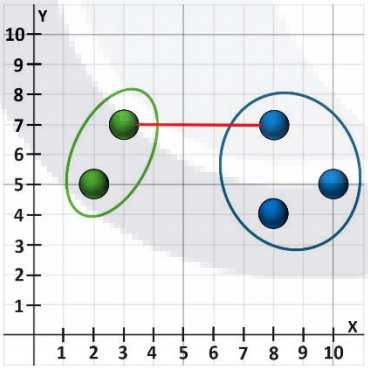


Figura 14 – Método de enlace simple

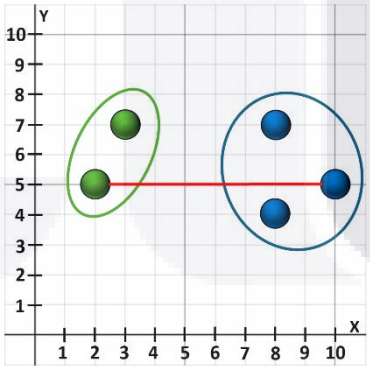


Figura 15 – Método de enlace completo

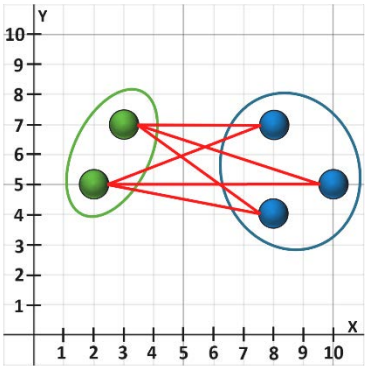


Figura 16 – Método de enlace promedio

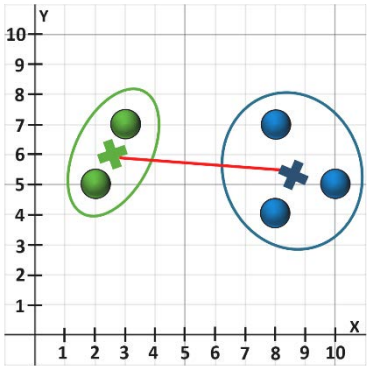


Figura 17 – Método de enlace del centroide

En las figuras 14 a 17 se muestra un ejemplo de cada método de enlace, mostrando con una línea roja cuál sería la distancia, o distancias, que el algoritmo consideraría entre los clústeres para cada método.

Como se puede apreciar, el agrupamiento jerárquico implica mucho más procesamiento y cálculo que el método K-means, por lo que suele ser más costoso, computacionalmente hablando, en cuanto a conjuntos de datos grandes.

En la literatura, se puede encontrar que esta técnica de agrupamiento se ha empleado con las nubes de puntos para trabajos como:

- Un nuevo enfoque de método de enlace en la aplicación de la agrupación jerárquica de nubes de puntos al enlazar los clústeres considerando tanto la distancia entre dos puntos como las diferencias de vectores normales en esos puntos [25].
- Un método que une nubes de puntos de planos 3D obtenidos con sensores Kineck para obtener planos compuestos de varias tomas de una manera rápida [26].
- Una propuesta de representación de nubes de puntos invariante de rotación que utiliza agrupación jerárquica para explorar la estructura geométrica presente en las nubes de puntos [27].
- Un método que usa agrupamiento jerárquico que emplea tanto clasificación de formas como reasignación de valores atípicos para segmentar nubes de puntos de entornos urbanos con el fin de identificar las estructuras que componen los edificios [28].
- Una propuesta para segmentar nubes de puntos coloreadas que se basa en considerar tanto el color como la información espacial para conseguir una segmentación de objetos en interiores de edificios [29].

2.4: Realidad Virtual

Para desarrollar entornos de realidad virtual se necesitan tres componentes clave: un motor de desarrollo, los modelos 3D a utilizar (en este caso las nubes de puntos) y el hardware (gafas de realidad virtual) para poder manipular aquello que se quiera programar en dichos entornos.

En cuanto a los motores de desarrollo, principalmente son utilizados aquellos destinados al rubro de los videojuegos, aunque, como se detalla más adelante, no se especializan solamente en esta rama y, aunque permiten desarrollar entornos virtuales, cada uno tiene su enfoque que lo hace adecuado dependiendo de lo que se busque desarrollar.

En cuanto a los modelos 3D, se debe de considerar que, como las nubes de puntos se manejan en diversos formatos como ply, xyz, entre otros no muy conocidos, es probable que los motores de desarrollo no tengan las herramientas suficientes para trabajar con esos formatos, pero aquí se puede utilizar una alternativa para transformar el formato de una nube de puntos: utilizar un software de modelado 3D para crear una representación de las nubes en un formato diferente.

En cuanto a las gafas de realidad virtual, se pueden encontrar diferentes opciones que se pueden usar al desarrollar entornos virtuales en conjunto con los motores de desarrollo, aunque solo se consideran aquellas opciones que tienen más soporte para la integración con los motores de desarrollo.

A continuación, se hace la explicación de las opciones más utilizadas en cuanto a motores de desarrollo, software de modelado 3D y gafas de realidad virtual.

Motores de desarrollo de entornos virtuales

- **Unity**

En [30] se define Unity como un motor de desarrollo de videojuegos y aplicaciones inmersivas creado por la fundación Unity Technologies. En este motor se pueden utilizar librerías predeterminadas para simulación de físicas, creación de objetos, programación de colisiones, entre otras cosas; además de que permite el desarrollo de librerías nuevas a través del uso del lenguaje C#.

Unity es reconocido por su capacidad de crear aplicaciones multiplataforma, por lo cual, se pueden desarrollar aplicaciones y/o videojuegos para dispositivos móviles, computadoras de escritorio, consolas de videojuegos, navegadores web, entre otros. Sin mencionar que también es conocido por ser una herramienta de desarrollo con una interfaz intuitiva que permite una agilidad para ser aprendida de manera ágil y así desarrollar proyectos en un tiempo rápido, sin mencionar que no requiere de altos requerimientos computacionales para funcionar [31].

En la literatura, podemos encontrar que Unity ha sido utilizado en el campo de la investigación con relación a la realidad virtual y las nubes de puntos para:

- ❖ Creación de librería para visualizaciones e interacciones con nubes de puntos [32]

- Comparar y clasificar nubes de puntos masivas [1]
- Creación de aplicación interactiva para visualizar nubes de puntos en tiempo real de ejecución [33]
- Creación de motor de renderizado para nubes de puntos independiente para implementar en aplicaciones [34].

- **Unreal Engine 5 (UE 5)**

Como se expone en [35], UE 5 es un motor de desarrollo 3D desarrollado por la empresa de videojuegos Epic Games que sirve para visualizar diseños, o crear tanto experiencias cinematográficas como videojuegos; por lo mismo, es que esta herramienta tiene funcionalidades avanzadas con respecto a la calidad de materiales, de iluminación, de simulación de físicas, de efectos visuales, entre otros.

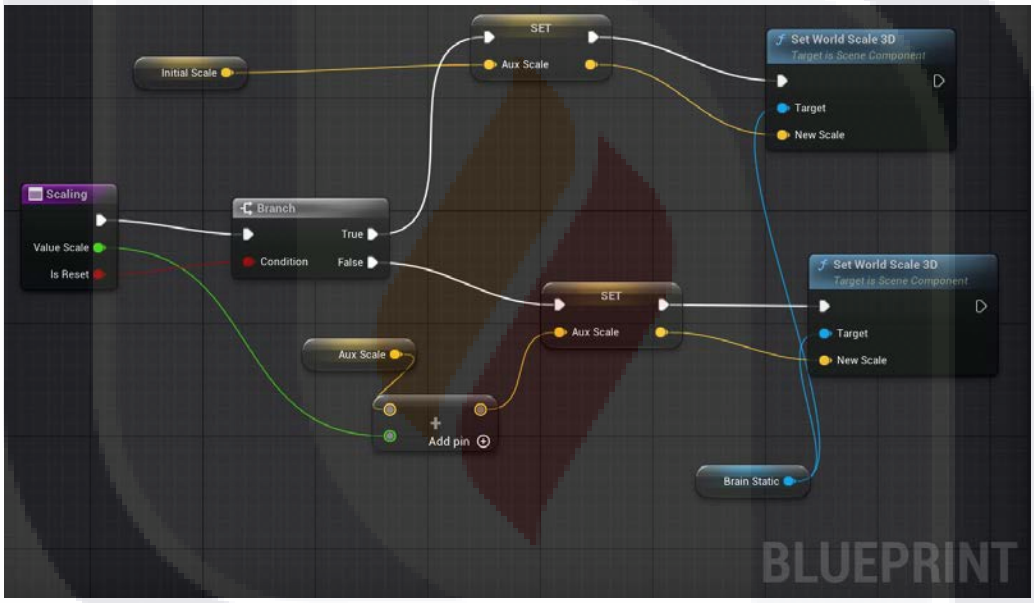


Figura 18 – Programación por Blueprints en UE 5

Cabe mencionar que en este motor de desarrollo se puede trabajar con un entorno visual de programación llamado "Blueprints", con el cual se puede programar cualquier funcionalidad que se necesite tanto para la entrada de comandos a través de controles, como las interacciones, las funcionalidades y los efectos que pueden tener los componentes de un entorno virtual. En la figura 18 se muestra un ejemplo de este sistema de programación que representa métodos, instancias de objetos, entre otras cosas como nodos. Aunque también es importante mencionar que este motor es lo suficientemente robusto como para permitir tanto modificar las clases existentes que se manejan a través de blueprints como para desarrollar nuevas, en lenguaje C++, a conveniencia de los desarrolladores que trabajen con él.

Cabe resaltar que, como se menciona en diversos foros [31], [36], las características anteriormente mencionadas de UE 5 lo hacen una opción bastante robusta para crear experiencias virtuales que requieran una detallada funcionalidad física y/o visual, pero por esta misma razón es que esta herramienta demanda tanto un poder computacional intermedio-alto como una curva de aprendizaje más prolongada, en comparación con Unity.

En la literatura, podemos encontrar que UE 5 ha sido empleado para el rubro de la investigación utilizando nubes de puntos en varios aportes, algunos ejemplos son:

- Creación de plugin nativo para visualizaciones e interacciones con nubes de puntos (transformaciones básicas) [37]
- Creación de aplicación para visualizar la digitalización de árboles en la representación de entornos urbanos realistas con nubes de puntos [38]
- Creación de aplicación para visualización y percepción de interiores en situaciones de crisis con nubes de puntos [39]
- Creación de aplicación para analizar cambios en nubes de puntos de serie temporal de grandes cantidades de puntos para extraer información de ellas [40]

Software de modelado 3D

- **Autodesk Maya**

Tomando en cuenta la información de [41], el software Maya, perteneciente a la familia de programas de diseño Autodesk, es un software de modelado 3D con herramientas que permiten tanto crear modelos detallados como entornos realistas, ya que incorpora efectos y simulaciones de texturas avanzadas. Además, permite de igual manera crear animaciones con los modelos 3D, e incluso los entornos, que se diseñen ahí. Todo esto hace que esta herramienta sea utilizada por empresas grandes de videojuegos, e incluso empresas productoras de películas animadas.

Como se menciona en algunos foros [42], esta herramienta es efectiva cuando se quiere modelar objetos con animación avanzada, de gran tamaño, cuando se necesite de herramientas de rigging complejas o cuando se le quiera añadir una simulación física avanzada. Sin embargo, cabe mencionar que esta herramienta necesita pagar periódicamente una licencia para poder utilizarse.

En la literatura se puede encontrar que Maya ha sido utilizado en conjunto con los motores de desarrollo anteriormente mencionados para diversas publicaciones. Por ejemplo:

- En un libro sobre la visualización de diseños 3D inmersivos, con Maya y Unreal Engine 4 [43]
- En un estudio sobre el análisis comparativo de producción de CGI de Maya con Unreal Engine [44]
- En un estudio sobre el diseño de software interactivo en realidad virtual, usando Maya para la parte del modelado y Unity para programar el entorno de realidad virtual [45]

- En un estudio sobre la creación de videojuegos 3D usando Maya con Unity [46]

- **Blender**

Blender es un software de modelado 3D de código abierto desarrollado por Blender Foundation que también ofrece herramientas para crear modelos y entornos 3D con el uso de texturas, efectos y simulaciones; así como herramientas para hacer animación, rigging, efectos visuales, edición de video y una API de programación en Python para desarrollar herramientas especializadas a la necesidad de los usuarios [47].

Todas estas características hacen que Blender sea utilizado por personas independientes que necesiten modelar objetos 3D para animar, crear videojuegos, entre otras opciones; y, como se menciona en [42], ha crecido en popularidad por su comunidad, y su capacidad de adquirir innovaciones nuevas de forma rápida al permitir el desarrollo de funcionalidades por parte de los usuarios, las cuales a veces terminan siendo plugins integrados a Blender en versiones posteriores.

En la literatura, a diferencia de Maya, se puede encontrar que Blender ha sido utilizado tanto en el rubro de las nubes de puntos como en conjunto con los motores de desarrollo anteriormente mencionados para diversas publicaciones. Por ejemplo:

- En un estudio sobre la creación de un simulador de escáner láser terrestre para producir nubes de puntos a partir de modelos 3D en tiempo real, que fue diseñado como un complemento para Blender [48]
- En un estudio sobre la manipulación de mayas de nubes de puntos que se puede hacer sobre paisajes urbanos con las herramientas de procesamiento de Blender [49]
- En un estudio sobre simulación de telas 3D utilizando nubes de puntos con las herramientas de Blender [50]
- En un estudio sobre el modelado 3D y la visualización 3D de un campus virtual usando técnicas de optimización para el procesamiento de los datos usando Unity para la visualización y Blender para parte del preprocesamiento de datos [51]
- En un estudio sobre desarrollo de videojuegos de Unity y Blender [52] y Unreal Engine con Blender [53]
- En un estudio sobre la integración de modelos fotogramétricos de ciudades 3D, haciendo uso de Blender para escalar los modelos de las ciudades y Unreal Engine para programar un entorno de realidad virtual para explorar los modelos [54]

Gafas de realidad virtual

- **Pico y Meta Quest**

Tanto las gafas Pico, que son pertenecientes a la compañía china ByteDance, como las gafas Meta Quest, pertenecientes a la compañía estadounidense Meta, son gafas de realidad virtual y realidad mixta, consideradas de gama media, que están enfocadas principalmente al uso de videojuegos y entretenimiento [55], aunque ambas gafas también son utilizadas en el ámbito del desarrollo para realidad virtual y/o mixta [56], [57]. Como puede apreciarse en los blogs para desarrolladores de ambas gafas, pueden ser utilizadas en proyectos independientes en conjunto con Unity y Unreal Engine y ambas tienen soporte para poder exportar proyectos como aplicaciones ejecutables desde las mismas gafas [56], [57].

Es posible encontrar estudios que hicieron uso de algunas, o ambas, de estas gafas para desarrollar entornos virtuales, o incluso comparar el rendimiento entre ellas. Por ejemplo:

- Un estudio que presenta un sistema basado en Unity sobre la simplificación de las animaciones de los movimientos de las manos virtuales usando ambas gafas [58]
- Una tesis de licenciatura dedicada a la comparación de ambas gafas en su implementación para aplicaciones de realidad aumentada [59]
- Un estudio para la evaluación de la precisión de seguimiento para rastrear los movimientos de los usuarios en grandes espacios usando ambas gafas [60]
- Un estudio sobre la mejora de la experiencia de usuario y la usabilidad automotriz con un entorno de realidad virtual desarrollado con Unreal Engine utilizando las gafas Pico [61]
- Un estudio sobre la proposición de una aplicación de realidad mixta para identificar dispositivos del internet de las cosas y representar su posición con nubes de puntos usando las gafas Meta Quest [62]
- Un estudio sobre una aplicación diseñada en Unity para el procesamiento y filtrado espacial interactivo en tiempo real de nubes de puntos empleando las gafas Meta Quest [63]

Capítulo 3: Optimización de nubes de puntos

A continuación, se detalla la metodología que se propone para lograr la reducción de la distancia de Hausdorff que se encuentra entre una nube de puntos de alta densidad (original) y una nube de puntos de baja densidad (simplificada) de un mismo objeto 3D. Se comienza la explicación del análisis que derivó en los pasos concretos de la metodología que se siguieron para lograr el objetivo de la reducción de la distancia de Hausdorff, para posteriormente hablar del preprocesado que se hace en las nubes de puntos, luego se habla del procedimiento que conforma la optimización y, por último, se muestran los resultados que se obtuvieron al trabajar con un listado de objetos y un equipo computacional específico.

3.1: Análisis de la distancia de Hausdorff

Recordando que para mejorar la representación que se hace de un objeto 3D con una nube de puntos se debe disminuir el error que se obtiene con ella, se parte del análisis de cómo se debe abordar el problema si es que se quiere mejorar el error marcado por la distancia de Hausdorff.

Primeramente, y retomando el ejemplo mostrado en la sección "Preliminares" en la figura 3, se hace el análisis de qué ocurriría si se desplazan los puntos que conforman las distancias mayores de los puntos utilizados; anteriormente, a los puntos azules se les denominaba los puntos de la nube N_1 y los puntos verdes eran los puntos de la nube N_2, de ahora en adelante, se les denominará a los puntos azules como los puntos de la nube original, y a los puntos verdes como los puntos de la nube simplificada.

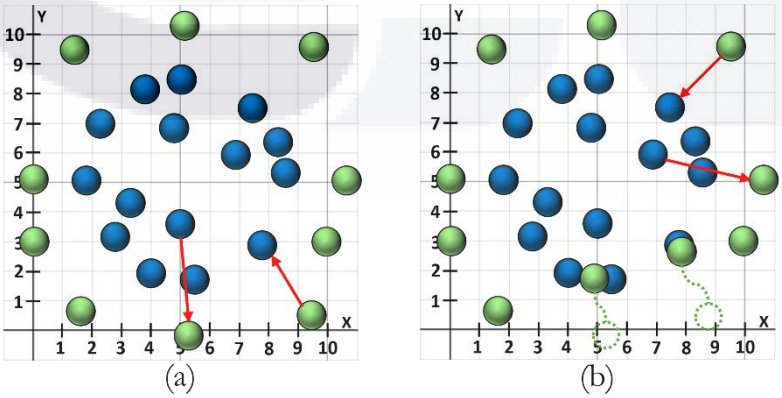


Figura 19 – Desplazando puntos para reducir la distancia de Hausdorff. (a) Distancias máximas antes del desplazamiento. (b) Distancias máximas después del desplazamiento

En la figura 19, se puede apreciar que, si se desplazan los puntos de la nube simplificada para reducir las distancias máximas anteriormente encontradas, ahora otros pares de puntos conformarán las distancias máximas que, a partir de este punto, serán las que determinen el valor del error de esta representación. Con esto, se puede apreciar que hay varias consideraciones para llevar a cabo la tarea de optimizar la nube simplificada: al mover un solo punto de la nube simplificada, la distancia de Hausdorff puede cambiar, pero será bastante costoso estar calculando esta métrica entre las dos nubes cada vez que se haga un solo movimiento; por lo que es necesario encontrar la manera de reordenar todos los puntos de la nube simplificada sin tener que estar calculando la distancia de Hausdorff global en cada movimiento.

Para poder solventar estas consideraciones, se aplica la estrategia: Divide y vencerás; ya que, si se logra crear una segmentación de vecindarios en los puntos de la nube original, los puntos de la nube simplificada que sean ligados a cada vecindario solo tendrán que considerar la distancia de Hausdorff local que haya en su respectivo vecindario.

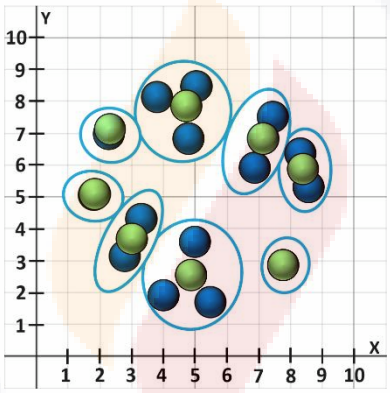


Figura 20 – Vecindarios formados para delimitar los puntos originales que considerará cada punto simplificado

En la figura 20 se puede ver de manera visual cómo es que esta estrategia permite solventar la cuestión de la optimización, tomando en cuenta dos ventajas que ahora se tienen: la primera, si se busca reasignar los puntos de la nube simplificada de manera que queden lo más cerca posible a los puntos originales que conforman el vecindario al que están ligados, la distancia de Hausdorff global disminuirá eventualmente; la segunda, es que para saber la distancia de Hausdorff local, solo se necesita saber cuál es la distancia máxima entre los puntos originales del vecindario con los puntos simplificados que lleguen a ser reasignados ahí (lo cual implica un menor tiempo de cálculo).

3.2: Preprocesado

Al obtener las nubes de puntos mediante el procedimiento propuesto por Tapia-Dueñas [12], los rangos en los que quedan son bastante alejados del rango que tienen las nubes de puntos originales. Esto se debe a la manera en la que opera el procedimiento mencionado. En él, primeramente, se toma un objeto 3D para voxelizarlo y, después de ello, por cada nivel de voxeles en el objeto, se obtienen representaciones mediante símbolos (códigos de cadena) que describen el contorno de cada nivel y, a partir de esa representación con código de cadena, obtienen los puntos principales que representan el contorno en cada nivel de voxeles. De esta manera, se obtiene una nube de puntos simplificada cuyo rango se modifica considerablemente. En la figura 21 se puede observar un ejemplo visual de la progresión que sigue un objeto 3D con el proceso descrito

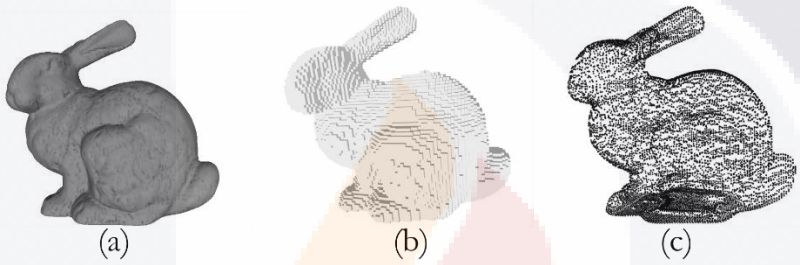


Figura 21 – Proceso de simplificado de Tapia-Dueñas. (a) Objeto original. (b) Objeto voxelizado. (c) Nube de puntos simplificada

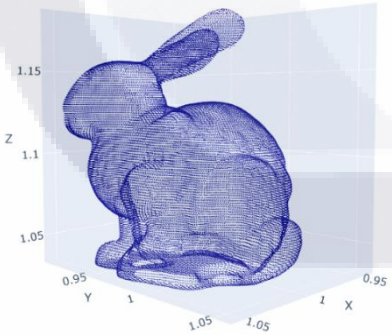


Figura 22 – Rango de la nube de puntos original

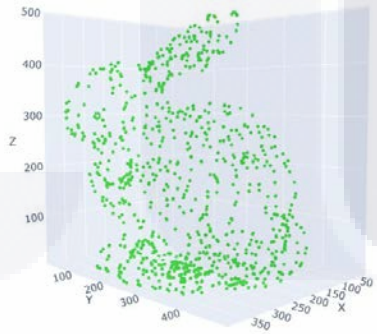


Figura 23 – Rango de la nube de puntos simplificada

Como puede apreciarse en las figuras 22 y 23, el rango en el que están ambas nubes de puntos dista con gran amplitud de sus rangos, por lo que se procede a aplicar un escalamiento sobre ellas para que ambas estén en un rango [0, 1]. Esto se hace para poder tener un marco de referencia más claro al momento de calcular las métricas de error que se obtienen con la nube

simplificada antes del proceso de optimización y con la nube optimizada después del proceso. Además, para asegurar una correcta alineación con ambas nubes de puntos, se procede a realizar una sobreposición de ambas nubes.

Escalamiento de nubes de puntos

Para el escalamiento de la nube de puntos se utiliza la técnica Min-Max ya que, como se mencionó en el capítulo 2, es una transformación que no afecta la naturaleza de los datos, siempre y cuando se respeten las proporciones al momento de escalar los valores en cada uno de los ejes.

Para lograr lo anterior, primero se parte de escalar a la nube de puntos original siguiendo los siguientes pasos:

i. Identificar el eje con mayor longitud, ya que este será escalado al rango [0, 1]
ii. Calcular el rango de escalamiento de los otros ejes partiendo de su rango antes del escalamiento, o sea, se toman los valores mínimo y máximo de su rango para ser divididos sobre la longitud del eje con rango mayor antes del escalamiento.

Por ejemplo, utilizando la nube de puntos original del objeto "Brain", se obtienen los valores que tendrá cada uno de sus ejes después del escalamiento como se muestra en la tabla 1. En ella se puede apreciar que el rango con mayor longitud es del eje z con una longitud de 6.4221, por lo tanto, su rango después del escalamiento será de [0:1]; por su parte, los rangos del eje x se establecen en el rango [0/6.4221, 5.6174/6.4221] y los del eje y en [0/6.2421, 5.6174/6.4221]. De esta manera, se mantienen las proporciones de los objetos al momento de hacer esta transformación de escalamiento sobre ellos.

Tabla 1 – Rangos del objeto "Brain" antes y después del escalamiento

Eje	Rango (antes del escalamiento)	Longitud (antes del escalamiento)	Rango (después del escalamiento)	Longitud (después del escalamiento)
X	[0, 5.2785]	5.2785	[0.0, 0.8219]	0.8220
Y	[0, 5.6174]	5.6174	[0.0, 0.8747]	0.8747
Z	[0, 6.4221]	6.4221	[0, 1]	1

Para la nube simplificada, se toman los mismos rangos de escalamiento asignados en la nube original.

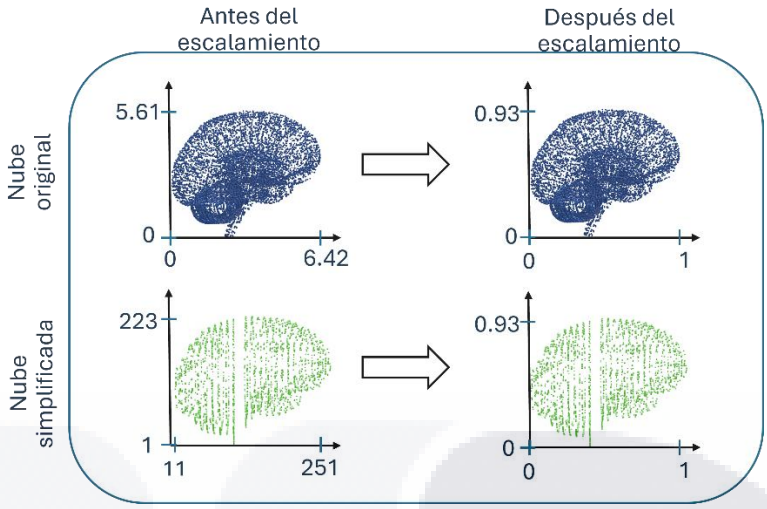


Figura 24 – Ejemplo visual del antes y el después del escalamiento en las nubes de puntos

En la figura 24 se puede apreciar un ejemplo visual de cómo se ven los rangos en ambas nubes de puntos antes y después; además de ver que, después del escalamiento, ambas tienen los mismos rangos.

Sobreposición de nubes de puntos

Para asegurar una correcta alineación de las nubes de puntos, se hace un centrado en cada una de manera que se recorren los puntos en cada eje para que queden centrados en el rango [0, 1]. Retomando los rangos que se obtuvieron para el objeto "Brain", cada uno de los rangos de sus ejes después de ser centrados se muestran en la tabla 2. Como se ve en ella, solo se terminan modificando los valores de los 2 ejes que no se encuentran en el rango de [0, 1].

Tabla 2 – Rangos del objeto "Brain" antes y después de la sobreposición (centrado)

Eje	Rango anterior	Rango centrado
X	[0.0, 0.8219]	[0.089, 0.911]
Y	[0.0, 0.8747]	[0.0626, 0.9374]
Z	[0.0, 1.0]	[0.0, 1.0]

En la figura 25 se puede encontrar un ejemplo visual del proceso descrito anteriormente, en el cual se puede ver que ambas nubes de puntos terminarán tanto con el mismo rango como con una alineación detallada. Si se imprimieran ambas nubes de puntos, se obtendría el resultado visual que se aprecia en la figura 26.

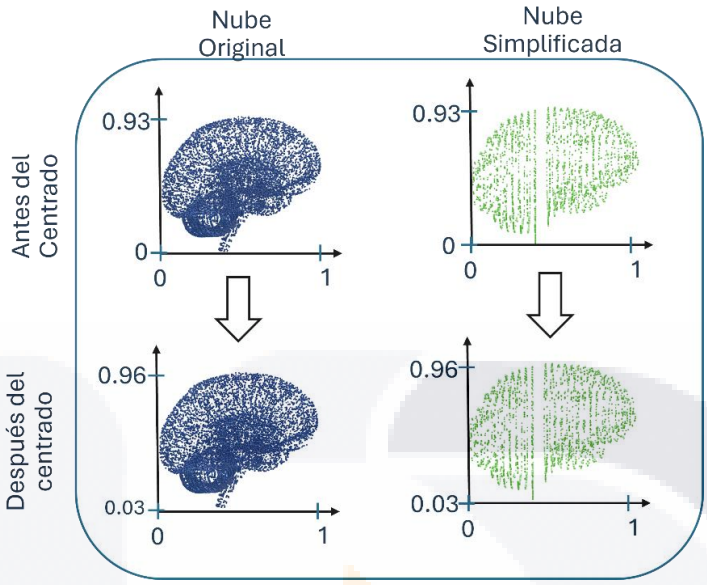


Figura 25 – Ejemplo visual de la sobreposición de nube de puntos (centrado)

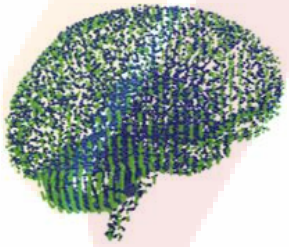


Figura 26 – Ejemplo visual de imprimir ambas nubes de puntos después del preprocesado

3.3: Optimización

Una vez que ambas nubes de puntos fueron escaladas y superpuestas de manera que ambas están sobre el mismo rango de espacio, ahora se procede a realizar la optimización de la distancia de Hausdorff que se obtiene entre ellas. Como se ha mencionado anteriormente, la estrategia "Divide y vencerás" se puede aplicar en este problema al delimitar cuáles puntos de la nube original tendrá que considerar cada punto de la nube simplificada. Por lo tanto, el algoritmo seguido para la optimización se compondrá de dos módulos principales: Ligado de puntos (decidir a dónde va cada punto de la nube simplificada) y Reasignación de puntos (decidir la posición que ocupará cada punto de la nube simplificada).

Visto de una manera general, el algoritmo que se sigue con esta estructura se muestra en la figura 27.

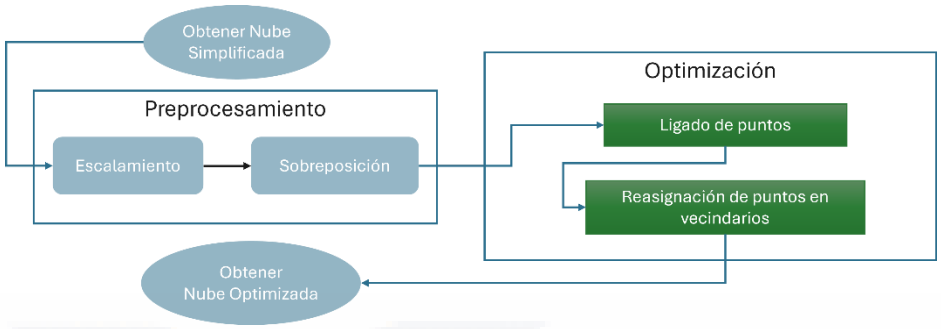


Figura 27 – Diagrama general de la metodología

Ligado de puntos

El objetivo de este módulo es delimitar cuáles puntos de la nube original son considerados por cada punto de la nube simplificada. Para esto, se necesita tanto de una segmentación inicial (vecindarios) en la nube de puntos original como de los enfoques para determinar cuántos puntos deben ser asignados en cada vecindario y, de esta manera, poder comparar diferentes maneras en las que esta tarea puede llevarse a cabo y evaluar cómo se comporta la distancia de Hausdorff en cada una de ellas.

- **Primer enfoque**

Como se ha mencionado, el primer paso es delimitar los vecindarios de puntos que se encontrarán en la nube original. Para ello, se toman como referencia los puntos de las nubes simplificadas para delimitar los vecindarios con ellos. Como la nube simplificada siempre tiene un número menor de puntos, esto puede ser aprovechado para crear la segmentación en vecindarios. Por ello, se propone el siguiente método de formación de vecindarios por medio de una esfera de detección.

Sea

$$N_o=\{o_1,o_2,\dots,o_n\},\ o_i \in \mathbb{R}^3 \quad (9)$$

el conjunto de puntos que representa a la nube original. Y sea

$$N_s=\{s_1,s_2,\dots,s_m\},\ s_j \in \mathbb{R}^3, m < n \quad (10)$$

el conjunto de puntos que representa a la nube simplificada.

Para determinar los vecindarios de los que se va a conformar cada nube original, se aplica en cada punto o_i una esfera de detección partiendo de un radio $r = 0$ y usando solo distancias euclidianas al seguir los siguientes pasos:

i. Obtener un rango $[axis - r, axis + r]$ con cada uno de los ejes del punto o_i.
ii. Si no se detectan puntos s_j en ese rango, repetir el paso anterior con $r = r + \alpha$, donde $\alpha = 0.1$.
iii. Si se detectaron puntos s_j, solo considerar aquellos que cumplan con la condición: $\|o_i, s_j\| \geq r$.
iv. Si ningún punto s_j detectado cumple la condición anterior, repetir el proceso desde el paso i con $r = r + \alpha$, donde $\alpha = 0.1$.
v. De los puntos s_j que cumplieron la condición anterior, seleccionar el punto con menor distancia y añadir el punto o_i al vecindario que está ligado a ese punto.

Estos pasos se pueden ver ejemplificados visualmente en el ejemplo 2D de la figura 28, en donde se aplican sobre un punto o_i hasta que este punto forma parte del vecindario de un punto s_j.

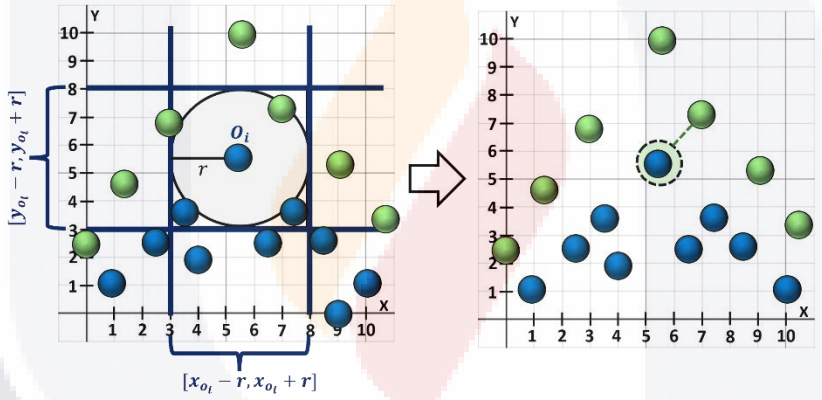


Figura 28 – Proceso de detección de puntos sobre un punto o_i

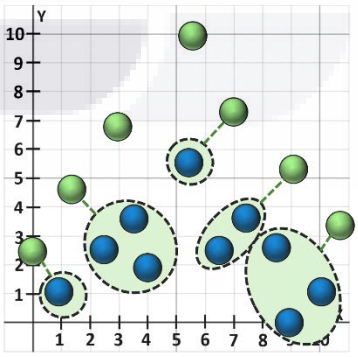


Figura 29 – Resultado de aplicar el proceso de detección sobre todos los puntos de N_O

Al repetir este procedimiento en todos los puntos de la nube N_o, se consigue la segmentación que se necesitaba al tener a todos los puntos de N_o en vecindarios con n' cantidad de puntos en

cada uno. En la figura 29 se puede apreciar el ejemplo visual de esto, mostrando además que, aunque todos los puntos de N_o están ligados, no todos los puntos de N_s lo están.

Entonces surge la pregunta: ¿qué pasaría si solo se reasignan los puntos s_j que están ligados y se descartan aquellos que no lo están? De manera que ahora se tiene un primer enfoque en el cual se tiene un punto s_j para cada vecindario de N_o y se puede hacer una prueba para ver cuánto puede reducirse la distancia de Hausdorff global utilizando un solo punto s_j por vecindario ya que, eventualmente, al hacerlo así el número de puntos en la nube optimizada será menor al número de puntos que se encuentran en N_s.

En la figura 30, se muestra el diagrama de la metodología siguiendo este primer enfoque de optimización.

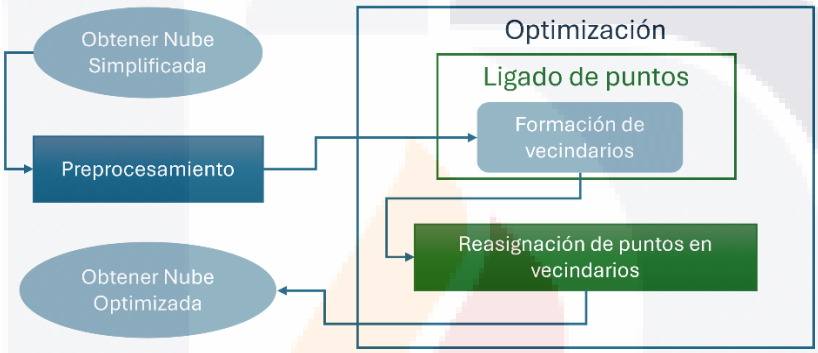


Figura 30 – Diagrama de la metodología: primer enfoque de optimización

- **Segundo enfoque**

Como se explicó en el primer enfoque, se obtuvieron vecindarios con n' cantidad de puntos originales en cada uno de ellos; sin embargo, algunos puntos simplificados resultaron sin estar ligados a algún vecindario de la nube N_o. Entonces, surge la interrogante: ¿qué pasaría si se utilizan todos los puntos de N_s ligándolos a su vecindario más cercano?

Para ello, una vez que se tienen todos los puntos de la nube N_o segmentados en vecindarios y ligados a un punto de la nube N_s, se aplica el proceso de la esfera de detección sobre los puntos s_j no ligados a un vecindario (s_u); aunque en este proceso debe de considerarse que, como se quiere aprovechar todo el número de puntos en N_s, el número de puntos originales en el vecindario (n') siempre debe ser mayor o igual al número de puntos s_j ligados (m').

Por lo tanto, se aplican los siguientes pasos sobre los puntos s_u:

i. Obtener un rango $[axis - r, axis + r]$ con cada uno de los ejes del punto s_u.
ii. Si no se detectan puntos o_i en ese rango, repetir el paso anterior con $r = r + \alpha$, donde $\alpha = 0.1$.

iii. Si se detectaron puntos o_i, solo considerar aquellos que cumplan con la condición: $\|s_j, o_i\| \geq r$.

iv. Si ningún punto o_i detectado cumple la condición anterior, repetir el proceso desde el paso i con $r = r + \alpha$, donde $\alpha = 0.1$.

v. De los puntos o_i que cumplieron la condición anterior, seleccionar el vecindario más cercano que no rompa la condición $m' \leq n'$ al añadir al ligar el punto s_u en él. Si ninguno de los vecindarios detectados puede mantener la condición al ligar el punto s_u en ellos, se repite el proceso desde el paso i con $r = r + \alpha$, donde $\alpha = 0.1$.

En la figura 31 se muestra un ejemplo visual de estos pasos aplicado a un punto s_u y también se muestra el resultado final de aplicar este procedimiento sobre todos los puntos s_u para que ahora todos los puntos de N_s estén ligados a un vecindario y se puedan aprovechar en el paso de la reasignación.

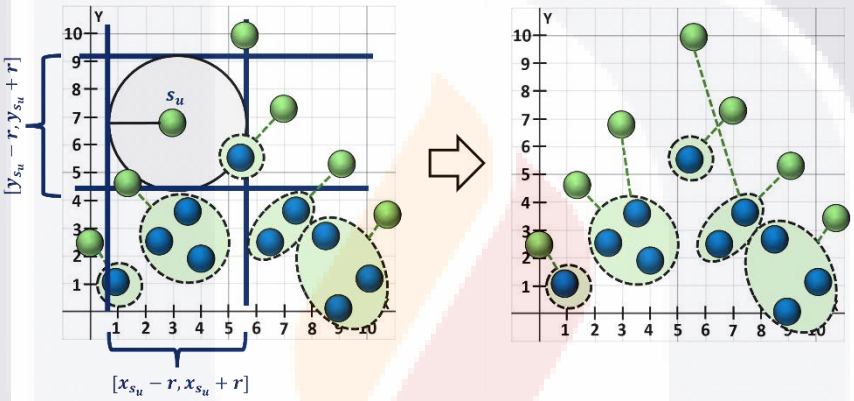


Figura 31 – Procedimiento y resultado de ligar los puntos simplificados sueltos (s_u)

En la figura 32, se muestra el diagrama de la metodología siguiendo este segundo enfoque de optimización.

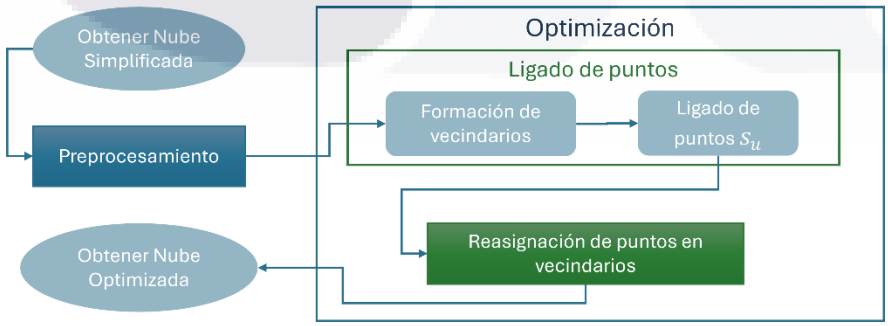


Figura 32 – Diagrama de la metodología: segundo enfoque de optimización

- **Tercer enfoque**

Como se puede apreciar en el resultado de ligar todos los puntos de la nube N_s en la figura 31, se puede apreciar una situación en la que dos puntos s_j quedaron ligados a un vecindario con dos puntos o_i, sin embargo, utilizar así los puntos s_j no permitirá la mayor reducción de la distancia de Hausdorff local en los vecindarios ya que, inmediatamente después, se encuentra un vecindario con tres puntos o_i. Por esta razón, surge una cuestión que delimita este tercer enfoque de optimización: ¿qué pasaría si se utilizan los puntos de la nube N_s en los vecindarios donde la distancia de Hausdorff es mayor en vez de utilizarlos en sus vecindarios más cercanos?

Para abordar este nuevo enfoque, se parte primeramente de la formación de vecindarios que se empleó en el primer enfoque para proceder a realizar los siguientes pasos:

i. Hacer la reasignación del único punto s_j que los vecindarios formados tienen ligado, de manera que al hacer dicha reasignación se calcula la distancia de Hausdorff local que hay en cada uno de los vecindarios.
ii. Si hay puntos simplificados no ligados s_u disponibles, asignar un punto s_u al vecindario con mayor distancia de Hausdorff local y actualizar dicha distancia.
iii. Repetir el paso anterior hasta que no haya más puntos s_u.

En la figura 33 se puede encontrar el ejemplo visual que involucra el primer paso de la reasignación de los puntos s_j ligados con la formación de vecindarios y en la figura 34 se puede apreciar la asignación de un punto s_u al vecindario con la distancia de Hausdorff mayor en ese momento.

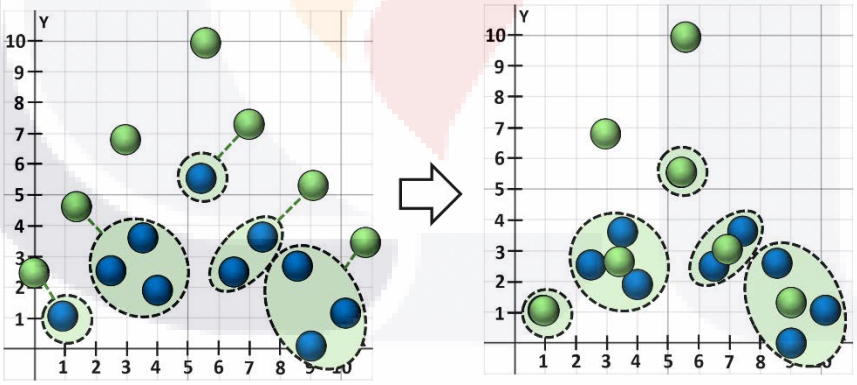


Figura 33 – Reasignación del primer punto s_j a los vecindarios creados

Con esto puede apreciarse que el resultado del ligado de puntos es diferente al ligado de puntos que se obtienen con el segundo enfoque de optimización.

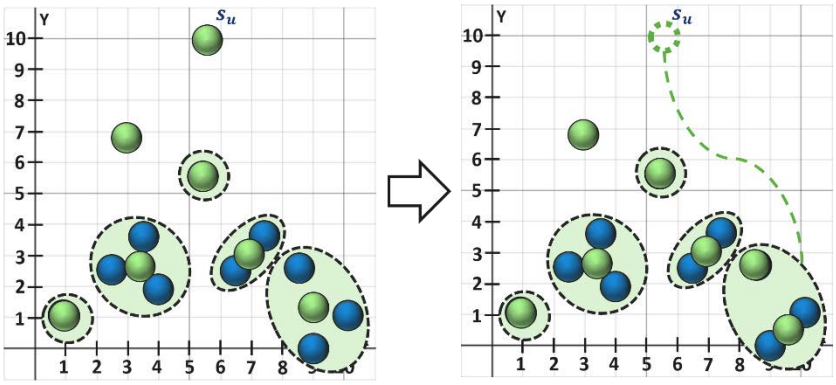


Figura 34 – Asignación de un punto s_u a la vecindad con mayor distancia de Hausdorff

En la figura 35, se muestra el diagrama de la metodología siguiendo este tercer enfoque de optimización.

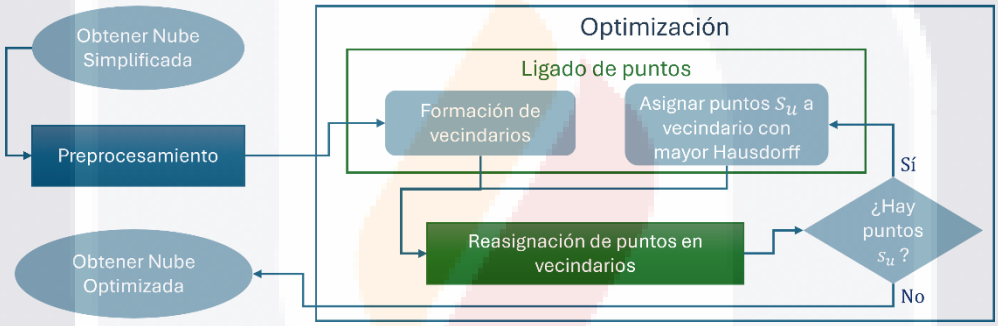


Figura 35 – Diagrama de la metodología: tercer enfoque de optimización

Reasignación de puntos

El objetivo de este módulo es determinar estratégicamente cómo serán reasignados los puntos s_j de la nube simplificada N_s que han sido ligados a cada vecindario formado de puntos o_i de la nube original N_o. Dicho de otra manera, el objetivo es determinar una ubicación estratégica para cada punto simplificado de manera que se asegure una reducción en la distancia de Hausdorff local de cada vecindario, este es un aspecto bastante importante por el hecho de que analizar todas las combinaciones posibles que habría al acomodar los m' puntos simplificados ligados a los n' puntos originales sería una tarea con alta complejidad computacional y tiempo de ejecución.

Por lo tanto, se parte de analizar los puntos que conforman los vecindarios que se obtuvieron en la nube N_o y la cantidad de puntos de la nube N_s que fueron ligados en el módulo del ligado de puntos para entender cómo es que los puntos s_j pueden ser reasignados a una ubicación estratégica.

De esta manera, es que se identifican dos escenarios generales de reasignación que dependen del valor que tenga m' y, posteriormente, se identifican escenarios específicos en cada escenario general dependiendo del valor que tenga n' para intentar aliviar la carga computacional al hacer la menor cantidad de operaciones computacionales posibles.

- **Primer escenario general**

El primer escenario general ocurre cuando $m' = 1$, o sea, cuando solo se tiene un punto simplificado asignado a un vecindario de la nube N_o. En este escenario general se pueden encontrar los siguientes escenarios específicos:

- **1 a 1**: cuando $n' = 1$
- **1 a 2**: cuando $n' = 2$
- **1 a varios**: cuando $n' \geq 3$

En el **escenario 1 a 1**, la posición estratégica que puede ocupar el punto s_j para asegurar una reducción local de la distancia de Hausdorff es la ubicación del único punto o_i que conforma el vecindario (figura 36).

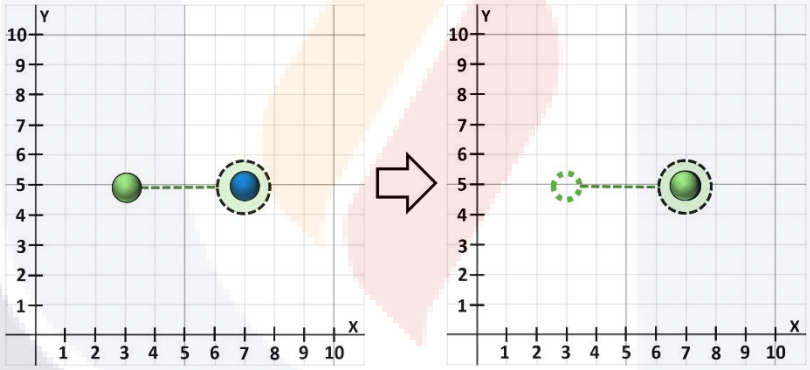


Figura 36 – Escenario 1 a 1

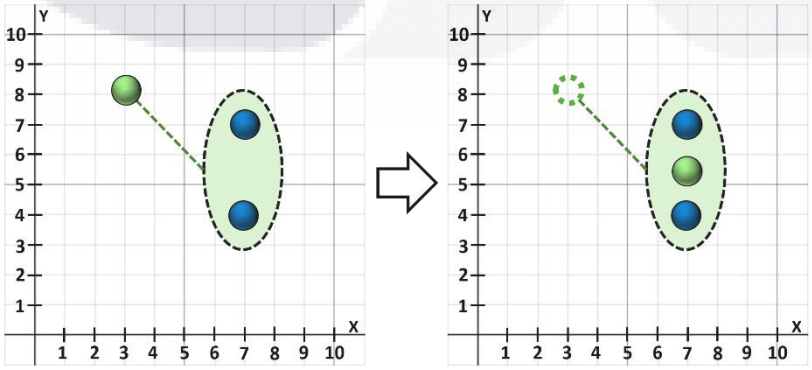


Figura 37 – Escenario 1 a 2

En el **escenario 1 a 2**, la posición estratégica que puede ocupar el punto s_j para asegurar una reducción local de la distancia de Hausdorff es la ubicación que se obtiene en medio de los dos puntos o_i que conforman el vecindario (figura 37).

En el escenario **1 a varios**, la posición estratégica que puede ocupar el punto s_j se busca a partir de dos pasos principales: la validación del punto medio y la búsqueda alrededor del centro de masa.

La validación del punto medio sigue los siguientes pasos:

i. Identificar a los puntos o_i más distantes del vecindario
ii. Posicionar el punto s_j en medio de los puntos o_i más distantes
iii. Calcular la distancia de esa ubicación del punto s_j con todos los integrantes o_i del vecindario. Si la distancia mayor de esa ubicación es con respecto a los puntos más distantes o_i, entonces esa ubicación representa la posición estratégica con la mejor reducción de la distancia de Hausdorff.

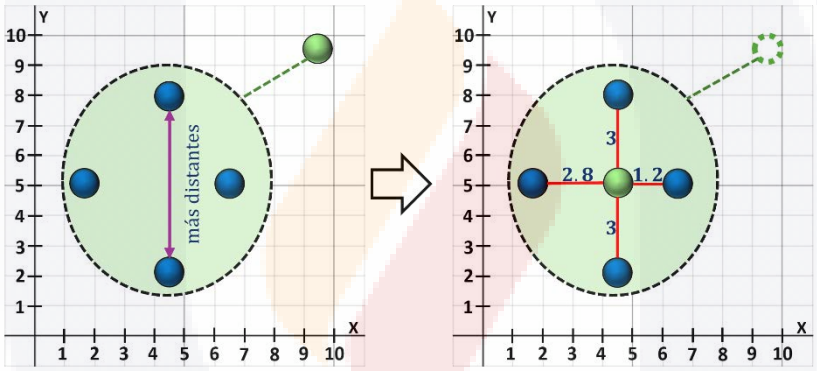


Figura 38 – Escenario 1 a varios: Validación del punto medio

En la figura 38 se puede apreciar un ejemplo visual 2D en el que la validación del punto medio fue positiva al determinar que la posición media de los dos puntos o_i más distantes resultó ser la ubicación con la mejor reducción de la distancia de Hausdorff, ya que cualquier otra ubicación no podrá ofrecer una distancia de Hausdorff menor a 3.

Sin embargo, existen casos en los que la validación del punto medio no es positiva. En la figura 39 se puede apreciar un ejemplo en el que, al ubicar el punto s_j en medio de los dos puntos o_i más distantes y al calcular las distancias de esa ubicación con todos los miembros del vecindario, la distancia mayor no se da con los puntos más distantes, se da con otro miembro del vecindario.

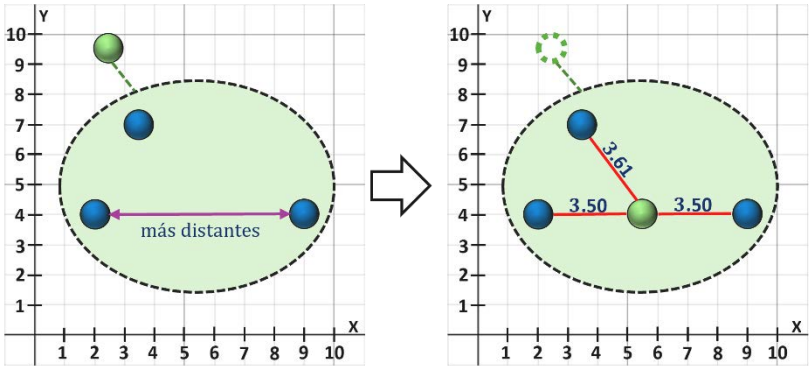


Figura 39 – Ejemplo donde no se cumple la validación del punto medio

En estos casos, cuando no se cumple la validación del punto medio, se hace la búsqueda alrededor del centro de masa. La cual sigue los siguientes pasos:

i. Ubicar el punto s_j en el centro de masa del vecindario.
ii. Recorrer el punto s_j a lo largo del rango que tiene el vecindario en el eje 'x', de manera que el punto recorre 20 posiciones intermedias, equidistantes, que cubren todo el rango. En cada una de esas posiciones, se calcula la distancia de Hausdorff que tendría el punto s_j con respecto a los miembros del vecindario.
iii. Se elige la posición donde la distancia de Hausdorff es menor (P_{Hm}) y, a lo largo de la posición anterior y posterior de P_{Hm}, se vuelve a recorrer el punto s_j en 20 posiciones intermedias en este nuevo rango y se vuelve a calcular la distancia de Hausdorff en cada de una de esas posiciones.
iv. Si la distancia de Hausdorff no llega a reducirse en alguna de esas nuevas 20 posiciones, se termina la búsqueda. En caso contrario, se toma la nueva P_{Hm} y se repite iterativamente el proceso de generar 20 posiciones nuevas intermedias en las posiciones anterior y posterior de la nueva P_{Hm}, para recorrer el punto s_j sobre ellas, hasta que deje de haber reducción en la distancia de Hausdorff.
v. Cuando deja de haber reducción en la distancia de Hausdorff, se toma la posición P_{Hm} como la posición final de x donde el punto s_j será reasignado.
vi. Se repiten los pasos ii al v con los ejes y y z para determinar la posición final de cada eje donde el punto s_j será reasignado.

En las figuras 40 – 42, se puede apreciar un ejemplo visual 2D de los pasos anteriormente mencionados. En la figura 40 se representa el posicionamiento del punto s_j en el centro de masa y el inicio de la búsqueda en las 20 posiciones intermedias del rango en el que está el vecindario sobre el eje x. En la figura 41, se muestra el ejemplo del primer y el último P_{Hm} que se establece con las iteraciones de las 20 posiciones intermedias hasta que la distancia de Hausdorff ya no decrece; cabe mencionar, que solo se toman en cuenta cuatro dígitos decimales de Hausdorff

para este procedimiento. En la figura 42 se muestra el inicio de la búsqueda en las 20 posiciones intermedias del rango en el que está el vecindario sobre el eje **y** a la altura de la posición final de x.

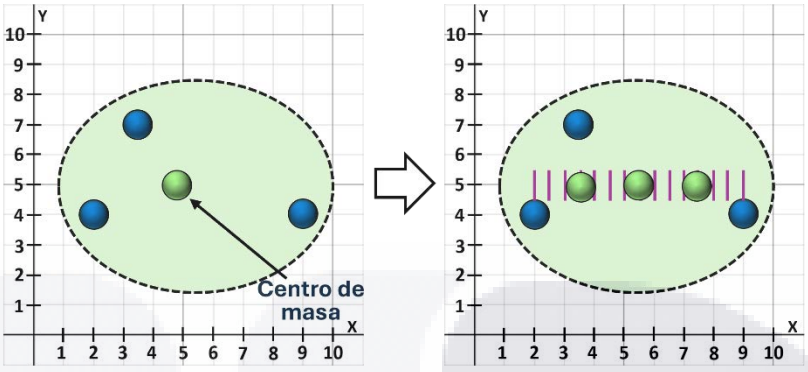


Figura 40 – Escenario 1 a varios: Búsqueda alrededor del centro de masa (búsqueda a lo largo del rango del eje *X*)

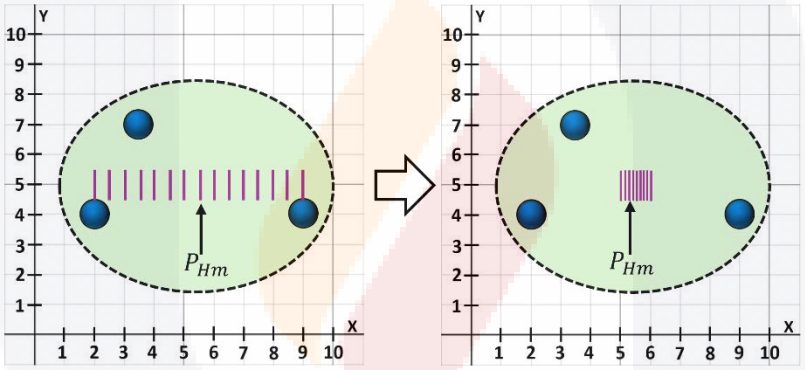


Figura 41 – Escenario 1 a varios: Búsqueda alrededor del centro de masa (Primer y último P_{HM} del eje *X* en las iteraciones de las posiciones intermedias)

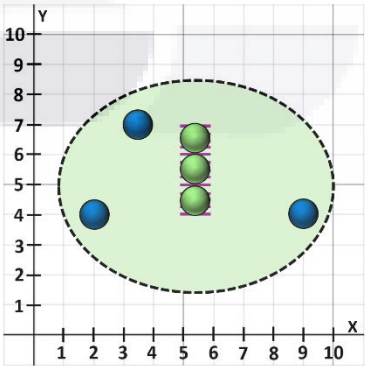


Figura 42 – Escenario 1 a varios: Búsqueda alrededor del centro de masa (Búsqueda a lo largo del rango del eje *y*)

- **Segundo escenario general**

El primer escenario general ocurre cuando $m' > 1$, o sea, cuando se tienen más de un punto simplificado asignado a un vecindario de la nube N_o. En este escenario general se pueden encontrar los siguientes escenarios específicos:

- Cuando $n' = m'$
- Cuando $n' = m' + 1$
- Cuando $n' \geq m' + 2$

El **escenario $n' = m'$** se da cuando se tiene que la cantidad de puntos ligados s_j es igual a la cantidad de puntos o_i en el vecindario. En este escenario, las posiciones que aseguran una reducción local son las mismas que tienen los puntos o_i; de manera que cada punto s_j se posiciona sobre una de esas ubicaciones (figura 43).

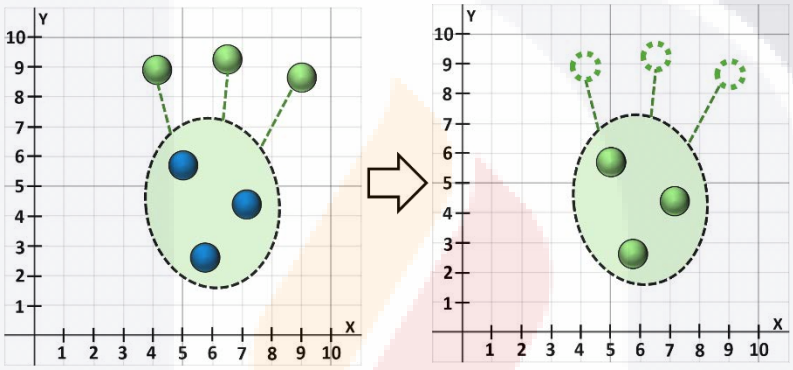


Figura 43 – Escenario $n' = m'$

El **escenario $n' = m' + 1$** se da cuando la cantidad de puntos ligados s_j es menor por un solo punto a la cantidad de puntos o_i en el vecindario. En este escenario, se siguen tres pasos principales para encontrar las posiciones que aseguren una reducción en la distancia de Hausdorff local:

i. Asignar cada punto s_j a la posición de su punto o_i más cercano
ii. Cuando ya no queden puntos s_j por asignar, se busca el vecino o_i más cercano de aquel que quedó sin un punto s_j en su posición.
iii. Se posiciona el punto s_j en medio de esos dos puntos o_i.

En la figura 44 se encuentra un ejemplo 2D de estos pasos. En esta figura, se puede apreciar que, primeramente, los puntos s_j ocupan la ubicación de un integrante del vecindario para posteriormente posicionar uno de estos puntos entre el punto o_i que quedó sin uno en su posición y su vecino o_i más cercano; de esta manera, se aprovecha la cantidad de puntos s_j disponibles.

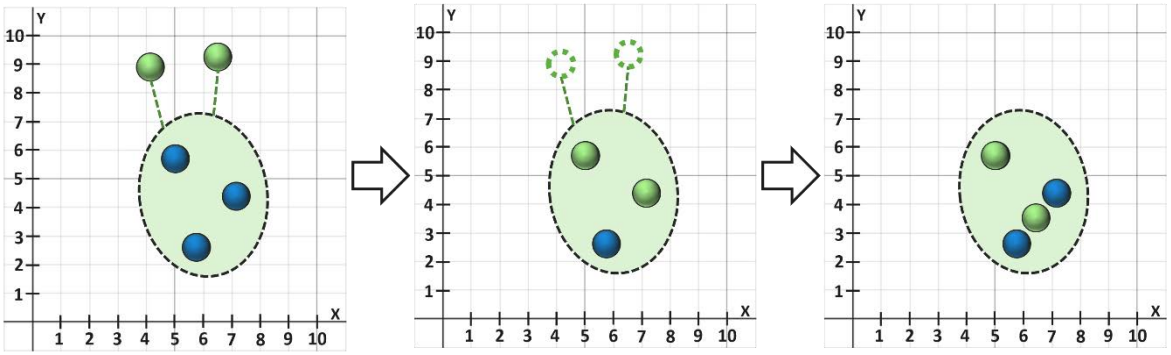


Figura 44 – Escenario $n' = m' + 1$

El **escenario $n' \geq m' + 2$** se da cuando la cantidad de puntos ligados s_j es menor por dos o más puntos a la cantidad de puntos o_i en el vecindario. En este escenario, se siguen dos pasos para encontrar las posiciones que aseguren una reducción en la distancia de Hausdorff local:

i. Segmentar los n' puntos del vecindario en m' clústeres con agrupamiento jerárquico, usando el método de enlace simple.
ii. Asignar cada punto s_j a su clúster más cercano.
iii. En cada clúster aplicar los procedimientos del primer escenario general.

En la figura 45 se muestra un ejemplo 2D de estos pasos. Como se puede apreciar, la estrategia de crear m' clústeres en el vecindario para asignarles solo un punto s_j a cada uno permite las condiciones adecuadas para que se puedan volver a aprovechar las estrategias desarrolladas en el primer escenario general y, así, buscar la mayor eficiencia computacional.

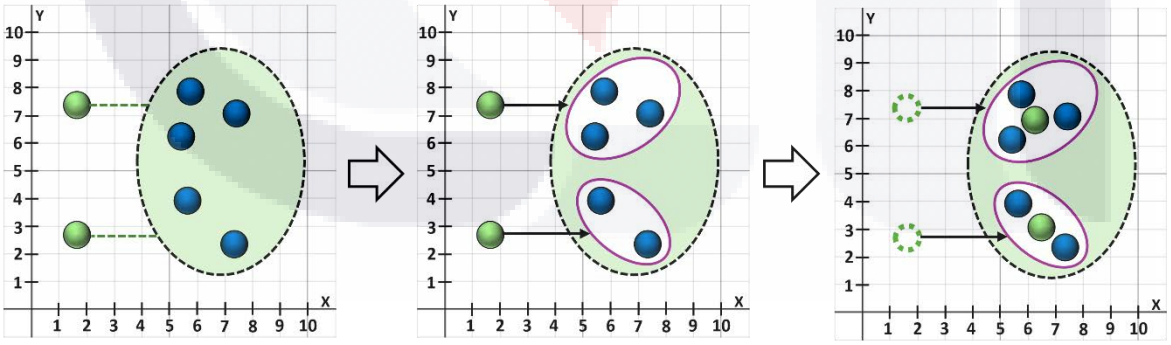


Figura 45 – Escenario $n' \geq m' + 1$

3.4: Resultados

Una vez establecida la metodología para preparar y optimizar las nubes de puntos simplificadas, se procede a realizar las pruebas correspondientes para ver su efectividad en cada uno de los enfoques que se proponen para la parte del ligado de puntos en el flujo de la optimización.

Equipo y herramientas empleadas

La metodología explicada a lo largo de este capítulo fue desarrollada en el lenguaje de Python 3.10 y ejecutada con las nubes de puntos originales y simplificadas de los siguientes objetos 3D:

- "Bunny"
- "Brain"
- "Cow"
- "Dragon"
- "Happy Buddha"
- "Hand"
- "Hepteroid"
- "Hippo"
- "Horse"
- "Lucy"

Tanto el código de la metodología como el código para obtener las métricas de error, así como los archivos de las nubes de puntos que se utilizan en formato ply, se encuentran en el siguiente repositorio: https://github.com/MRiv-98/Hausdorff-Optimization-in-3D-Point-Clouds. Dentro del repositorio, se detallan las características técnicas (librerías) que necesita el código para ser ejecutado. Cabe mencionar que los resultados expuestos a continuación fueron obtenidos utilizando lo siguiente:

- Equipo de cómputo con procesador Intel Core i5 de 11va generación con una velocidad de 2.60 GHz, 32 GB de RAM y ½ terabyte de almacenamiento en estado sólido.
- IDE Spyder 6 del entorno de Anaconda Navigator.

Estas características no tienen injerencia sobre los resultados en cuanto a la optimización, sin embargo, pueden hacer que los tiempos de ejecución sean diferentes.

Cabe mencionar que el código que se obtiene al desarrollar la metodología de este trabajo tiene una complejidad $O(n)$. Esto se establece por la complejidad que presenta cada proceso principal que sigue el código y, basándonos en el tercer enfoque que es el que mayor desarrollo en código presenta, se tiene que cada proceso obtiene la siguiente complejidad:

- Preprocesamiento de los datos: $O(n)$
- Formación de vecindarios: $O(n \cdot s')$
- Primera reasignación de puntos simplificados en vecindarios: $O(log[n] * {s'}^3)$
- Reasignación iterativa: $O\big(s' * \langle log(s') + \big(s' + \{(n' * s') + n'\} + \{n' + n' + (s' * n') + (s' * {s'}^3)\}\big)\rangle\big)$

Ahora, tanto s' como n' son cantidades bastante menores a n, por lo que el proceso de mayor complejidad es de $O(n)$.

Análisis de resultados en las nubes optimizadas (3 enfoques)

Como punto de partida, se obtiene el error de la distancia de Hausdorff presente en las nubes de puntos obtenidas por el método de Tapia-Dueñas [12]. En la tabla 3 se muestra el listado de objetos utilizado junto con la cantidad de puntos presente en la nube original (Puntos de N_o), la cantidad de puntos presentes en la nube simplificada (Puntos de N_s) y la distancia de Hausdorff entre ambas ($Hau(N_o, N_s)$). Como se observa en ella, el promedio de la distancia de Hausdorff entre la nube original y simplificada es de 0.10, lo cual habla de que el resultado de seguir el método de Tapia-Dueñas [12] ofrece un buen resultado considerando que se presenta una considerable reducción entre los puntos de N_o y los puntos de N_s.

Tabla 3 – Comparación de las nubes originales y simplificadas

Objeto	Puntos de N_o	Puntos de N_s	$Hau(N_o, N_s)$
Bunny	35,947	1,202	0.12
Brain	18,844	3,085	0.19
Cow	2,903	346	0.11
Dragon	22,998	2,694	0.05
Happy Buddha	7,108	2,229	0.05
Hand	327,323	2,059	0.06
Hepteroid	2,886,678	2,068	0.12
Hippo	23,105	330	0.12
Horse	48,485	256	0.14
Lucy	262,909	392	0.08

Ahora, se procede a observar los resultados obtenidos en las nubes optimizadas que se obtienen con cada uno de los enfoques descritos en la metodología de optimización. A partir de este punto, a cada enfoque se le nombrará de la siguiente manera: 1 por vecindario (primer enfoque), vecindarios más cercanos (segundo enfoque), asignación iterativa (tercer enfoque).

En la tabla 4, se pueden encontrar los resultados de cada uno de los enfoques propuestos en este trabajo, así como los resultados que se obtienen con ellos; tanto el número de puntos que resulta en la nube optimizada (puntos de N_{op}), como la distancia de Hausdorff que hay entre la nube original y la nube optimizada ($Hau(N_o, N_{op})$) y el tiempo de ejecución en segundos que se demoró cada una de las pruebas.

Como se puede observar en ella, todos los enfoques obtuvieron una reducción en la distancia de Hausdorff, aunque no de la misma manera. El hecho de que el primer y segundo enfoque obtuvieron la misma reducción confirma el análisis que se hizo de la posibilidad de no estar aprovechando todos los puntos de la nube N_s al asignar los puntos a su vecindario más cercano.

Por tal manera, y confirmando los análisis realizados en la elaboración de cada uno de los enfoques, el tercer enfoque fue el que mayor reducción tuvo en los experimentos al reducir la distancia de Hausdorff a un promedio de 0.06. Aunque el hecho de que el primer enfoque obtuvo tanto una reducción del error, dejándolo en un promedio de 0.07, como una reducción en la cantidad de puntos, que en promedio se redujo un 18%, también confirma que las estrategias planteadas en los escenarios generales de reasignación fueron efectivas al buscar posiciones estratégicas que ofrecieran una reducción en la distancia de Hausdorff. Sin embargo, el único resultado que puede ser mejorado es el de tiempo de ejecución, cuyo promedio es de 2,074 segundos. Aunque, sobre este aspecto, en el desarrollo del código, se observó que conforme se fue estandarizando para que cada función estuviera en un método independiente, los tiempos de ejecución crecían cada vez más. Esto puede ser un factor que está directamente relacionado con que Python es un lenguaje interpretado y no compilado.

Tabla 4 – Resultados de optimización

Objeto	Enfoque	Puntos de N_{op}	$Hau(N_o, N_{op})$	Tiempo (s)
Bunny	1 por vecindario	**1,097**	0.080	140.16
	Vecindarios más cercanos	1,202	0.080	140.34
	Asignación iterativa	1,202	**0.065**	148.70
Brain	1 por vecindario	**2,587**	0.105	68.62
	Vecindarios más cercanos	3,085	0.102	70.02
	Asignación iterativa	3,085	**0.066**	81.75
Cow	1 por vecindario	**302**	0.073	7.10
	Vecindarios más cercanos	346	0.073	7.12
	Asignación iterativa	346	**0.060**	8.25
Dragon	1 por vecindario	**1,245**	0.0391	110.39
	Vecindarios más cercanos	2,694	0.0391	70.38
	Asignación iterativa	2,694	**0.0347**	63.75
Happy Buddha	1 por vecindario	**2,062**	0.032	17.7
	Vecindarios más cercanos	2,229	0.032	18.31
	Asignación iterativa	2,229	**0.028**	17.76
Hand	1 por vecindario	**2,042**	0.030	1,505.56
	Vecindarios más cercanos	2,059	0.030	1,536.31
	Asignación iterativa	2,059	**0.028**	1,498.60
Hepteroid	1 por vecindario	**1,155**	0.0631	8,694.43
	Vecindarios más cercanos	2,068	0.0631	7,975.35
	Asignación iterativa	2,068	**0.0531**	11,100.28
Hippo	1 por vecindario	**294**	0.080	493.91
	Vecindarios más cercanos	330	0.080	476.16
	Asignación iterativa	330	**0.068**	567.27
Horse	1 por vecindario	245	0.107	968.96
	Vecindarios más cercanos	256	0.107	969.15
	Asignación iterativa	256	**0.106**	1,073.11

Lucy	1 por vecindario	832	0.059	8,115.44
	Vecindarios más cercanos	839	0.059	8,112.48
	Asignación iterativa	839	0.055	8,164.20

Como ya se comparó el resultado de la propuesta del trabajo actual con el resultado del trabajo que se tomó como punto de partida [12], ahora se procede a realizar la comparación con otros trabajos de la literatura. En este caso, se compara tanto las nubes de puntos que se obtienen con la metodología presente y el método de Tapia-Dueñas [12], como las nubes que se obtienen con el método de El Sayed [10] y E. Leal [11]. Sin embargo, cabe mencionar que estos dos últimos autores no comparten la métrica de error que se empleó anteriormente, ellos comparten los resultados de sus trabajos usando las métricas de la distancia máxima del error (Δ_{max}) y la distancia promedio del error (Δ_{avg}).

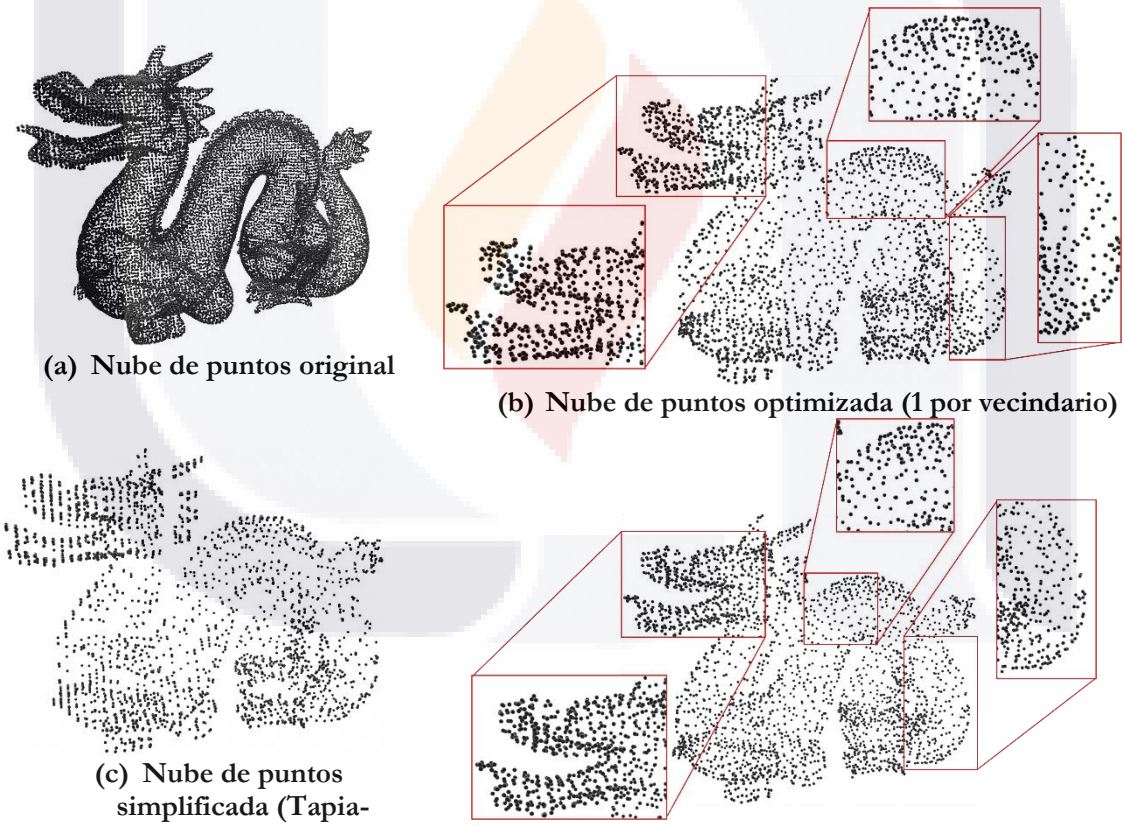
Por ello, se calcula cada una de esas métricas en las nubes de puntos: "Bunny", "Dragon" y "Horse", que se obtienen tanto con los 3 enfoques de este trabajo como con el método de Tapia-Dueñas [12]. Ya que esos tres objetos son los que se usan en común con los trabajos de El Sayed [10] y E. Leal [11]. Los resultados se muestran en la tabla 5. En ella, se puede apreciar que el primer enfoque de este trabajo sigue siendo el que obtiene una menor cantidad de puntos en las nubes resultantes, aunque en las métricas de error, se puede ver que el resultado varía considerablemente. El Sayed [10] logra un mejor resultado con "Bunny" y "Horse", aunque con "Dragon" solo logra un mejor resultado en la distancia promedio del error, ya que la distancia máxima del error tiene un mejor resultado con el tercer enfoque de este trabajo. Aunque, si se analiza con detenimiento tanto la cantidad de puntos como las métricas de error, se puede apreciar que es coherente que tanto El Sayed [10] como E. Leal [11] generalmente logren un mejor resultado ya que ellos obtienen nubes de puntos con mayor densidad; por lo cual se puede ver reflejada la relación que se viene presentando desde la introducción de este trabajo: a mayor cantidad de puntos hay mayor fidelidad del objeto, que es igual a, menor error en la nube de puntos.

Tabla 5 – Comparación de resultados entre diversos métodos

Objeto	Puntos de N_o	Método	Puntos en las nubes	Δ_{max}	Δ_{avg}
Bunny	35,947	El Sayed [10]	16,476	**0.0052**	**0.00027**
		E. Leal [11]	1,797	0.0060	0.00051
		Tapia-Dueñas [12]	1,202	0.1168	0.0324
		1 por vecindario	**1,097**	0.1046	0.0236
		Vecindarios más cercanos	1,202	0.01012	0.0220
		Asignación iterativa	1,202	0.0698	0.0178
Dragon	437,645	El Sayed [10]	33,960	0.4432	**0.00001**
	22,998	Tapia-Dueñas [12]	2,694	0.0548	0.0147
		1 por vecindario	**1,235**	0.0391	0.0121

		Vecindarios más cercanos	2,694	0.0391	0.0120
		Asignación iterativa	2,694	**0.0347**	0.0118
Horse	48,485	El Sayed [10]	2,428	**0.0035**	**0.00032**
		Tapia-Dueñas [12]	**256**	0.1369	0.0559
		1 por vecindario	245	0.1065	0.0381
		Vecindarios más cercanos	256	0.1065	0.0379
		Asignación iterativa	256	0.1064	0.0372

Para terminar de analizar los resultados, se procede a visualizar las nubes de puntos optimizadas que se obtuvieron con los enfoques para observar las mejoras físicas que pudieron haber presentado cada una, ya que si el error ha disminuido, la representación debería ser mejor. Para ello, en la figura 46 se muestra la comparación de cómo es que se ve el objeto "Dragon" en la nube de puntos original (a), la nube simplificada de Tapia-Dueñas [12] (c) y los resultados de la metodología presentada en el primer enfoque (b) y el tercer enfoque (d).



(a) Nube de puntos original

(b) Nube de puntos optimizada (1 por vecindario)

(c) Nube de puntos simplificada (Tapia-Dueñas [12])

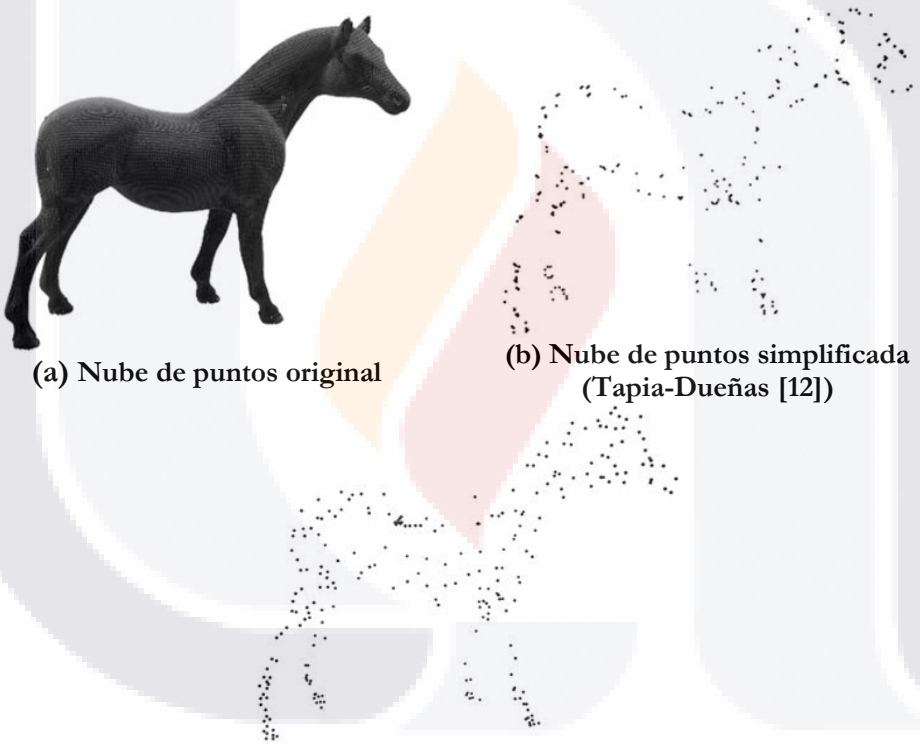
(d) Nube de puntos optimizada (asignación iterativa)

Figura 46 – Comparación entre las nubes de puntos del mismo objeto 3D (Dragon)

Como puede apreciarse en los resultados mostrados en la figura 46, algunas características visuales del objeto original que fueron alteradas durante el proceso de simplificación de Tapia-Dueñas [12] se recuperaron tanto en la nube del primer enfoque como en la nube del tercer

enfoque; en este caso específico del "Dragon", las piernas recuperan parte de su forma original. Además, en la nube simplificada se puede apreciar que hay efecto de capas que da como resultado zonas en las que se ve una clara ausencia de puntos; en las nubes optimizadas, tanto reduciendo el número de puntos presentes en la nube simplificada, como preservándolos, se puede apreciar que se fue esa apariencia de una figura segmentada en capas, y que la distribución de los puntos es más homogénea. Lo que confirma el hecho de que, al reducir el error en una nube de puntos, se mejora su fidelidad con respecto al objeto que representa.

En las figuras 47 – 49 se pueden apreciar más comparaciones visuales entre las nubes de puntos original, las simplificadas de Tapia-Dueñas [12] y las optimizadas del tercer enfoque. En esos casos también se puede apreciar que las nubes optimizadas recuperaron características del objeto original y que se consigue una distribución más homogénea de los puntos.

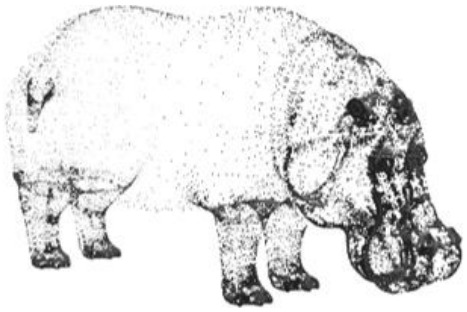


(a) Nube de puntos original

(b) Nube de puntos simplificada (Tapia-Dueñas [12])

(c) **Nube de puntos optimizada (asignación iterativa)**

Figura 47 – Comparación entre las nubes de puntos del mismo objeto 3D (Horse)



(a) Nube de puntos original

(b) Nube de puntos simplificada (Tapia-Dueñas [12])

(c) **Nube de puntos optimizada (asignación iterativa)**

Figura 48 – Comparación entre las nubes de puntos del mismo objeto 3D (Hippo)

(a) Nube de puntos original

(b) Nube de puntos simplificada (Tapia-Dueñas [12])

(c) **Nube de puntos optimizada (asignación iterativa)**

Figura 49 – Comparación entre las nubes de puntos del mismo objeto 3D (Happy Buddha)

Publicaciones derivadas

Dados los resultados obtenidos con la metodología de optimización propuesta, se tomó la decisión de elaborar un artículo de congreso para someterlo en el "Mexican Conference on Pattern Recognition" (MCPR2025), en el cual se expuso solo los dos primeros enfoques del ligado de puntos ya que la metodología en un inicio había sido concebida para comparar un enfoque que redujera la cantidad de puntos en la nube optimizada y otra que utilizara todos los puntos de la nube simplificada. En [64] se puede encontrar el artículo publicado del congreso MCPR2025 que fue celebrado en Junio del 2025.

A finales del mes de octubre del 2025, se recibió un correo de invitación para publicar en la revista indexada "Pattern Recognition Letters" (PRL) ya que el artículo publicado en el congreso MCPR2025 fue seleccionado para recibir el premio "Student Paper Award". Por tal motivo, fue que se desarrolló el tercer enfoque del ligado de puntos buscando mejorar los resultados en cuanto a la optimización del error que se habían obtenido con respecto a los primeros enfoques desarrollados. De esta manera, se elaboró un nuevo artículo siguiendo el formato de publicación del sistema Elsevier y, a la fecha de publicación de este trabajo, se sigue en la espera de recibir una segunda ronda de comentarios por parte de los revisores de la revista PRL. En la sección de Anexos se puede apreciar en la figura 126 la lista de artículos premiados e invitados a publicar en la revista PRL. Dicha lista, se puede encontrar de igual manera publicada en la página: https://mcpr2025.eventos.cimat.mx/paper_awards_mcpr2025 .

Capítulo 4: Representación y manipulación de nubes de puntos en realidad virtual

A continuación, se detalla cómo se elaboró el entorno de realidad virtual utilizado para el presente trabajo, partiendo de la mención del equipo que se utiliza para este propósito, para luego mencionar la explicación de cómo es que resulta la programación del entorno virtual, la configuración de los botones de los controles y llegando al punto de explicar cómo se integraron y/o transformaron las nubes de puntos optimizadas para poderlas utilizar en el entorno de realidad virtual creado.

4.1: Equipo utilizado

A continuación, se mencionan tanto el hardware y las herramientas de software que se utilizan como la justificación de por qué se selecciona cada una de ellas:

- **Unreal Engine 5 (UE 5)**: Se utiliza este motor de desarrollo para explorar las opciones que presenta para el uso de nubes de puntos 3D, con la finalidad de entenderlo y, de ser necesario, proponer nuevas herramientas para poder utilizar todo su poder en trabajos futuros en un sistema que utilice la simulación física avanzada que ofrece al usar modelos con nubes de puntos.
- **Meta Quest 3s:** Se utiliza esta opción de gafas de realidad virtual por sus dos puntos fuertes: buen soporte e integración con el motor UE 5 y facilidad económica de acceder a ellas, en comparación con otros modelos.
- **Blender**: Se utiliza este software de modelado 3D para transformar las nubes de puntos optimizadas que se obtienen con la finalidad de explorar las opciones que presenta y, en caso de ser necesario, identificar posibles áreas de mejora para desarrollar herramientas que puedan ser de utilidad en el manejo de puntos 3D sobre él.

Cabe mencionar que se utilizó el mismo equipo de cómputo que se utilizó en el capítulo 3. Aunque, para el uso de UE 5, fue necesario implementar una tarjeta de video Nvidia Rtx 4070 en el equipo de cómputo para utilizar sus funciones.

4.2: Creación y diseño del entorno de realidad virtual

Para crear un proyecto de realidad virtual en UE 5 se tienen dos opciones: utilizar el "starter content" que el mismo motor trae por defecto en cada uno de sus proyectos para tener las funcionalidades básicas listas y así empezar a interactuar con el entorno virtual o hacer la

configuración y la programación de las funcionalidades básicas paso a paso para emplear lo que se necesite de acuerdo con el tipo de proyecto que se quiera crear.

En este caso, se opta por la segunda vía de configurar y programar las funciones básicas para entender cómo es que funciona el motor y, de esta manera, ir forjando el conocimiento necesario para avanzar al punto de crear proyectos más avanzados en trabajos futuros. Aunque, igual se hace uso del "starter content" para utilizar materiales y objetos que vienen incluidos en él y son de utilidad al momento de diseñar un entorno virtual personalizado.

Creando el proyecto

Al momento de abrir el programa de UE 5, se puede apreciar que hay apartados para ver los proyectos recientes y para crear nuevos, en esos apartados, se puede encontrar varias veces la opción "Virtual Reality" para crear aplicaciones de realidad virtual, en específico, se puede encontrar en el apartado "Games" y en el apartado "Simulation". Sin embargo, para los propósitos de este proyecto que solo se centra en interacciones básicas con las nubes de puntos con mandos de realidad virtual, utilizar la opción del apartado "Games" será más que suficiente, ya que la opción del apartado "Simulation" incluye librerías y plugins que son más enfocados a la precisión y el fotorrealismo.

De cualquier manera, si se llegara a necesitar incluir simulación en un proyecto que fue creado en el apartado "Games", esas librerías y plugins para simulaciones pueden añadirse sin problemas después de haberse creado. En la figura 50 se muestra esta pantalla desde la que se pueden crear proyectos.

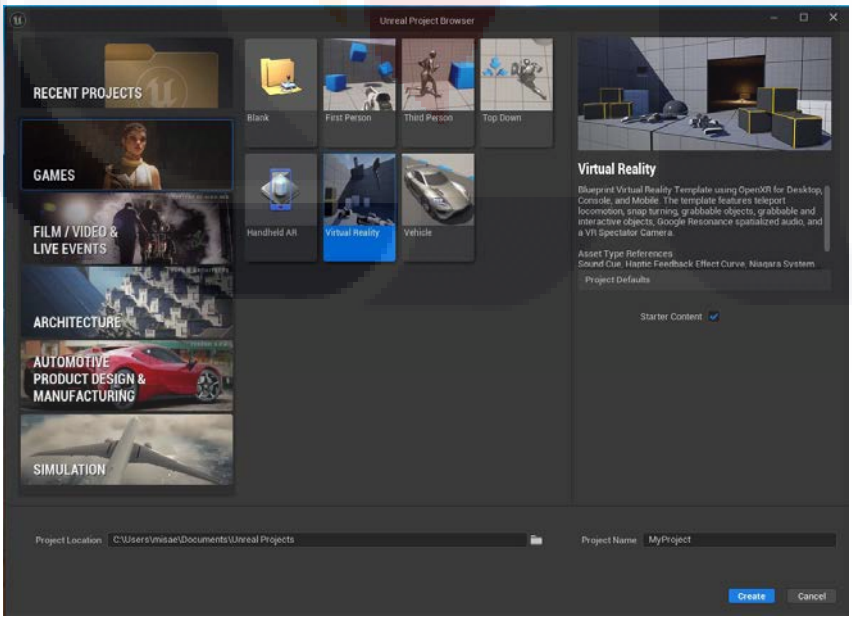


Figura 50 – Pantalla de UE 5 para crear proyectos

Una vez creado el proyecto de realidad virtual, se puede ver un proyecto inicial como el de la figura 51 en el que se mostrará el contenido que incluye el "starter content" en el editor de nivel. En este editor es donde se permite añadir elementos a un nivel, modificarlos visualmente, o incluso modificar algunos de sus atributos. En él se puede apreciar que también hay varias secciones en las que se puede ver o modificar los diferentes aspectos del proyecto. Por ejemplo, en el panel superior, se encuentran pestañas que permiten desplazarse entre el editor visual o las diferentes clases de Blueprints que se tienen abiertas; en el panel derecho llamado "Details", se encuentran las opciones para configurar objetos que se seleccionan en el editor visual y en el panel inferior llamado "Content browser" se puede tanto explorar los directorios y archivos que conforman al proyecto como crear nuevos directorios y archivos. Además, pueden añadirse otros paneles que permiten modificar o añadir otras funcionalidades en el proyecto, pero esas pueden añadirse u ocultarse en cualquier momento que se necesite.

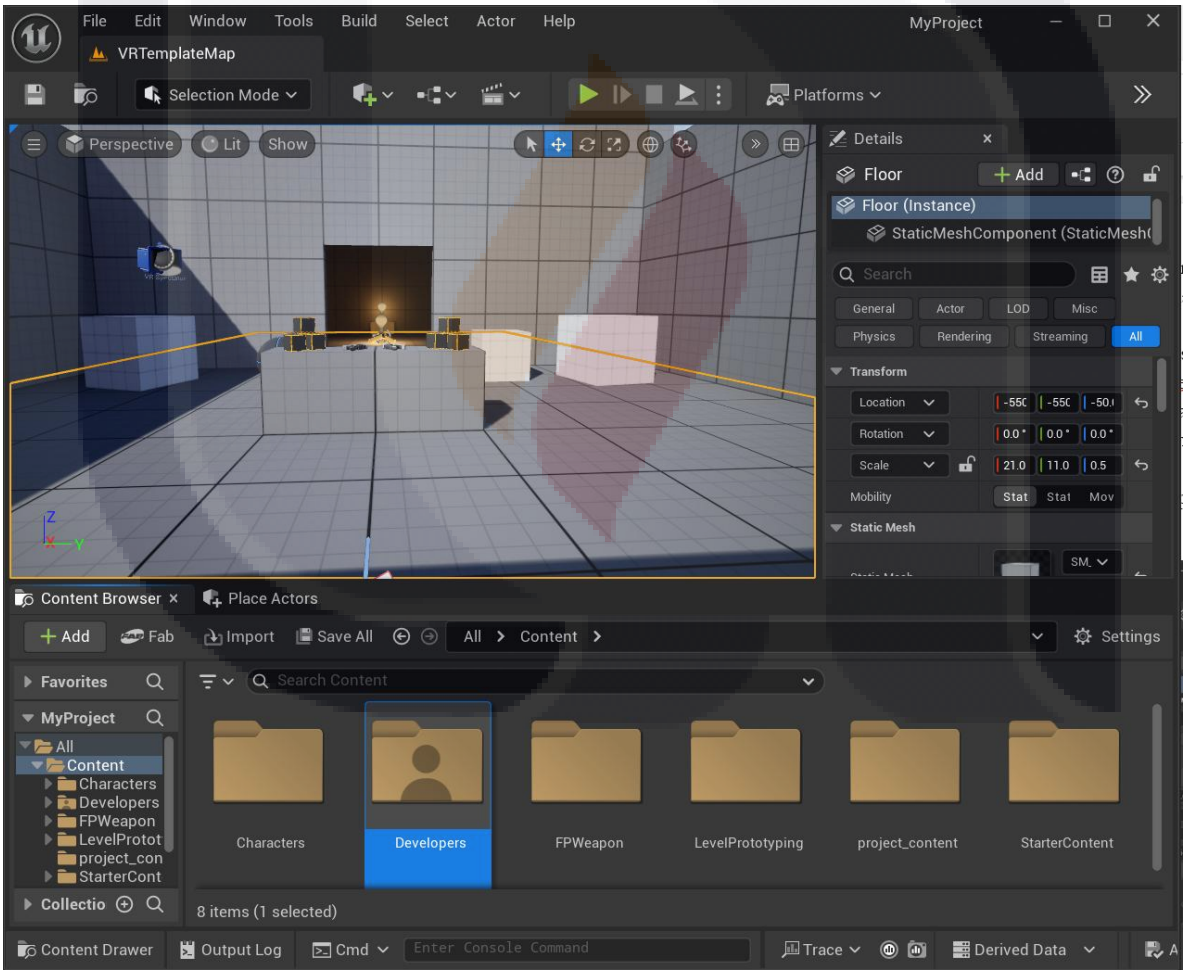


Figura 51 – Interfaz del editor de nivel en un proyecto nuevo de UE 5

Configuración y programación para el usuario (jugador)

- **Creación de clase para el jugador**

Para configurar los controles en un proyecto de UE 5, sea del tipo que sea, es necesario crear una clase de Blueprint en el proyecto de tipo "Pawn". En UE 5 hay varios tipos de clases que se pueden utilizar; cada una tiene diferentes características dependiendo de lo que se necesite programar en el proyecto a desarrollar. En este caso, el Blueprint de tipo "Pawn" es para programar aquello que hará el usuario, o el jugador, en la aplicación, por lo que es en esta clase que se debe programar las entradas de información, o sea, qué hará cada botón del control a utilizar. Ahora, por supuesto, es posible manejar diferentes clases de este estilo en UE 5, así que, al momento de crear una, se debe establecer en los "World Settings" del proyecto cuál es la que se debe ejecutar.

En la figura 52 se muestra un ejemplo de cómo se añade una clase de Blueprint directamente sobre el "Content Browser", en la figura 53 se muestran los tipos de clases que se pueden crear en un proyecto y en las figuras 54-55 se muestra la manera de configurar los "World Settings".

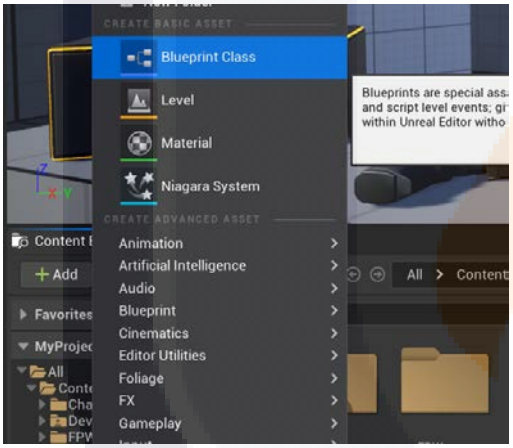


Figura 52 – Creación de clase de Blueprint

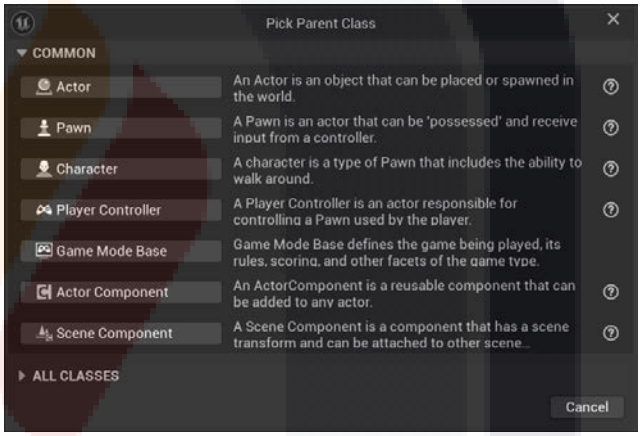


Figura 53 – Diferentes tipos de clases de Blueprint

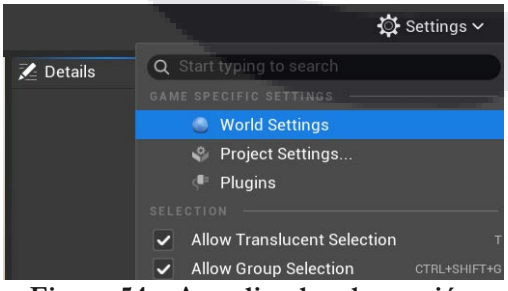


Figura 54 – Accediendo a la sección "World Settings" del proyecto

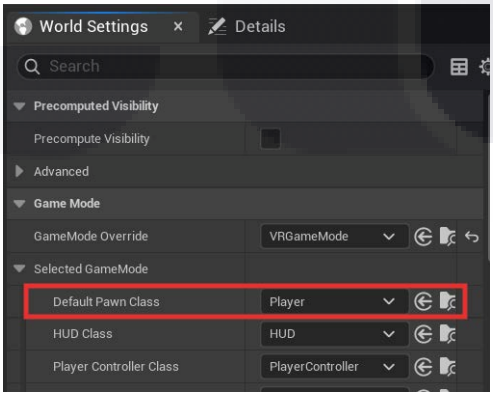


Figura 55 – Estableciendo la clase del jugador

Una vez creado el Blueprint que contendrá la programación del jugador, se procede a añadir lo básico que deberá tener para ser usado. Lo primero, es añadir un elemento de tipo "Camera"

sobre él para poder establecer hacia dónde verá el jugador cuando se ejecute el proyecto. Cabe mencionar que cuando un Blueprint es creado y abierto, se muestra el editor de Blueprints como en la figura 56, donde no hay elementos añadidos, y solamente se pueden apreciar las pestañas básicas que se manejan para editar Blueprints. Las dos más utilizadas son "Viewport", que es donde se acomodan visualmente los elementos u objetos que tendrán un Blueprint, y "Event Graph", que es donde se añade la programación.

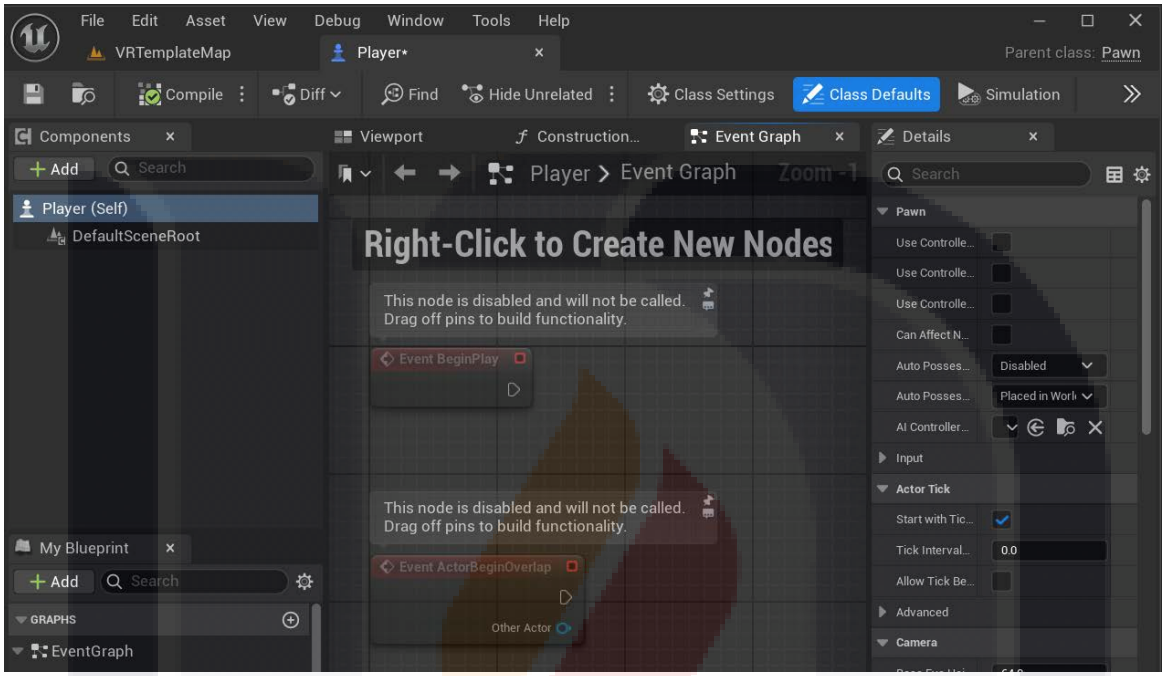


Figura 56 – Editor de Blueprint

En la figura 57 se muestra el componente "Camera" añadido a la clase Blueprint del jugador, donde se recomienda dejarla en la posición que aparece por defecto en el "Viewport". Para establecer hacia dónde ve el jugador cuando se ejecuta el proyecto, se debe arrastrar el Blueprint recién creado en el editor del nivel. La figura 58 muestra cómo se ve el Blueprint del jugador en el nivel una vez que tiene añadido el componente "Camera".

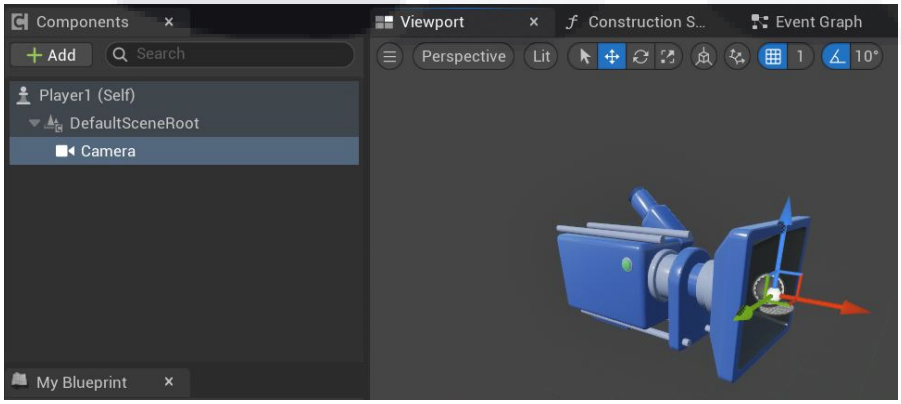


Figura 57 – Componente "Camera" añadido al Blueprint

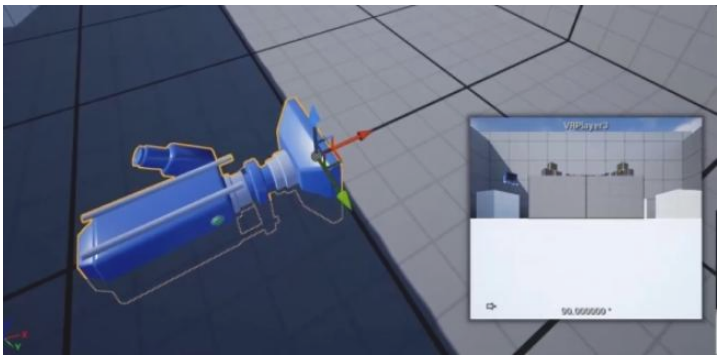


Figura 58 – Blueprint del jugador añadido en el editor del nivel

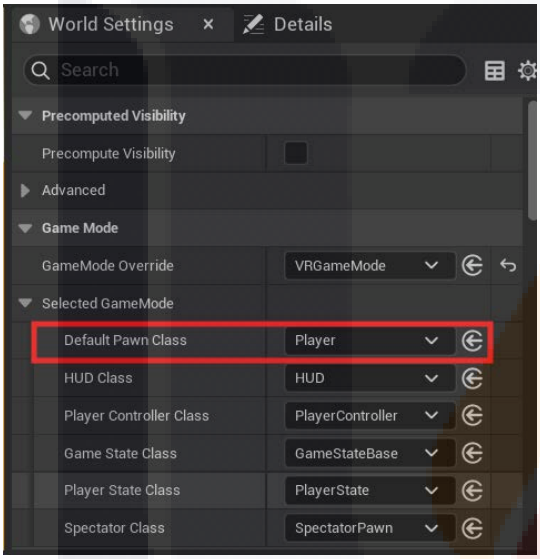


Figura 59 – Configuración de la cámara en el editor del nivel

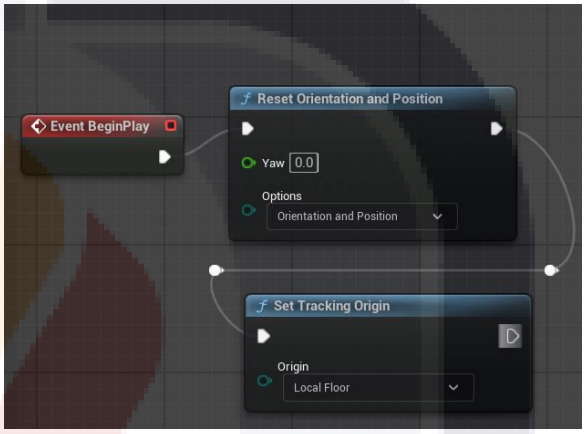


Figura 60 – Configuración de la cámara en el Blueprint del jugador

Cabe mencionar que es necesario ajustar el atributo "Auto Posses Player" en "Player 0", para que al ejecutar el proyecto, se ejecute sobre la cámara y no sobre donde se está ejecutando el proyecto. Esto se consigue seleccionando el Blueprint del jugador en el editor del nivel y yendo a la pestaña "Details" (Figura 59). Además, suele ser común que en el evento BeginPlay (en el Blueprint del jugador) se conecten los nodos "Reset Orientation and Position" y "Set Tracking Origin" (Figura 60). El primero, reinicia la orientación y la posición donde apuntan los visores de realidad virtual (apuntan hacia donde apunte el componente "Camera" después de ejecutar este nodo). El segundo también asigna una posición, pero la opción "origin" puede establecer si la altura se cambia al suelo o a otro lugar.

- **Configuración de las entradas con los controles de las gafas**

Para configurar entradas con controles de cualquier tipo, se necesita crear un "Input action" por cada botón a utilizar y luego un "Input Mapping Context" para coordinarlos en el "Content

Browser". En la figura 61 se muestra la manera de crear estos archivos en el "Content Browser" y en la figura 62 la visualización de los archivos creados.

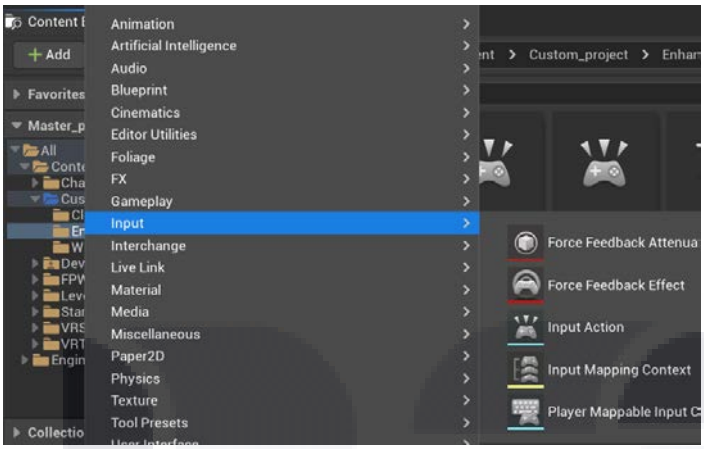


Figura 61 – Creando archivos para configurar entradas con los controles

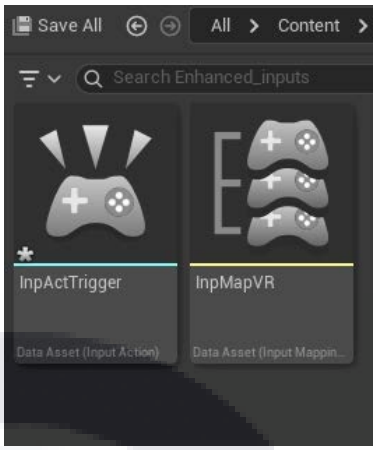


Figura 62 – Archivos creados para la entrada de los controles

Para entender cómo configurar las entradas con los controles de los visores, primero hay que entender los diferentes botones que tienen. En la figura 63, se muestra una imagen de los controles de los visores de las gafas Meta Quest. En ella se puede ver que los controladores tienen: palancas (joysticks), botones de acción rápida (botones A, B, X, Y) y gatillos que tienen una acción dependiendo de cómo sean presionados (botones laterales que se accionan con los dedos del corazón y posteriores que se accionan con los dedos índice).

Figura 63 – Controles de gafas Meta Quest

Lo anterior es importante de entender porque dentro de los archivos "Input Action" se maneja un parámetro de configuración llamado "Value Type" que determina el valor que se recibe al presionar, o accionar, tanto los botones como los joysticks en los controles (figura 64). El tipo de valor "Digital(bool)" suele ser el utilizado en los botones A, B, X, Y, ya que solo arroja verdadero cuando se acciona el botón y falso cuando no. El tipo de valor "Axis1D(float)" suele ser el utilizado en los gatillos, ya que arroja un valor continuo dependiendo de cuánto se esté

accionando un botón y, en el caso de estos botones que tienen un rango para accionarse, puede utilizarse para recibir un valor que refleje cuánto están siendo presionados. Y el tipo de valor "Axis2D(Vector2D)" suele ser utilizado en los joysticks, ya que este valor regresa los valores de cuánto se está accionando algo en dos dimensiones y, en un joystick que puede arrojar valores de desplazamiento en 'x' o en 'y', con este tipo de valor se puede saber cuánto se están desplazando vertical u horizontalmente.

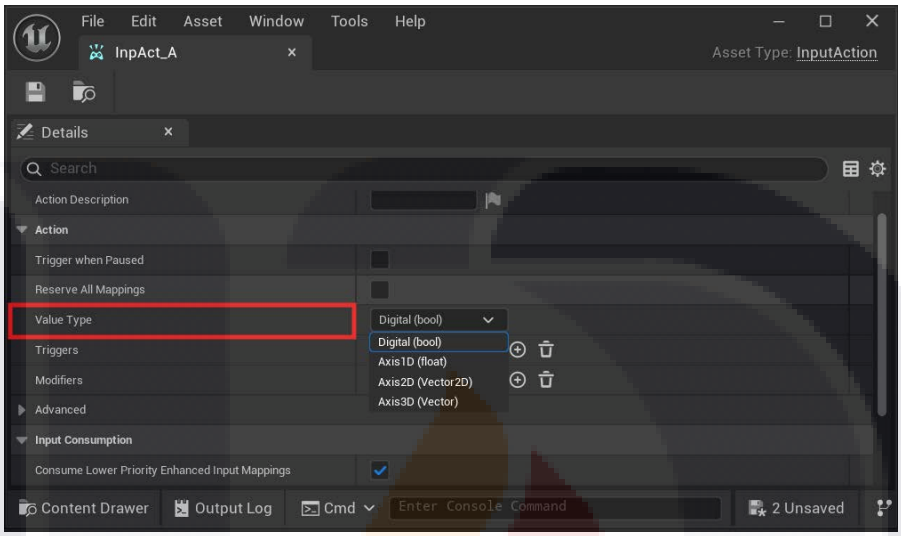


Figura 64 – Tipos de valores para los botones de los controles

Una vez creados los archivos de configuración para todos los botones, en el archivo "Input Mapping" se debe referenciar cada uno de esos archivos de configuración con los botones que corresponden (figura 65). Al asignar los botones, se podrá apreciar que en UE 5 los tipos de controles están seccionados dependiendo del tipo de control; en este caso, están en la categoría "Oculus Touch". Dentro de esa categoría, los botones del control izquierdo están denotados por la etiqueta (L) y los del control derecho por la etiqueta (R). Para seleccionar los botones que se presionan con los dedos índices, se utiliza las opciones llamadas "Trigger"; para seleccionar los botones que se presionan con los dedos corazón, se utilizan las opciones llamadas "Grip"; para cada uno de los botones de acción rápida (A, B, X, Y) hay una opción con su etiqueta; para seleccionar el joystick, se utilizan las opciones llamadas "Thumbstick 2D-Axis" (para detectar valores en 'x' y en 'y'), "Thumbstick X-Axis"" (para solo detectar valores en 'x') o "Thumbstick Y-Axis" (para solo detectar valores en 'y') dependiendo de qué es lo que se quiera detectar con cada joystick. Esto último debe de coincidir con la configuración "Value Type" del archivo "Input action" del botón, si solo se va a trabajar con un eje de los joysticks, el tipo de valor debiera ser "Axis1D(float)".

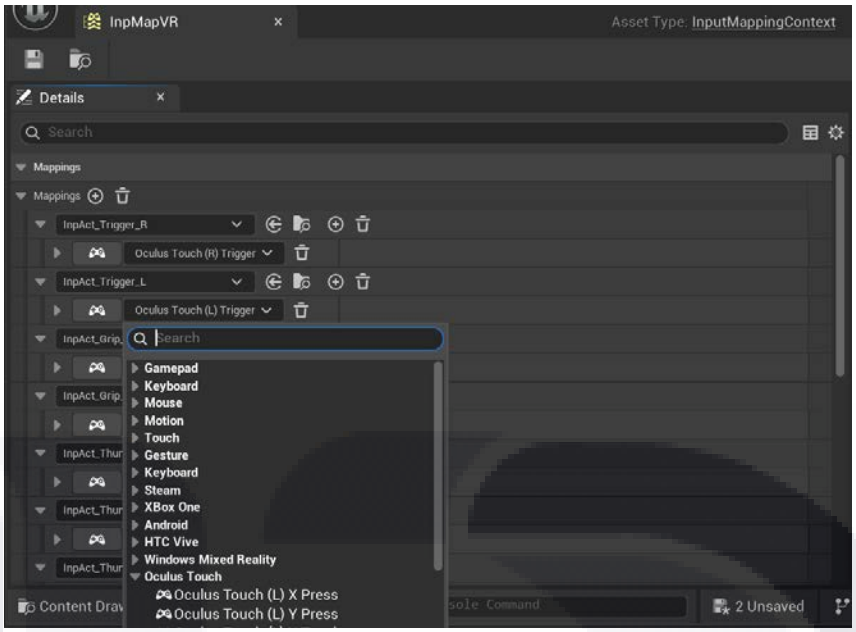


Figura 65 – Enlazando los botones con sus archivos de configuración en el "Input Mapping"

Para que el proyecto tome en cuenta estos archivos de configuración para las entradas, es necesario agregar el archivo "Input Mapping Context" recién creado en el "Project Settings", más en específico, en la sección "Engine/Enhanced Input", en el apartado "Default mapping context" (figura 66).

Por último, para poder programar las acciones que desencadenarán los botones al accionarse, es necesario hacer que el Blueprint del jugador pueda usar los "Input Mapping Context" que hay en el proyecto; para esto se conecta el nodo del método "Add Mapping Context", que recibe el controlador del jugador con el nodo del método "Get Player Controller", en el flujo del nodo evento "Begin Play" como se muestra en la figura 67. Después de esto, ya pueden añadirse nodos de eventos del tipo "EnhancedInputAction" (figura 68), cada uno de esos representa cada botón y joystick, por lo que aquello que se agregue y se conecte después de ese evento, se interpreta como las acciones que desencadena cada botón al ser presionado, o desplazado en el caso del joystick.

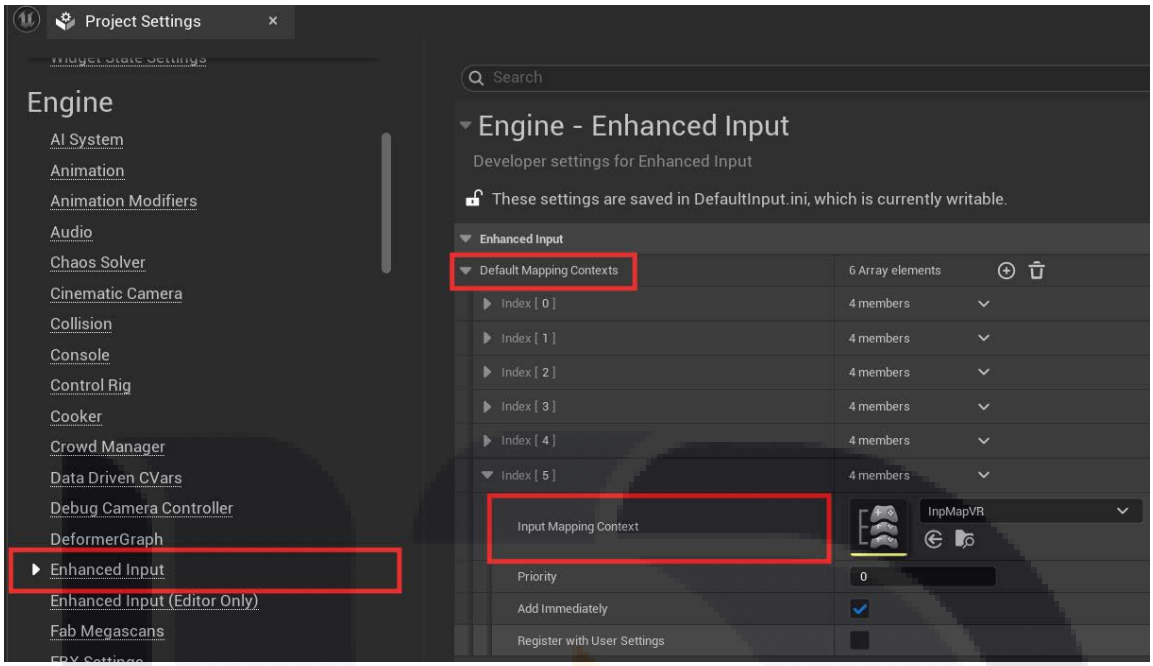


Figura 66 – Añadiendo el archivo "Input Mapping" al "Project settings"

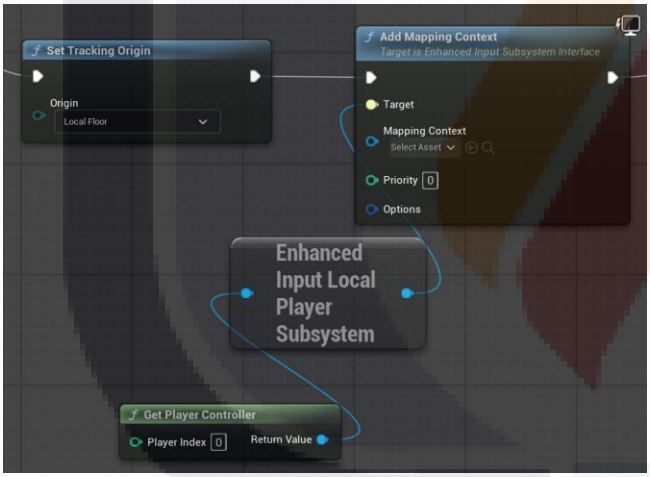


Figura 67 – Conectando los "Input Mapping Context" al Blueprint del jugador

Figura 68 – Evento que corresponde a un botón de los controles

- **Manos virtuales y sus animaciones**

Para detectar el movimiento que hacen los controles de las gafas, es necesario agregar el componente "MotionController" al Blueprint del jugador; debe tenerse cuidado de que sean hijos del "DefaultSceneRoot" como tal y no del componente "Camera" (figura 69). Hay que agregar uno por cada control a utilizar y hay que especificar a cuál control está conectado cada "MotionController" en la pestaña "Details" en la sección "Motion Controller" en el campo "Motion Source" (figura 70).

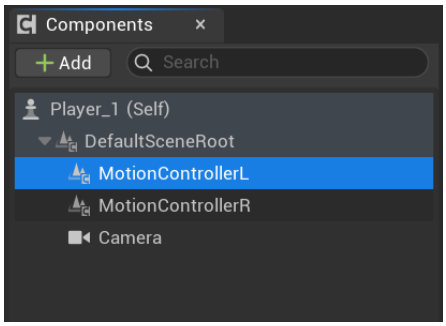


Figura 69 – Componentes "MotionControllers" en el Blueprint del jugador

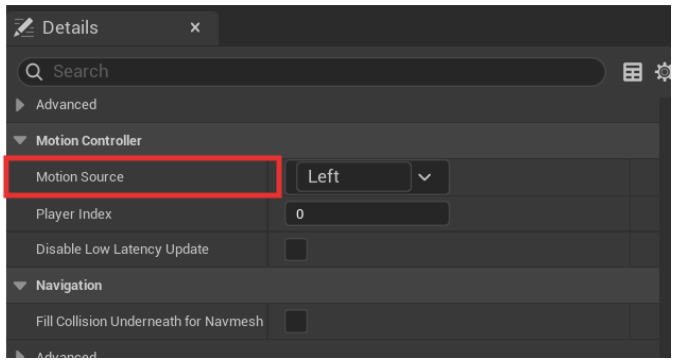


Figura 70 – Estableciendo el control del que se detecta el movimiento

Para poder añadir manos virtuales que puedan tener movimiento en los dedos, es necesario crear un Blueprint del tipo "SkeletalMeshComponent" en el proyecto (figura 71), en esa clase, solo es necesario añadir un objeto de tipo "Skeletal Mesh" en la pestaña "Details" en el campo "Skeletal Mesh Asset" (figura 72). En este caso, se utiliza un modelo de manos que viene dentro del "starter content" de UE 5 llamado "SKM_MannyXR_left". Además, dentro de este Blueprint, se crea una variable de tipo booleano llamada "Is_mirror" que sirve para distinguir la mano derecha de la izquierda cuando se agreguen las animaciones.

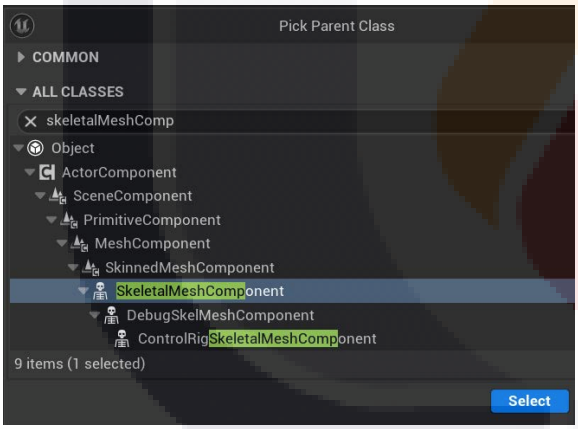


Figura 71 – Creando Blueprint del tipo "SkeletalMeshComponent"

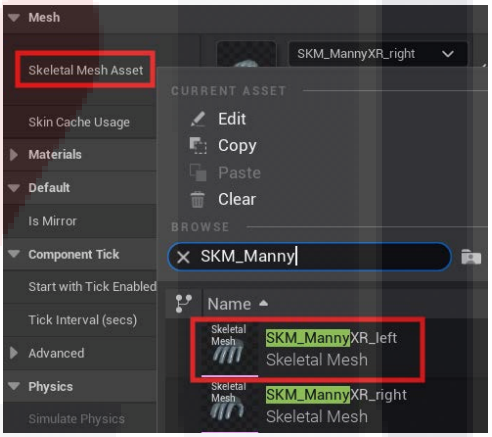


Figura 72 – Añadiendo modelo 3D de manos virtuales

Luego, el Blueprint creado con el modelo de la mano se agrega como componente en el Blueprint del jugador. Se agrega uno por cada control como hijos del "MotionController" correspondiente, y se alinean en el "Viewport" (figura 73). Claro, para el que es utilizado para la mano izquierda se puede cambiar el "Skeletal Mesh" de la mano derecha para utilizar el adecuado en "Details", además de que tiene que marcarse la casilla "Is_mirror" en la sección "Default", para que este booleano esté activo desde siempre y diferencie la mano izquierda de la derecha (figura 74).

Para hacer que cada dedo se mueva al presionar un botón en específico, es necesario que se establezcan las animaciones para ello. Para crearlas, se debe crear un "Animation Blueprint"

sobre el "Content Browser" del tipo "SK_MannequinsXR" (figura 75), que es el que corresponde al de las manos en RV.

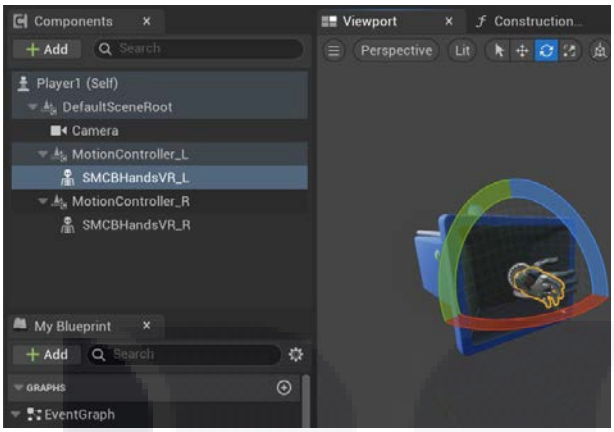


Figura 73 – Añadiendo y alineando las manos virtuales

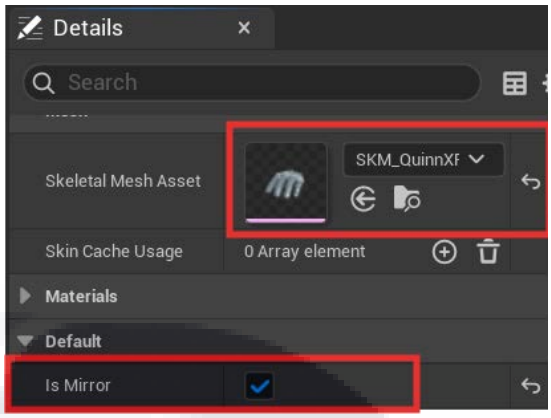


Figura 74 – Configuración de mano izquierda

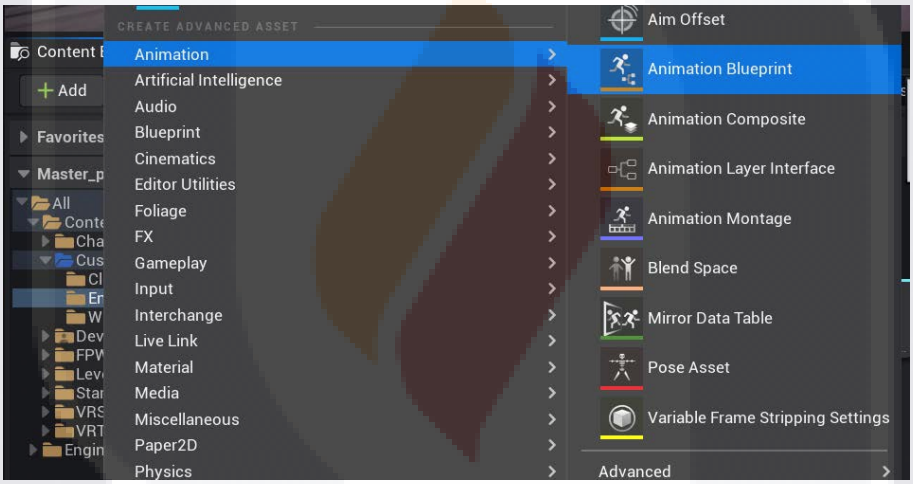


Figura 75 – Creación de un "Animation Blueprint"

Dentro del Blueprint de la animación, podremos encontrar un listado de las animaciones que ya vienen por defecto (figura 76). Al arrastrarlas al "AnimGraph" se podrán utilizar como nodos para programar su ejecución.

Es necsario entender que para programar, por ejemplo, la animación de señalar con un dedo y abrir la mano, se pueden conectar los nodos de las animaciones "A_MannequinsXR_Idle_Right" (mano abierta) y "A_MannequinsXR_Grasp_Right" (mano apuntando con el dedo índice y pulgar estirados) al un nodo "Two Way Blend", que decidirá la animación utilizada dependiendo del valor "Alpha" que reciba. Para ello, se suele crear una variable que recibe el valor que se obtiene al presionar un botón; en este caso, se asigna a la variable "Alpha_grip" el valor que se

recibe del gatillo que se presiona con el índice, ya que esta animación sirve como una representación de agarrar algo con la mano. Cuando "Alpha_grip" vale 0, se ejecuta la animación del nodo A (animación de mano abierta), y cuando vale 1, la del nodo B (animación de mano apuntando), y en sus valores intermedios transiciona entre las animaciones.

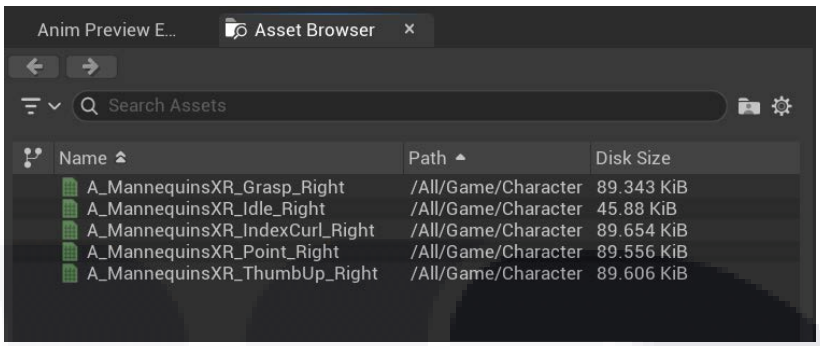


Figura 76 – Animaciones por defecto de las manos

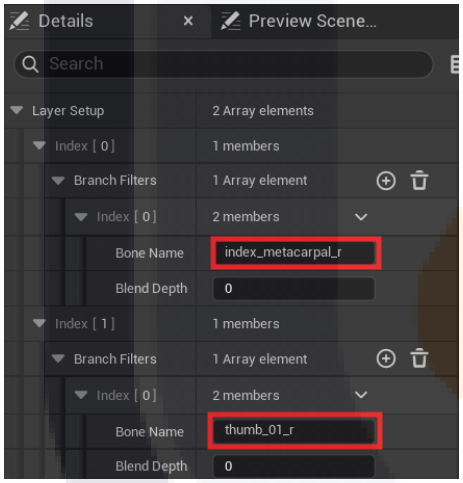


Figura 77 – Partes del "Skeletal mesh" de la mano a animar con "Layered blend per bone"

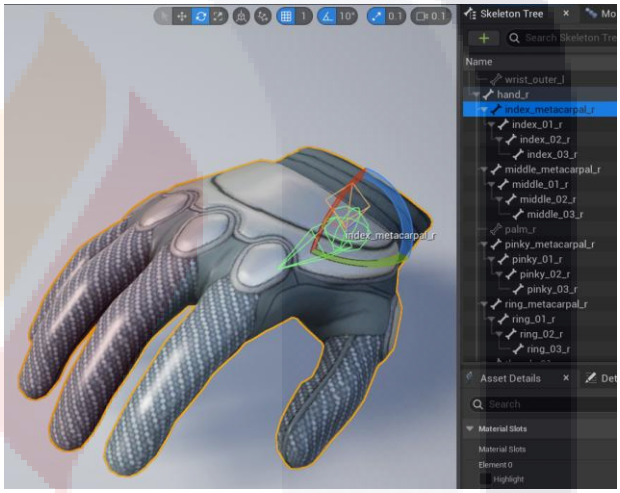


Figura 78 – Nombres de los huesos en el "Skeletal mesh" de la mano utilizada

Ahora, para lograr que se puedan usar más animaciones a la vez es necesario utilizar el nodo "Layered blend per bone", que puede combinar varias animaciones al mismo tiempo. En su conexión de entrada "Base Pose" se conecta el nodo "Two Way Blend" que ya trae la primera animación. En las otras conexiones de entrada "Blend poses" se conectan las animaciones que se quieren agregar, en este caso se conectan "A_MannequinsXR_IndexCurl_Right" (flexionar el dedo índice) y "A_MannequinsXR_ThumbUp_Right" (estirar el pulgar) para poder mover todos los dedos de la mano simultáneamente, y en las conexiones de entrada "Glend Weights" reciben los valores que determinan las transiciones de dichas animaciones, "Alpha_thumb" para la del pulgar y "Alpha_trigger" para la del índice. Además, en la pestaña "Details" (figura 77) en el campo "Layer Setup", se tiene que especificar el nombre de la parte del "skeletal mesh" que se quiere animar en las conexiones que no sean la "Base pose"; esos nombres se encuentran al abrir el modelo del "Skeletal Mesh" utilizado y se utilizan los nombres que tienen las partes anteriores

a las articulaciones de los nudillos (figura 78). En este caso, se "utiliza index_metacarpal_r" para la animación del dedo índice y "thumb_01_r" para la animación del pulgar. Es importante que el orden en el que se agregan las partes del "skeletal mesh" a animar coincida con el orden de las animaciones conectadas en "Blend Poses".

Una de las bondades que ofrece UE 5, es que se pueden utilizar las mismas animaciones para ambas manos. Para esto, se debe crear un "Mirror", este funciona como un espejo de animaciones que sirve para mover los dedos de ambas manos utilizando los mismos nodos de animación. Para crearlo, sobre el "Content Browser", se crea un archivo del tipo "Mirror Data Table" (figura 79) que se puede dejar con sus valores por defecto.

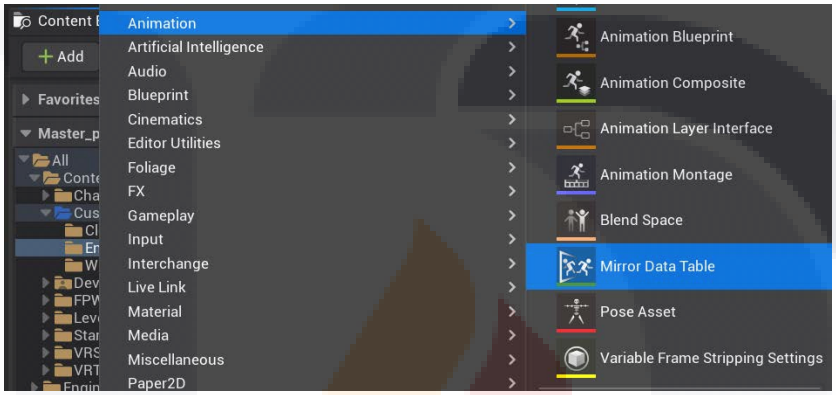


Figura 79 – Creando un archivo "Mirror Data Table"

Luego, es necesario agregarlo al Blueprint de la animación como nodo, debe estar conectado después del nodo "Layered blend per bone" para que se utilicen las animaciones de todos los dedos de la mano. Toda la programación descrita de las animaciones se muestra en la figura 80."

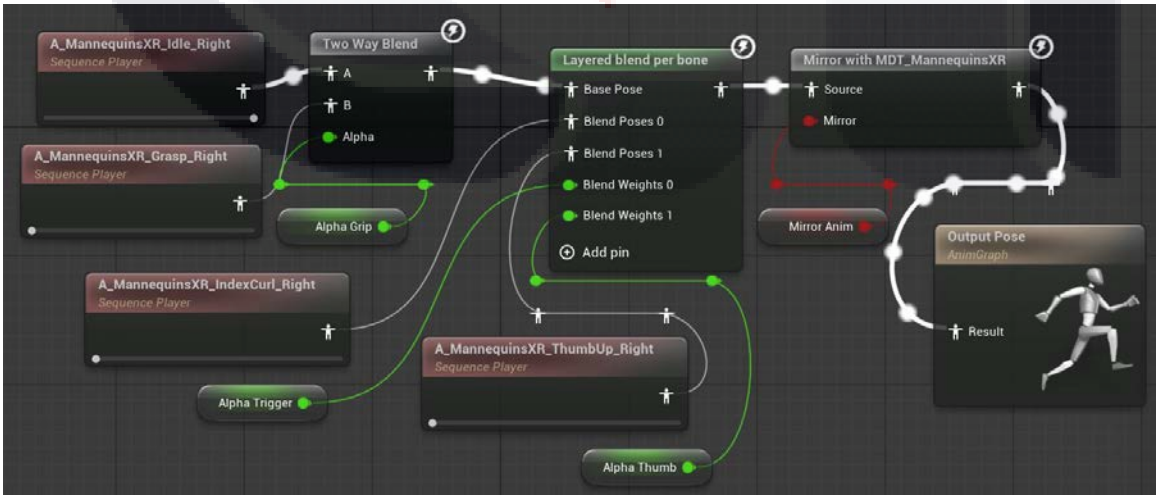


Figura 80 – Programación de la animación de las manos

Sin embargo, este nodo tiene que recibir un valor de tipo booleano que es el que establece si se tiene que ejecutar la animación de la mano derecha o la izquierda. Para esto, se debe añadir al "Event Graph" del Blueprint de la animación el nodo del método "Get Owing Component" que toma el Blueprint del que se está ejecutando de las manos virtuales, después hace un casteo para poder tomar el valor booleano que se establece en el Blueprint ejecutor (Blueprint las manos con su valor "Is_mirror" que se establece en el Blueprint del jugador), de manera que queda como se muestra en la figura 81.

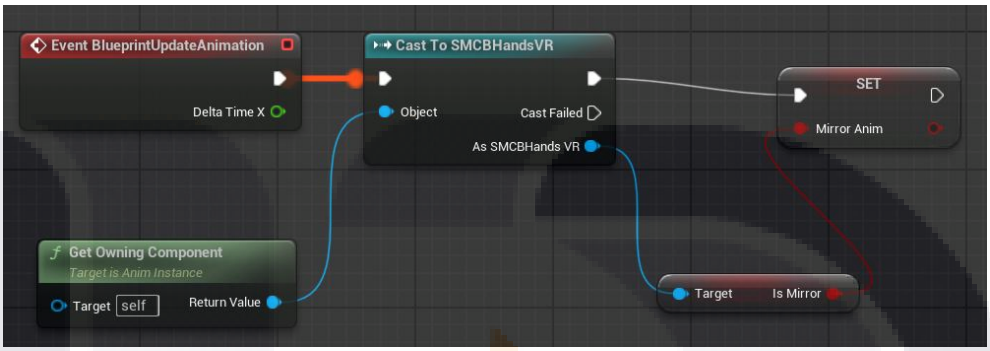


Figura 81 – Programación del "Event Graph" en la animación de las manos

Como se ha estado mencionando, estas animaciones necesitan recibir valores que se establecen tanto al presionar los botones de los controles como por parte de la variable creada en el Blueprint de las manos. Por lo que estas animaciones tienen que ser invocadas desde el Blueprint del jugador. Para esto, primero se deben establecer unas configuraciones necesarias para conectar los Blueprints de las manos, con las animaciones creadas. De esta manera, se debe seleccionar cada Blueprint desde el panel de "Components" en el Blueprint del jugador, para que en la pestaña "Details" se pueda seleccionar la animación en "Animation", "Anim Class" (figura 82).

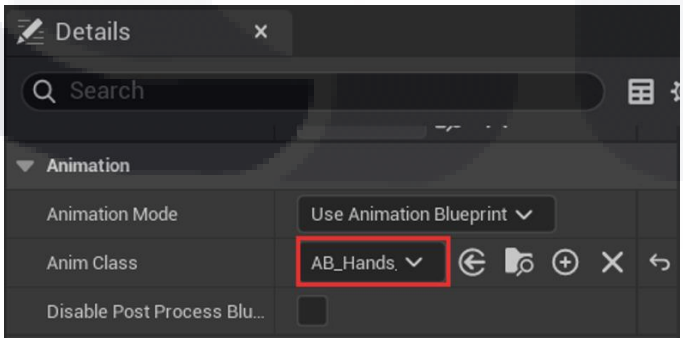


Figura 82 – Asignando animación al componente de las manos

Ahora, al arrastrar el componente del Blueprint de alguna mano al "Event Graph", este pasa a ser un nodo con solo una conexión de salida, de ella se puede obtener el nodo del método "Get

Anim Instance" para obtener la animación a la que está vinculado; y de ese nodo, se puede obtener el nodo "Cast to blueprint_animation" para poder enviar el valor que esté generando un botón al ser presionado; este envío se hace al obtener un nodo set desde el nodo de casteo, el cual apuntará a la variable alfa de la animación que se quiere ejecutar. Por ejemplo, en la figura 83 se puede apreciar la programación del gatillo que se acciona con el dedo índice del control derecho, por lo que se parte del nodo evento que representa su "Input Action" para, de la conexión de salida "Triggered", que se activa cuando el gatillo es presionado con cualquier intensidad y arroja el valor que genera constantemente, se ejecute el nodo de casteo de la animación y para, del nodo de salida "Action Value" enviar el valor que genera el botón en el nodo set a la animación. Como se pretende utilizar la animación del índice con este botón, en el nodo set se apunta a la variable "Alpha_Trigger" que es la que determina la propia animación del índice.

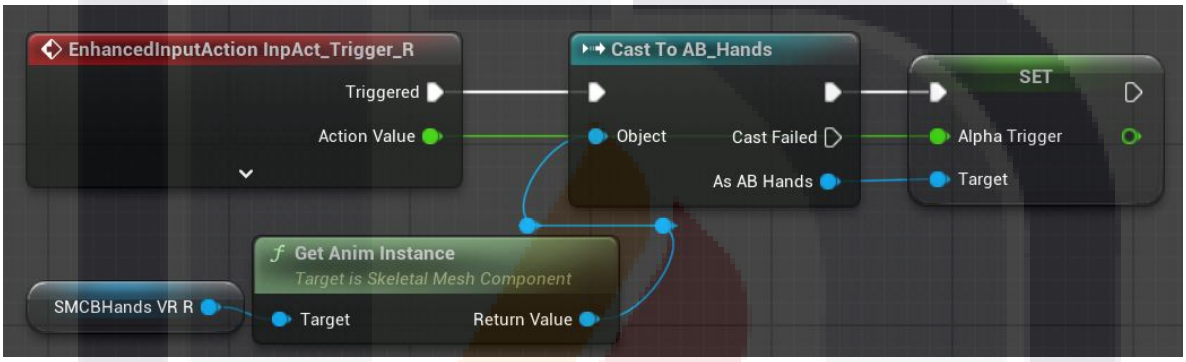


Figura 83 – Añadiendo nodos de la animación al nodo de un botón en el Blueprint del jugador

De esta manera es como se puede establecer qué animación es usada en la activación de cada botón. En este proyecto, además de la animación del dedo índice, se usa la animación de agarrar cosas (determinada por la variable "Alpha_grip") con el gatillo accionado por el corazón y el pulgar con el joystick (determinado por la variable "Alpha_thumb") de cada control.

- **Programación del movimiento del jugador en el entorno virtual**

El movimiento del jugador en el entorno virtual está compuesto de dos acciones principales: trasladarse por el entorno y rotar sobre su propio eje. Ambas acciones son programadas desde el Blueprint del jugador.

Para la primera acción del movimiento, se crean dos funciones que se encargan de teletransportar al jugador por el escenario: una función para detectar ubicaciones disponibles para la teletransportación y otra para actualizar la ubicación del usuario.

Para la primera función, que es nombrada "FuncLineTrace", se usa un componente de tipo "Arrow" que se ingresa como entrada de esta; esto se puede establecer en el panel "Details" al

crear la función mencionada (figura 84). Cabe mencionar que dicho componente se debe agregar en el "Viewport" de manera que sea hijo de la mano que podrá hacer la teletransportación, es recomendable que quede en la misma dirección en la que apunta la mano y que esté después de la posición de esta, porque puede chocar con las colisiones que la mano pueda llegar a tener.

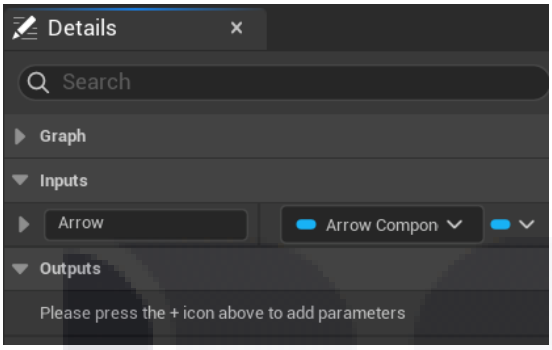


Figura 84 – Atributo de entrada para función de detección de ubicaciones

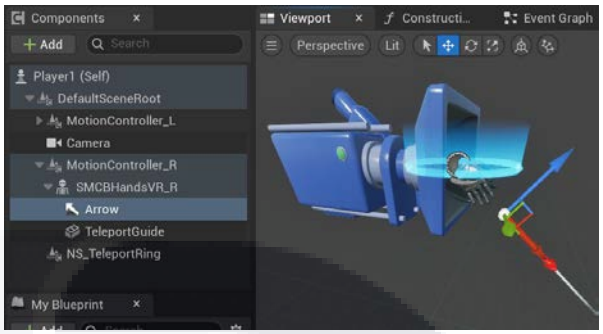


Figura 85 – Componentes utilizados en la teletransportación

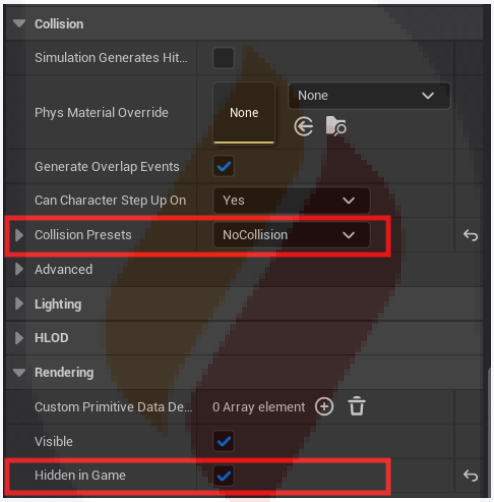


Figura 86 – Atributos para ocultar elementos y quitarles colisiones

Otros componentes que opcionalmente se pueden añadir por pura cuestión estética son el "NS_TeleportRing", que es un elemento del "Starter pack", y un componente de tipo "Cylinder", que se nombra como "Teleport_guide"; la idea es que ambos estén ocultos al iniciar la ejecución del proyecto y sin poder mover otros objetos. El componente "NS_TeleportRing" puede quedar a la par del componente "Camera" y "Teleport_guide" también debe ser hijo del componente de la mano que se encargará de hacer la teletransportación. Todo esto se puede apreciar en la figura 85, y en la figura 86 se muestran las opciones de "Details" que permiten ocultar los dos últimos elementos durante la ejecución del proyecto (atributo "Hidden in Game") y les deshabilitan la capacidad de interactuar físicamente con otros objetos ("Collision Presets" en "NoCollision").

La lógica que se debe de llevar para detectar la dirección hacia la que apunta una mano es primeramente obtener la dirección local de la mano, o de la flecha, con el método "Get World Location", para sumarle un vector de longitud n que apunte a su dirección con el método "Get Forward Vector", y con el método "Line Trace By Chanel" detectar aquello que se interpone desde el punto de origen hasta el punto al que llegue el vector creado. Esto se muestra en la figura 87, que corresponde a la programación dentro de la función creada.

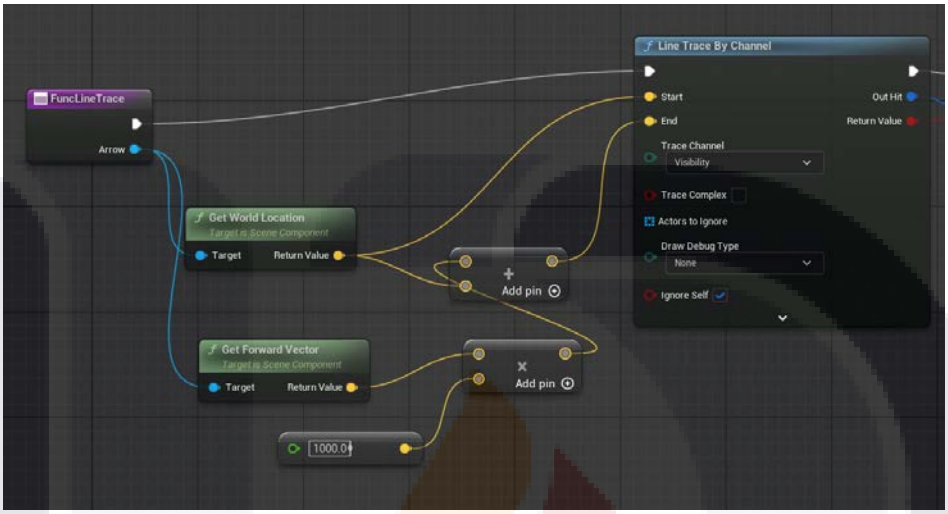


Figura 87 – Detectando objetos con el método "Line Trace By Channel"

Ahora, de la salida del "Line Trace by Channel" se puede obtener tanto la información de lo que llega a detectar, con la conexión de salida "Out Hit", como un booleano que indica si realmente se está detectando algo, con el nodo de salida "Return value". En base a esto, se puede desarrollar la siguiente lógica.

Con la conexión de salida "Out Hit", se puede obtener el nodo del método "Break Hit Result" con el que se pueden seccionar los datos de lo que se esté detectando. Las dos salidas de este método que se utilizan son "Location", que da la ubicación del objeto detectado, y "Normal", que da un vector del objeto detectado.

Utilizando la salida "Normal", se puede emplear el método "Brak Vector" para obtener cada uno de los valores que hay en él y el método "In Range (Float)" para validar que el objeto detectado tenga un valor en 'z' que se encuentre en el rango [0.75, 1]. Este rango, en este eje, indica que la superficie del objeto detectado no tenga una inclinación vertical, ya que, en una lógica de desplazamiento en un espacio real, alguien no debería ser capaz de moverse a una superficie vertical.

Empleando todo lo anterior, se puede construir la lógica que termina de delimitar esta función. Ya que se pueden emplear las siguientes validaciones: si se está detectando un objeto y el objeto en cuestión está posicionado de manera horizontal, entonces se puede guardar tanto una variable

local del tipo booleano ("Can_Teleport") para determinar si se puede dar la teletransportación como la ubicación del objeto ("Location_Teleport") para mover al jugador a esa posición (figura 88). Y, de manera visual y estética, después de asignar las variables necesarias, se puede tanto mostrar los elementos "NS_TeleportRing" y "Teleport_guide", como asignarle la ubicación que detecta "Line Trace by Channel" al elemento "NS_TeleportRing"; de esta manera, el jugador tiene una referencia visual de la ubicación a la que se estaría moviendo (figura 89).

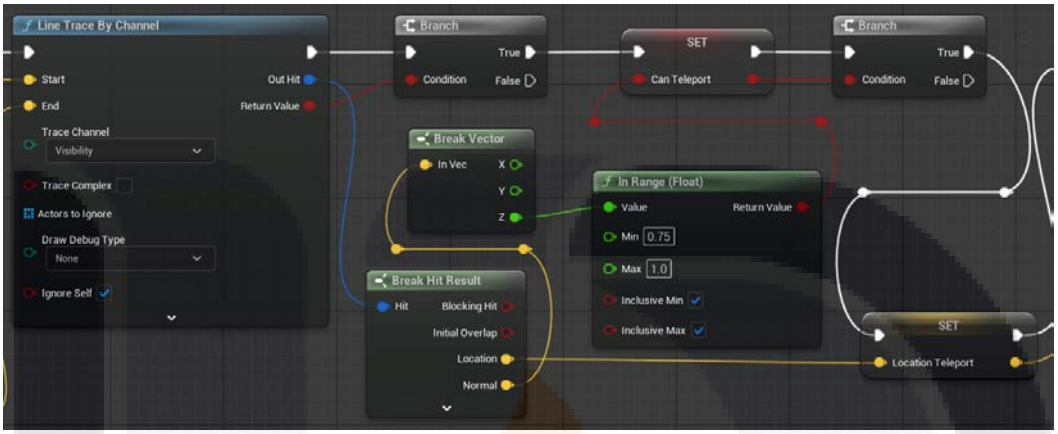


Figura 88 – Validando tanto la detección de objetos como su inclinación

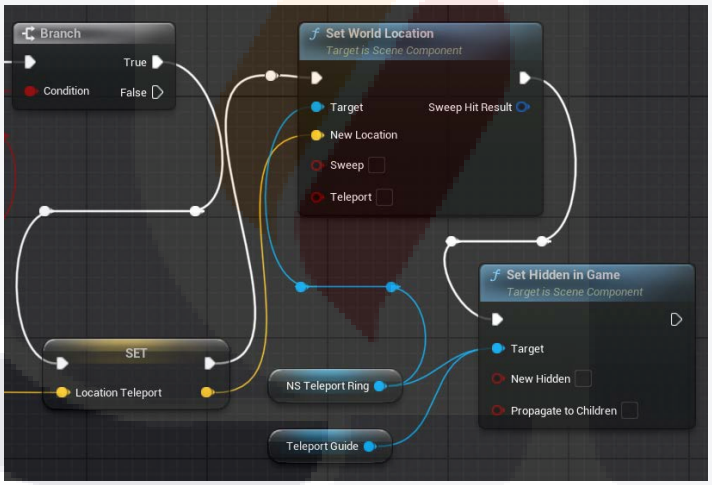


Figura 89 – Mostrando elementos de referencia visual

Para la segunda función de la acción del jugador de desplazarse por el entorno, llamada "FuncTeleport", simplemente hace la validación de si el actor puede moverse a una superficie, con el booleano "Can_Teleport" anteriormente asignado, para ejecutar el nodo de la función "Teleport" en la cual se recibe tanto la nueva ubicación del jugador, almacenada en "Location_Teleport", como la rotación actual del jugador, obtenida con el método "Get Actor Rotation". Además, como esta segunda función delimita el final de la acción del usuario de moverse por el mapa, al final de ella, tanto si el jugador se desplaza como si no lo hace, se vuelven

a ocultar los elementos visuales mostrados en la primera función ("NS_TeleportRing" y "Teleport_guide") con el método "Set Hidden in Game". En la figura 90 se muestra el resultado visual de esta segunda función.

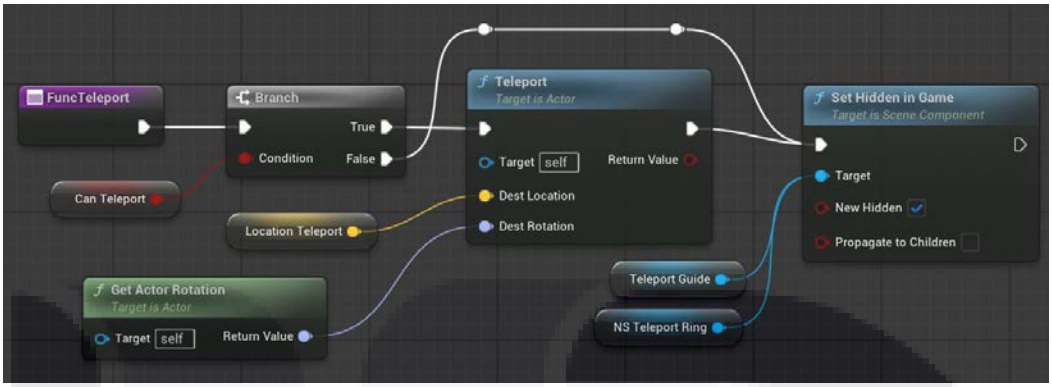


Figura 90 – Función para teletransportar al jugador

Para ejecutar esta primera acción de movimiento se emplea el joystick derecho. Para ello, simplemente se ejecutan ambas funciones creadas con las diferentes conexiones de salida que tiene el nodo del botón. Por ejemplo, con la conexión "Triggered" que ejecuta constantemente lo que tenga conectado, se agrega la primera función para detectar las ubicaciones a las que el jugador se puede teletransportar, queda conectado después de los nodos de la animación del dedo correspondiente; y con las conexiones de salida "Canceled" y "Complete", que se detonan cuando un botón deja de ser accionado o, en este caso, el joystick deja de ser desplazado, se conecta la segunda función de la teletransportación. Cabe mencionar que en este caso el joystick derecho es configurado para solo detectar los valores que arroja en su desplazamiento sobre el eje "y". Todo esto se muestra visualmente en la figura 91.

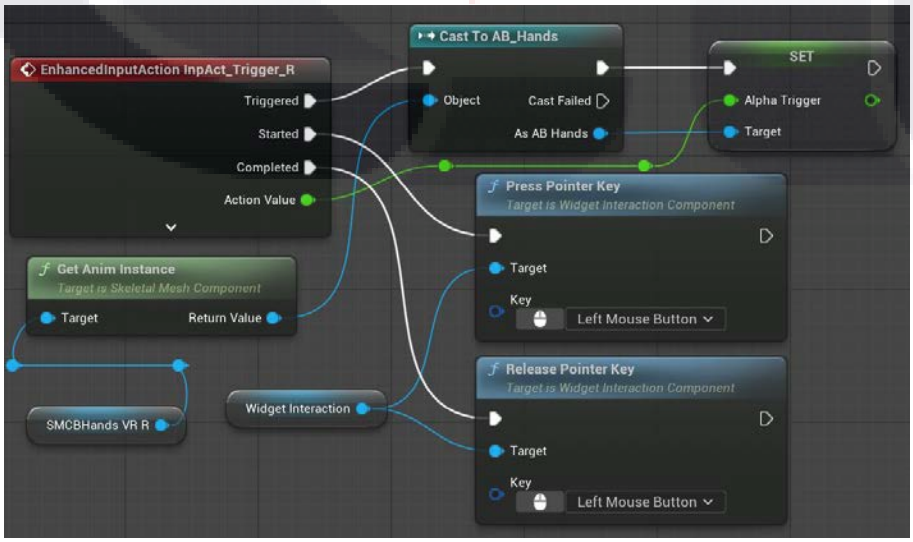


Figura 91 – Métodos para teletransportar al jugador conectados al joystick derecho

Para la segunda acción de movimiento, la rotación, solo se necesita trabajar con el método "Add Actor World Rotation"; el cual, en este proyecto, se ejecuta al accionar el joystick izquierdo. Para ello, del nodo del botón en su conexión de salida se separa el valor que arroja, el cual es un vector con los valores en 'x' y en 'y', para utilizar los valores de 'x' en la decisión de hacia dónde rotar al jugador; los valores en 'y' se utilizan para la animación del pulgar en la mano izquierda. Para no dar cambios de rotación abruptos, se utiliza un nodo de método "Select Float" de manera que si el valor percibido en 'x' es mayor a 0, se añade solo un grado al eje 'z' del jugador, y si es menor, se resta un grado al eje 'z'. Todo esto se muestra en la figura 92.

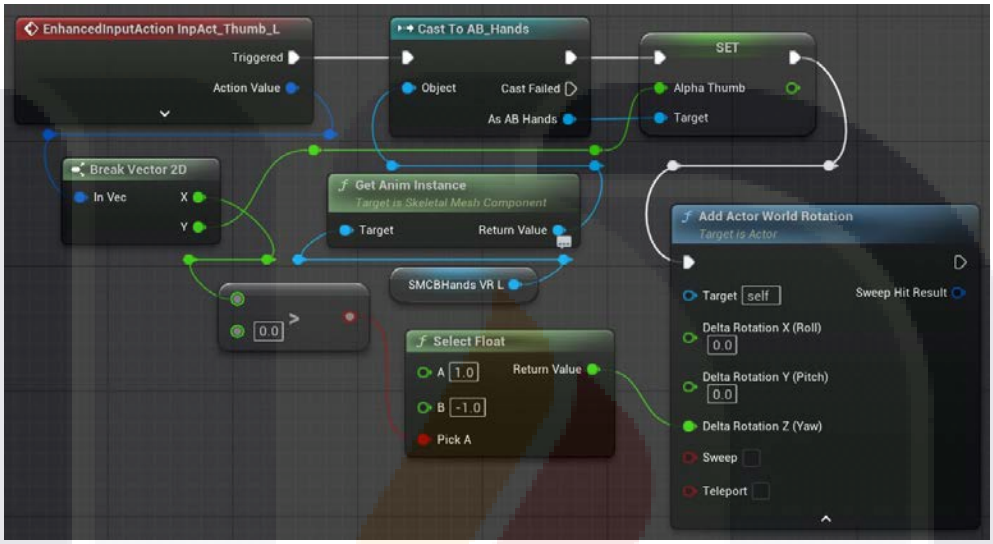


Figura 92 – Añadiendo rotación en joystick izquierdo

Interacción del jugador con los objetos virtuales

- **Agarrar objetos con las manos virtuales**

El primer paso para lograr que las manos virtuales puedan agarrar un objeto es hacer que puedan detectar dicho objeto. Para ello, UE 5 tiene componentes de tipo "Collision" que vienen a servir como un espacio de detección en el cual se puede identificar si algún objeto, o jugador, está colisionando con dicho espacio; UE 5 tiene estos componentes en forma de esfera, cubo y cápsula (cilindro). En este caso, se decide agregar una colisión en forma de esfera. Para ello, en el Blueprint del jugador se agrega un componente del tipo "Sphere Collision", nombrado "Collision_R" para que sea hijo de la mano en la que se planea desarrollar la función de agarrar objetos; en este proyecto, se deja en la mano derecha. En el "Viewport" se hace el ajuste del tamaño y la posición que tendrá con respecto al objeto padre (mano derecha), para que se visualice como se muestra en la figura 93. Después de haber añadido el elemento "Collision_R", se pueden utilizar dos eventos que son propios de ese tipo de componentes para poder programar lo que debe de suceder cuando un objeto entra en este espacio de detección y cuando sale de él. Dichos eventos son "On Component Begin Overlap" y "On component End

Overlap". Ambos pueden ser añadidos al "Event Graph" al seleccionar el nodo del componente "Collision_R" y al buscarlos en la pestaña "Details" (Figura 94).

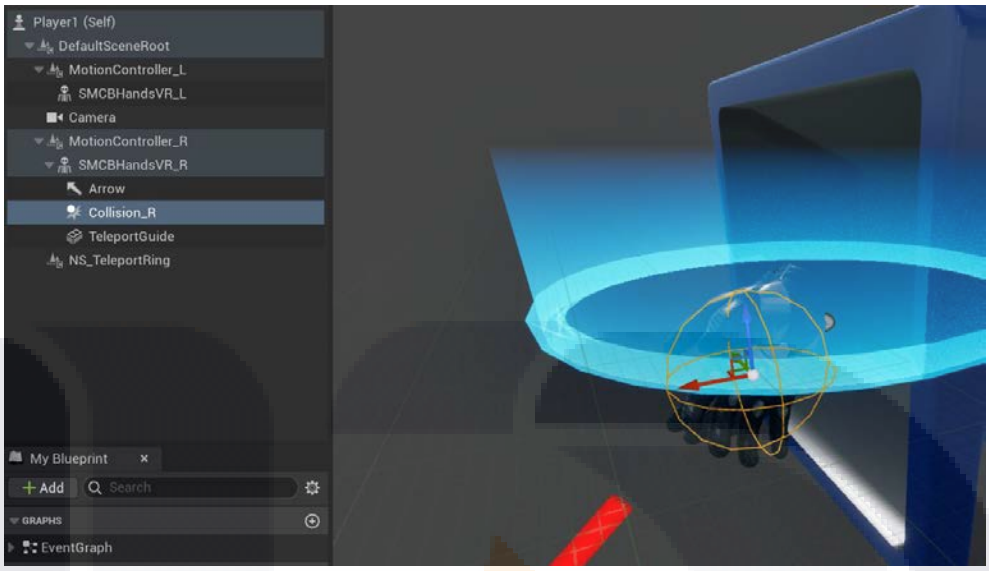


Figura 93 – Acomodando el componente "Sphere Collision"

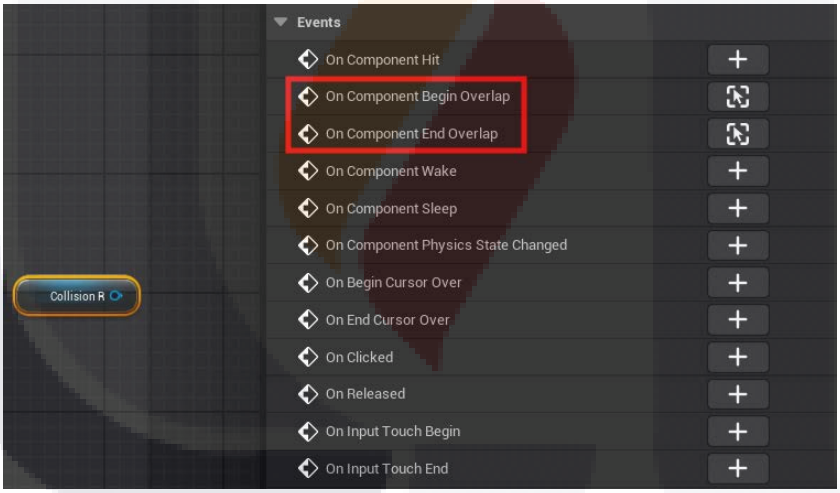


Figura 94 – Añadiendo eventos para detectar objetos

Una vez que se tiene acomodado el elemento en espacio visual y añadidos los eventos que servirán para detectar lo que colisione en él, ya se puede construir la siguiente lógica para hacer que las manos virtuales puedan agarrar objetos: al detectar algo, hacer una validación para saber si es aquello que se pretende agarrar con las manos y, en caso de ser positiva, almacenar el objeto en una instancia para poder accionar el agarre con lo programación de un botón en específico.

Primeramente, para lograr la lógica anterior, se parte de utilizar el evento "On Component Begin Overlap" que se ejecuta constantemente cuando algo colisione con el espacio de detección. De

él, se puede utilizar la conexión de salida "Other Actor" para obtener el nodo del método "Actor Has Tag", el cual retorna verdadero si el objeto que se está detectando tiene una etiqueta específica para identificarlo; en el caso de este proyecto, todo aquel objeto que tenga la etiqueta "Grabbable" se considera para ser agarrado por las manos virtuales. De esta manera, al ser positiva la validación de la etiqueta, se procede a almacenar el objeto detectado en una variable llamada "Actor_Grabbable". Dicha variable no siempre se pretende que tenga un objeto almacenado, por lo que se aprovecha el evento "On Component End Overlap", que se ejecuta cuando ya nada colisiona en el espacio de detección, para borrar la instancia que pudiera haber en "Actor_Grabbable". Todo esto se muestra en la figura 95 acompañado de una validación inicial con el método "Is Valid", ya que es recomendable utilizar este método antes de trabajar con algún objeto para validar que no se esté trabajando con un valor nulo o algún tipo de referencia vacía.

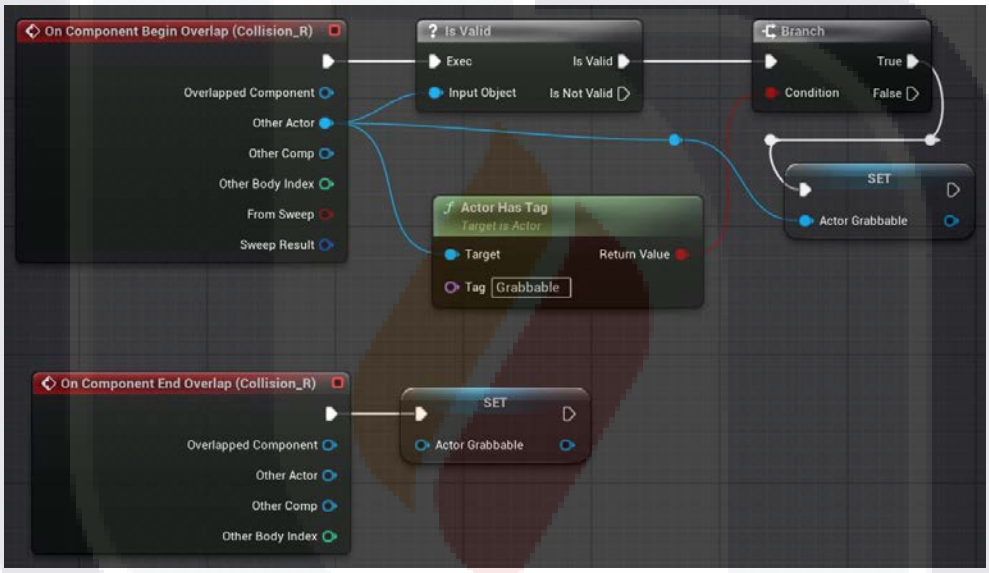


Figura 95 – Utilizando eventos de detección para filtrar los objetos que detectan

Ahora solo falta establecer qué es lo que debe de pasar con el objeto que ha sido almacenado en "Actor_Grabbable". En este proyecto se establece que el gatillo derecho accionado por el dedo corazón será el que podrá accionar la acción de agarrar con la mano virtual derecha. Para ello, después de ejecutar la animación que tiene asignada ese dedo en la conexión "Triggered", se vuelve a implementar la validación "Is Valid" para detectar si hay algún objeto en la variable "Actor_Grabbable", en caso de ser verdadera, se utiliza el nodo del método "Attach Actor To Component" para hacer que el objeto en la variable sea unido a la mano virtual derecha. Esto se logra haciendo que el objeto "Actor_Grabbable" sea conectado a través de la conexión de entrada "Target" y el componente del Blueprint con la mano derecha sea conectado a través de "Parent" en el método "Attach Actor To Component". Y para la acción de soltar, cuando el gatillo deje de ser accionado, lo cual se ejecuta con la conexión de salida "Canceled" y

"Completed", se ejecuta el método "Detach From Actor" que recibe nada más al objeto de "Actor_Grabbable" en sus conexiones de entrada; evidentemente, en esta acción también se ejecuta primero el método "Is valid" para asegurar que haya realmente un objeto en la variable que se quiere utilizar. La figura 96 muestra la programación detallada de este botón.

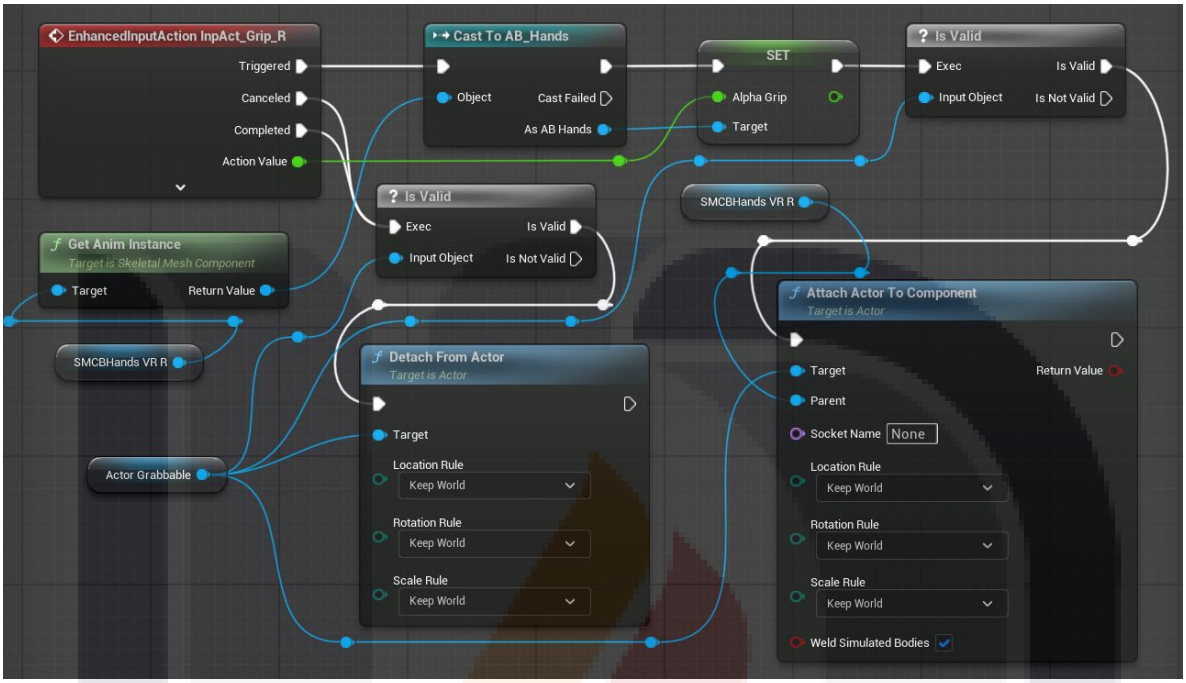


Figura 96 – Programación de botón para agarrar y soltar cosas

4.3: Nubes de puntos optimizadas en el entorno de realidad virtual

Para utilizar las nubes de puntos optimizadas obtenidas con el proceso de optimización anteriormente descrito, se toman dos caminos: utilizar el plugin nativo mencionado en [37] y procesar las nubes de puntos en un software de modelado 3D (Blender) para comparar cuál de esas dos opciones permite lograr tanto la visualización como la manipulación que se busca. Además, considerando que se quiere desplazar, escalar y rotar las nubes de puntos en el entorno virtual y que los controles de las gafas virtuales tienen una cantidad pequeña de botones, se decide utilizar los botones de los controles únicamente para desplazar las nubes y también se decide utilizar las interfaces de usuario gráficas que ofrece UE 5 para crear un sistema con botones virtuales que se puedan usar para hacer las tres transformaciones que se buscan sobre las nubes.

Por todo lo anterior, a continuación, se explica primeramente cómo es que se llevó a cabo la transformación y la exportación de las nubes de puntos en Blender para posteriormente pasar a añadirlas y programarlas sobre el sistema de realidad virtual creado y, de esta manera, terminar con la programación de los botones virtuales que ejecutarán las transformaciones.

Transformación de las nubes de puntos

Como se menciona en el capítulo 2, una de las bondades de Blender es que permite ejecutar código de Python para crear y modificar modelados 3D, por lo que se aprovecha esa característica para ejecutar un programa simple que lea el archivo de las nubes de puntos, extraiga las coordenadas de cada punto y cree una esfera 3D en cada una de esas coordenadas.

Para lograr esto se tuvo que hacer una conversión en el formato que se manejan las nubes optimizadas. El formato con el que resultan al final del procedimiento de optimización es ".ply"; sin embargo, este formato no puede ser leído por el método nativo "open" de Python, además de que se decide no modificar el núcleo de Python que trae por defecto Blender al instalar más librerías sobre él. De esta manera, es que se procede a utilizar el software de visualización de mallas 3D MeshLab para abrir en él las nubes en formato ".ply" y exportarlas al formato ".xyz". En la figura 97 se muestra cómo se ve un archivo de nubes de puntos abierto en Meshlab, en la figura 98 se muestra que para cambiar el formato que se está utilizando solo basta con usar la opción que está en "File/Export Mesh As" para seleccionar el formato deseado como se muestra en la figura 99.

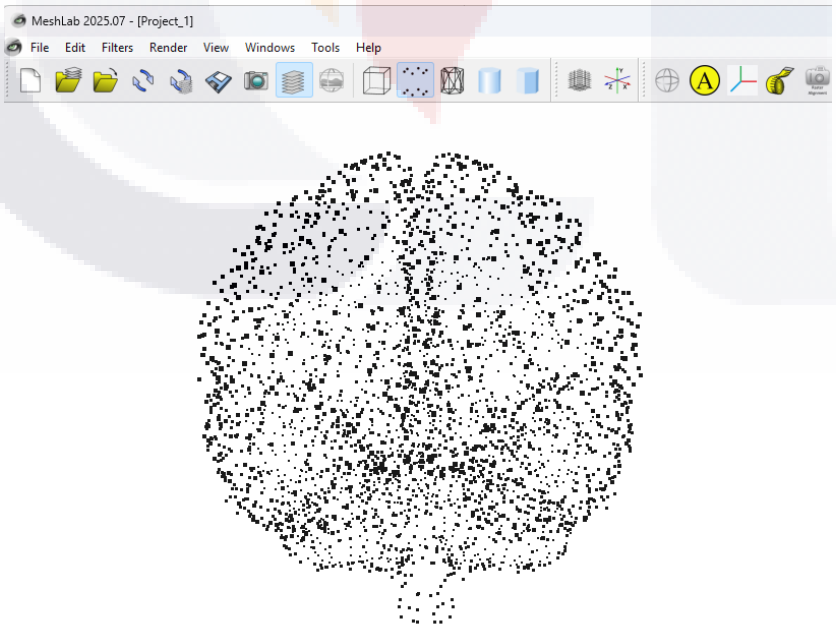


Figura 97 – Nubes de puntos en el software Meshlab

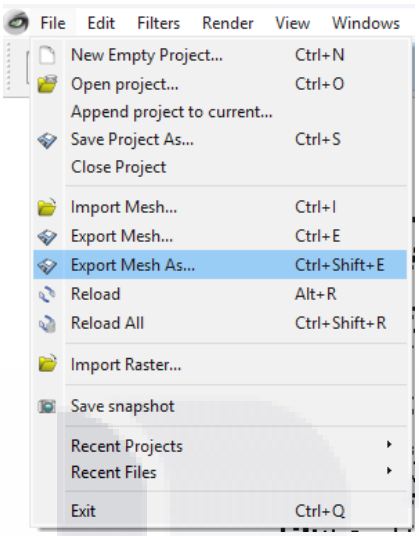


Figura 98 – Opción de exportado

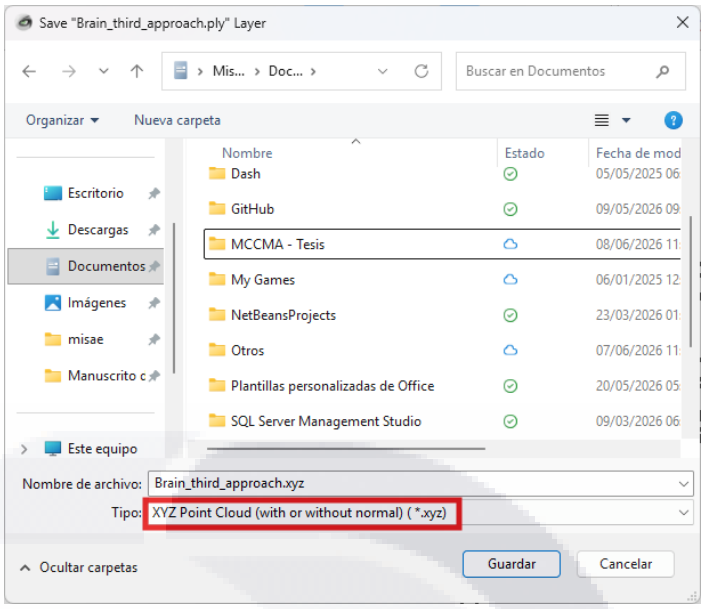


Figura 99 – Seleccionando el formato deseado

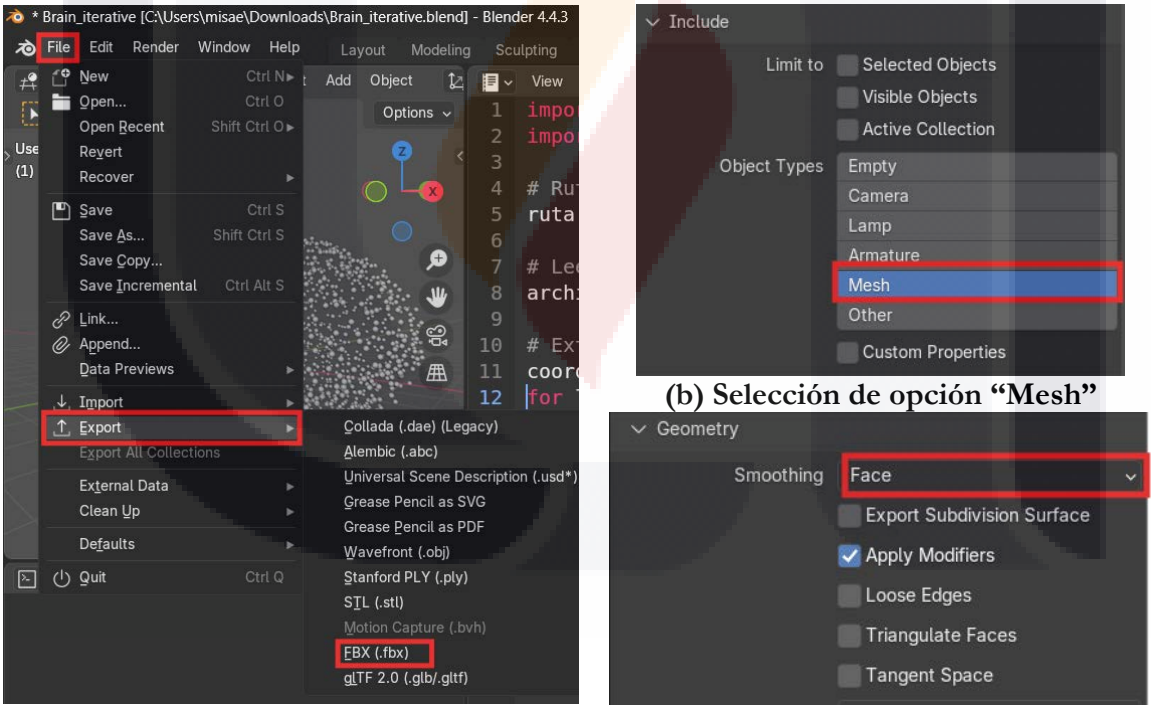
Ya con las nubes de puntos exportadas para poder ser leídas con el método "open" de Python, se utiliza el siguiente código sobre Blender:

```
import bpy
import bmesh

# Ruta al archivo XYZ
ruta = "C:/ruta/Brain_iterativo.xyz"

# Leer el archivo .xyz
archivo = open(ruta, 'r').readlines()

# Extraer coordenadas
coordenadas = []
for line in archivo:
    c = line.split(" ")[0:3]
    coordenadas.append(
                [float(c[0])*10,float(c[1])*10,float(c[2])*10])

# Crear esferas
for coord in coordenadas:
    mesh = bpy.data.meshes.new("SphereMesh")
    objeto = bpy.data.objects.new("Sphere", mesh)
    bpy.context.collection.objects.link(objeto)
    bm = bmesh.new()
    bmesh.ops.create_uvsphere(
              bm, u_segments=16, v_segments=8, radius=0.065)
```

```
25      bm.to_mesh(mesh)
26      bm.free()
27      obj.location = coord
```

Nótese que el método utilizado para crear las esferas es el método "create_uvsphere" de la librería "bmesh" que es nativa de Blender. Además, nótese de igual manera que los parámetros utilizados "u_segments" y "v_segments" delimitan respectivamente el número de segmentos horizontales y verticales que conformarán a la esfera creada; cabe mencionar que esa cantidad de segmentos, ocho horizontales y doce verticales, se considera como una cantidad de baja resolución, ya que intentar crear esferas de alta resolución para representar la nube de puntos hace que se sature el equipo de cómputo utilizado en este trabajo.

Después de haber creado satisfactoriamente las nubes de puntos, se procede a exportar el modelo de Blender a un formato ".fbx" para ser añadido al proyecto de realidad virtual. Los pasos para lograr se muestran en la figura 100, los cuales se componen de ir a la opción "File/ Export/ FBX (.fbx)" (a), y luego en las opciones para exportación seleccionar solo la opción "Mesh" en "Object Types" (b) y la opción "Face" de "Smoothing" (c).



(a) Exportar a formato ".fbx" **(b) Selección de opción "Mesh"** **(c) Selección de opción "Face"**

Figura 100 – Proceso de exportación en Blender

Adición de las nubes de puntos al entorno virtual

Como se mencionó con anterioridad, para añadir las nubes de puntos se utilizan dos maneras para la comparación en la efectividad de cada una: utilizando un plugin nativo de UE 5 y empleando modelado 3D en ellas con Blender para exportarlas en formato ".fbx".

Para la primera opción, primeramente, se tiene que ir a "Edit / Plugins" para buscar el plugin llamado "LiDAR Point Cloud Support" para añadirlo al proyecto (figura 101). Para que UE 5 lo añada correctamente, se tiene que reiniciar el motor.

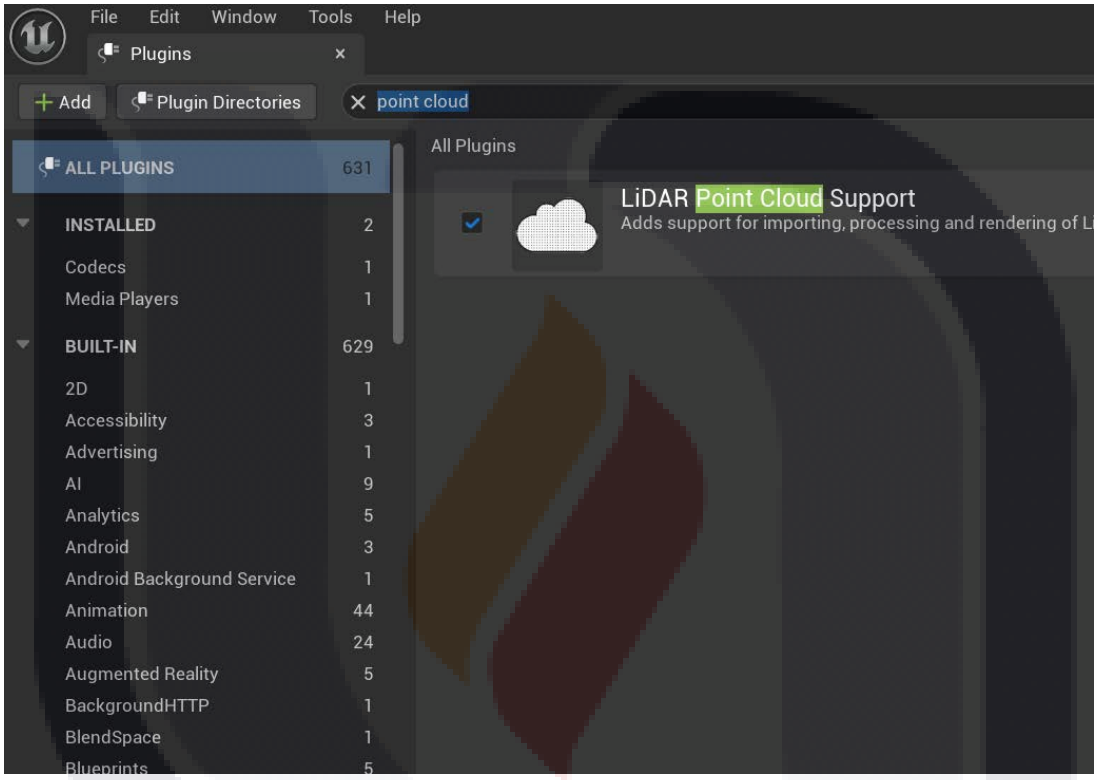


Figura 101 – Añadiendo plugin de nubes de puntos nativo de UE 5

Posteriormente, se tiene que importar el modelo en el que se quiera trabajar en el entorno de realidad virtual. Cabe señalar que, al haber añadido el plugin, en la ventana para buscar e importar los modelos, se habilitan las extensiones de archivos de nubes de puntos para poderlas añadir al proyecto. Las extensiones, o formatos, que soporta el plugin son: ".e57", ".las", ".laz", ".pts", ".txt" y ".xyz" (figura 102).

Una vez importado y añadido el modelo que se vaya a utilizar, se procede a crear una clase de Blueprint del tipo "Actor" (véase la figura 53 como referencia) para añadir ahí la nube de puntos como un componente de la clase. Esto se hace por dos razones, la primera es que cada nube de puntos tiene que pertenecer a una clase en la que se pueda agregar la programación necesaria para modificar sus atributos; esto bien podría quedar en la programación general del nivel, pero

por buenas prácticas en el desarrollo de UE 5, la programación que modifique un objeto específico tiene que estar en una clase de Blueprint específica. Y la segunda razón de que cada nube esté en una clase separada es que de esta manera se podrá hacer una comparación de la eficacia de las maneras en las que se añadieron las nubes al proyecto; ya que cada clase podría necesitar de diferentes elementos para funcionar como se espera.

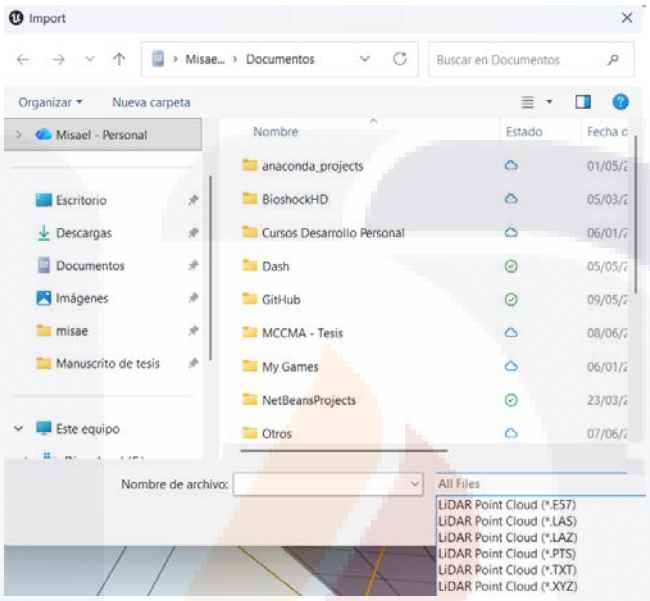


Figura 102 – Formatos de nubes de puntos admitidos por el plugin

Ahora, para añadir la nube que fue procesada en Blender, se procede a importarla en el "Content Browser". Como UE 5 acepta el formato ".fbx" sin plugins especiales, simplemente se busca el archivo y se añade al proyecto. Al hacerlo, aparece una ventana emergente (figura 103) con la cual se configuran los parámetros para importar la nube como un modelo 3D. Aquí hay varias opciones importantes a considerar:

- "Import Static Meshes": Tiene que ser verdadera porque toda la nube de puntos es un conjunto de mallas 3D, si queda en falso, no se importa nada.
- "Combine Static Meshes": Puede quedar en falso, pero eso haría que cada punto se importe como un elemento separado, lo cual borraría la forma del objeto original.
- "Build Nanite": Hay que considerar que "Nanite" es el sistema de geometría de UE 5, si esta opción queda en verdadero, toda la nube de puntos es renderizada por el sistema "Nanite" al momento de ser importada, lo cual trae el único inconveniente derivado de que la nube fue formada con pocos segmentos geométricos: utilizar este sistema de geometría no los representaría los puntos como esferas, sino como polígonos. Si la opción queda en falso, se consigue la representación con esferas, pero la nube de puntos

se importa sin colisiones, ya que "Nanite" calcula una representación geométrica del modelo 3D que se importa.

- "Import Skeletal Meshes": Un "Skeletal Mesh" es un modelo que puede tener articulación para poder mover sus partes, como si tuviera un "esqueleto" que estructurara cada una de sus partes para poder ser movidas. Considerando esto y, considerando que las nubes de puntos modeladas en Blender 3D no tienen algo parecido, no es necesario marcar esta opción en verdadero. Aunque, si se llega a hacer, UE 5 creará una para las nubes de puntos que se importen. Sin embargo, al tratarse de un conjunto numeroso de esferas, el "Skeletal Mesh" que construye UE 5 no resulta funcional.
- "Create Physics Asset": Un "Physics Asset" es un recurso que utiliza UE 5 para interpretar cómo funcionaría físicamente cada parte de un "Skeletal Mesh" al ejecutar un proyecto. Como no se utiliza ningún "Skeletal Mesh" con las nubes, no es necesario ni útil marcar esta opción o dejarla marcada como verdadera.

A manera de comparación, en este proyecto se importa el mismo modelo de nubes de puntos dos veces, una sin utilizar "Nanite" y otra utilizando ese sistema de geometría de manera que se puedan apreciar las diferencias de ambas formas de importación tanto visualmente como en el aspecto de la programación.

Programación de las nubes de puntos

A continuación, lo que sigue es diseñar y programar la clase de Blueprint en la que estará cada nube de puntos a utilizar: importada con plugin en formato ".xyz", importada en formato ".fbx" utilizando "Nanite" e importada en formato ".fbx" sin utilizar "Nanite".

Se empieza explicando la lógica que se sigue tanto en la importada en formato ".xyz" como en la que no utilizó "Nanite", ya que ambas presentan la misma limitante: no tienen colisiones, por lo que no pueden ser detectadas por el componente "Sphere_Collision" utilizado para la función de agarrar objetos con las manos virtuales. Por esta razón, es que se añade un componente de tipo "Sphere" que se establece como el objeto padre de las nubes de puntos para que la esfera nativa de UE 5 pueda ser detectada y agarrada con una mano virtual. Evidentemente, la esfera se moldea para poder quedar sobrepuesta en la nube utilizada (en este caso, la del objeto "Brain"). En la figura 103 se muestra tanto el acomodo del elemento "Sphere" y la nube en formato ".xyz" como los dos atributos que permitirán trabajar con esa esfera de manera que no bloquee visualmente a la nube de puntos: "Visible", el cual establece si se puede ver o no en el modo desarrollo a través del "Viewport" y en el editor del nivel, y "Hiden in Game", que establece si es visible o no al ejecutar el proyecto.

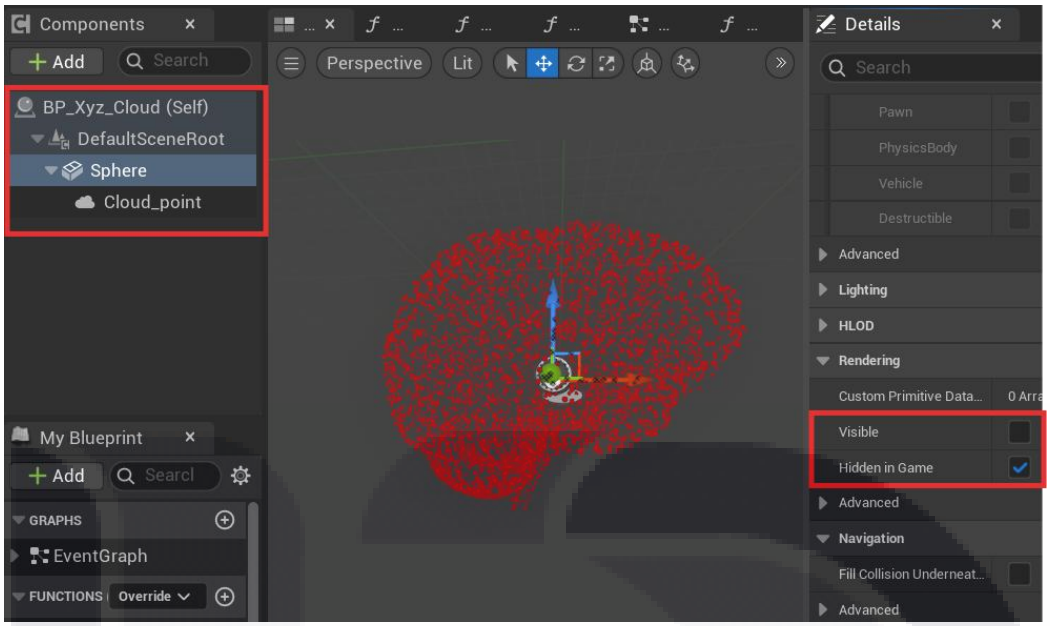


Figura 103 – Acomodo de nube de puntos ".xyz" con esfera invisible

Para la parte de la programación, se establece lo siguiente dentro de la clase de Blueprint:

- **Almacenamiento de variables para escalamiento**: sobre el "Event Graph" se toma el nodo evento "BeginPlay" para almacenar la escala original que se está manejando para la esfera, que a su vez es la escala que tendrá su componente hijo: la nube. Se crearán dos variables para esto: "Init_Scale" y "Aux_Scale" (figura 104).

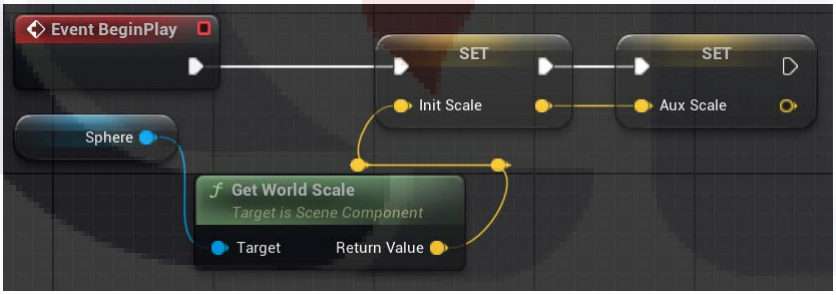


Figura 104 – Variables para escalamiento

- **Función de escalamiento**: recibirá como entrada el valor a modificar la escala ("Value_Scale") y un booleano que determinará si se restablecerá la escala original del objeto ("Is_Reset"). La lógica que se sigue es la siguiente: si se quiere reestablecer la escala original, se asigna el valor de "Init_Scale" al componente "Sphere", que consecuentemente reestablecerá también la escala de la nube; en caso contrario, se modifica el valor de

"Aux_Scale" añadiéndole "Value_Scale" y asignándolo como la escala del componente "Sphere". En ambos casos se utiliza el método "Set World Scale 3D" (figura 105).

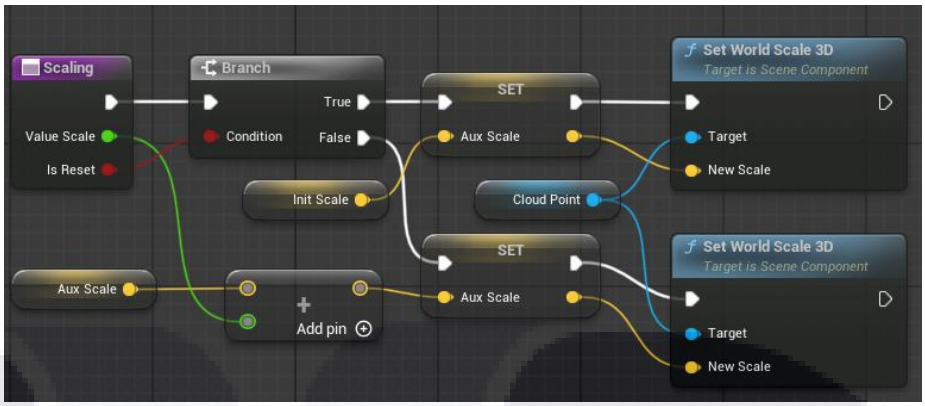


Figura 105 – Función de escalamiento

- **Función de rotación**: recibirá como entrada el valor a modificar en la rotación ("Rotation_Value") y un booleano que determinará si la rotación es horizontal o vertical ("Is_Vertical"). La lógica que sigue es la siguiente: si la rotación es horizontal, con el método "Make Rotator" se añade el valor que se tenga en "Rotation_Value" al eje 'x' y se añade la nueva rotación al componente "Sphere" con el método "Add Local Rotation". En el caso de que la rotación sea vertical, se usa el mismo procedimiento pero añadiendo el valor que se tenga en "Rotation_Value" al eje 'y' (figura 106).

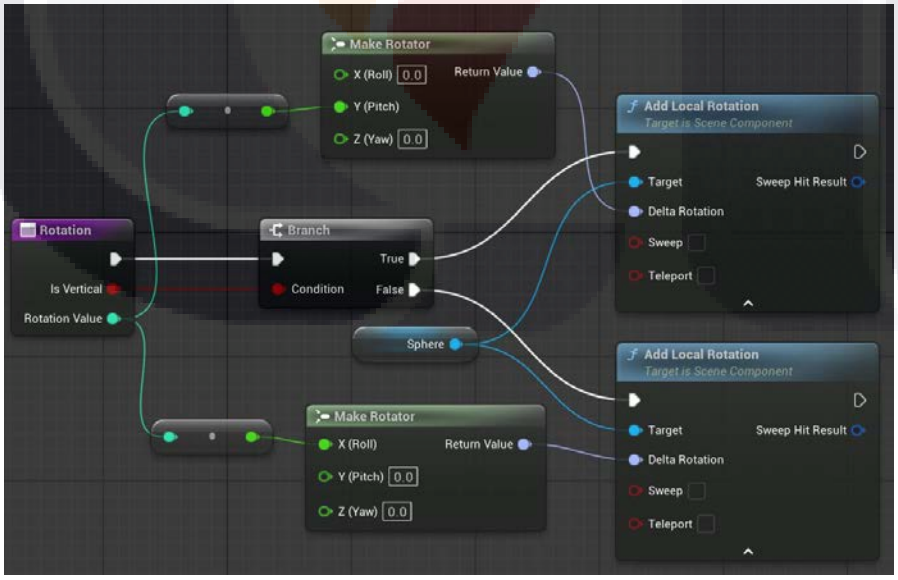


Figura 106 – Función de rotación

- **Función de desplazamiento**: recibirá como entrada el valor a modificar en las coordenadas ("Value_Moving") y un booleano que determinará si el desplazamiento es horizontal o vertical ("Is_Vertical"). La lógica que sigue es la siguiente: si el desplazamiento es horizontal, simplemente se le añade el valor que se tenga en "Value_Moving" al componente "Sphere" en el eje "z" con el método "Add World Offset". En caso contrario de querer un desplazamiento vertical, se añade el valor que se tenga en "Value_Moving" al eje 'z' del componente "Sphere".

Figura 107 – Función de desplazamiento

Como se mencionó con anterioridad, esto se hace tanto con la nube importada en formato ".xyz" como con la que no utilizó "Nanite". Sin embargo, para la nube que sí utilizó "Nanite" en su importación, solamente difiere en que no necesita un componente del tipo "Sphere", por lo que se establece en programación exactamente lo mismo que lo establecido con las nubes anteriores, con la única diferencia de que en vez de utilizar el componente "Sphere", se tiene que utilizar el componente de la nube como tal.

Programación de botones en la interfaz visual (widgets)

En este punto se tiene la programación en cada una de las clases de Blueprints que tienen las nubes de puntos. Sin embargo, un detalle que se presenta por la naturaleza de los controles de las gafas de realidad virtual es que hace falta una manera de accionar cada una de las funciones que se han establecido en las clases de las nubes; pues no hay botones suficientes en los controles para accionar cada una. Para ello, es que se decide aprovechar un componente de UE 5 para solventar este detalle: los widgets. Un widget viene siendo un panel virtual, con botones virtuales,

que pueden ser accionados con los controles de las gafas de realidad virtual, lo cual, hace que se aumente el número de funcionalidades que se pueden ejecutar con ellos.

La manera de crear uno, al igual que una clase de Blueprint, es directamente sobre el "Content Browser", como se muestra en la figura 108, aunque los widgets se crean usando la opción "User Interface/ Widget Blueprint".

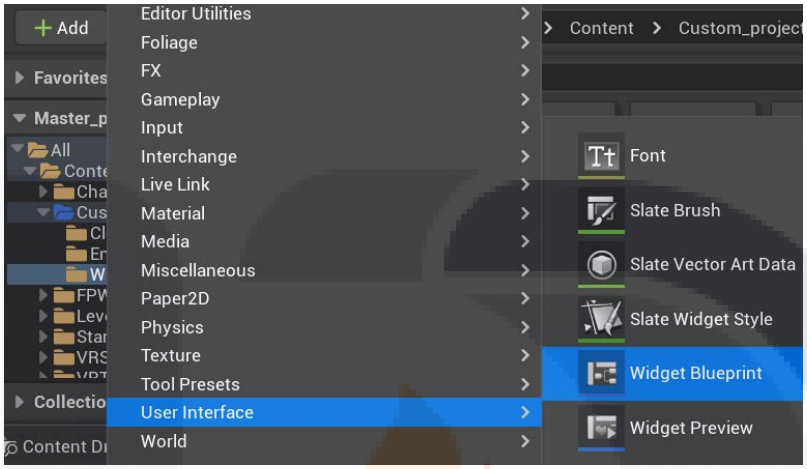


Figura 108 – Creación de widget

Una vez creado, al abrirlo se encontrarán dos principales secciones: "Designer" para dar forma visualmente al widget y "Graph" para programar la acción de cada botón añadido. Estas secciones se muestran sobre la pestaña "Details" como se ve en la figura 109.

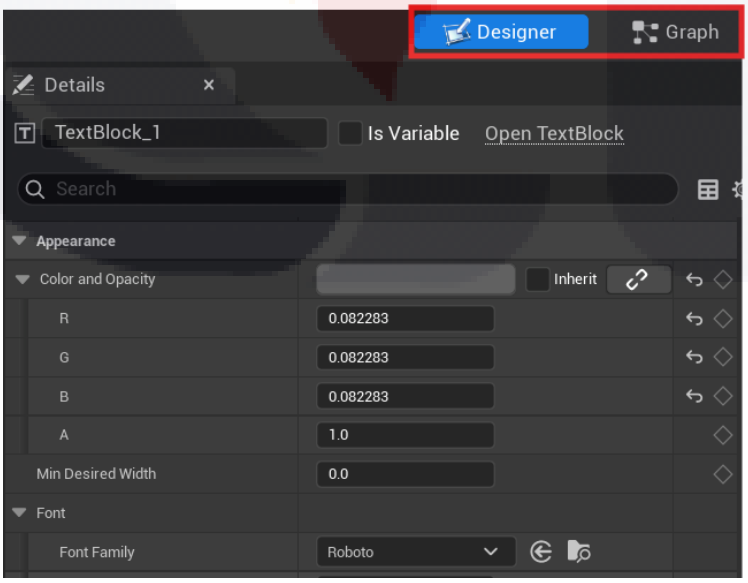


Figura 109 – Secciones para diseñar visualmente y para programar cada botón

En la sección "Designer", se pueden añadir los elementos visuales que sean necesarios para darle forma al widget. Nótese que en la figura 110 se puede apreciar que los componentes que se utilicen en el widget tienen que ser todos hijos del "Canvas Panel"; en este caso, se decide utilizar un "Grid Panel" para tener más versatilidad en el acomodo de los botones, los cuales serán hijos de este mismo y, a su vez, serán padres del cuadro de texto que tendrá aquello que se quiera imprimir sobre ello. Nótese de igual manera que el botón central ("Reload") tiene el símbolo de una pica, ese botón será utilizado para restablecer la posición original de las nubes, mientras que los otros botones las desplazarán horizontal o verticalmente. Cabe mencionar que, al seleccionar un botón o cualquier elemento del widget, se podrá ajustar tanto su color como su posición en la pestaña "Details".

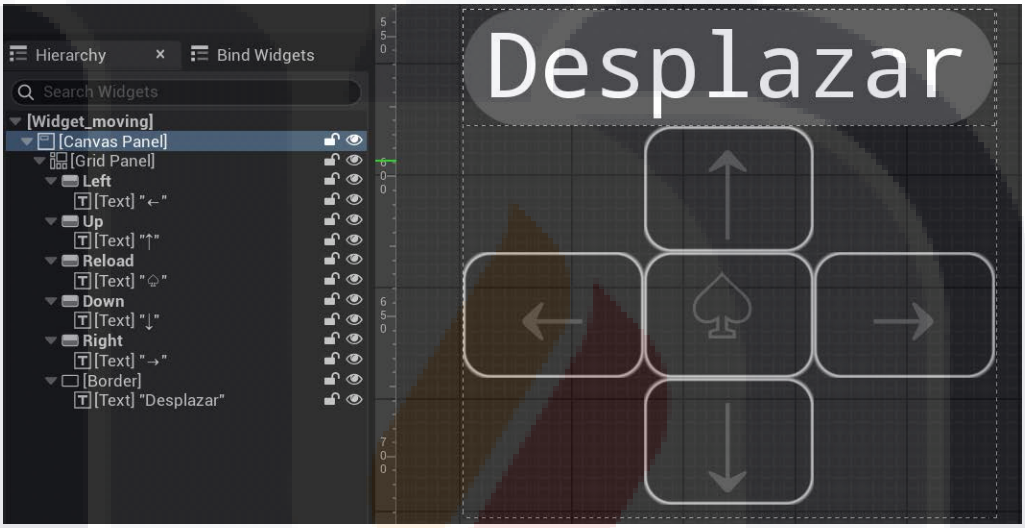


Figura 110 – Diseño visual de un widget

Para la parte de la programación en la sección "Graph", primeramente, se utiliza el nodo de evento "Event Construct" para tomar la instancia de la clase Blueprint en la que está cada nube con el método "Get All Actors Of Class" y para tomar su ubicación original con "Get Actor Location", que será necesaria al accionar el botón "Reload". En la figura 111 se muestra la manera en la que se hace esto con la nube ".xyz".

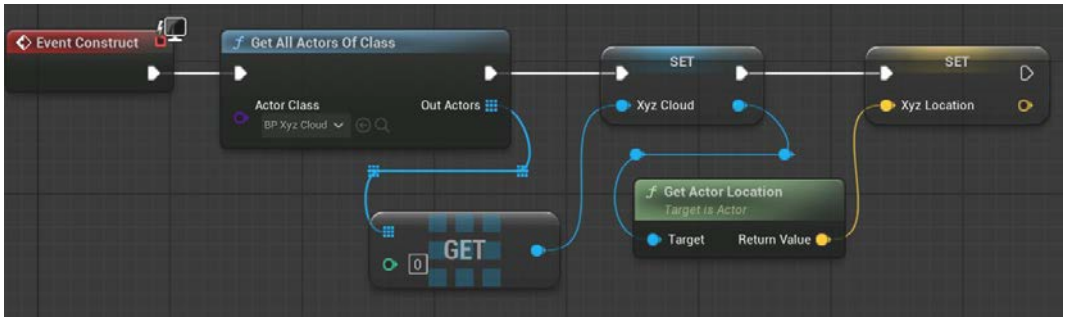


Figura 111 – Obteniendo instancias de las clases de las nubes

Para programar la acción de cada botón, se debe añadir el nodo evento "On Clicked" de cada uno que ejecutará las acciones que cada botón desencadenará al ser accionado. Esto se logra seleccionando un botón en el apartado "Variables" para proceder a buscar el nodo en "Details" estando en la sección "Graph" (figura 112).

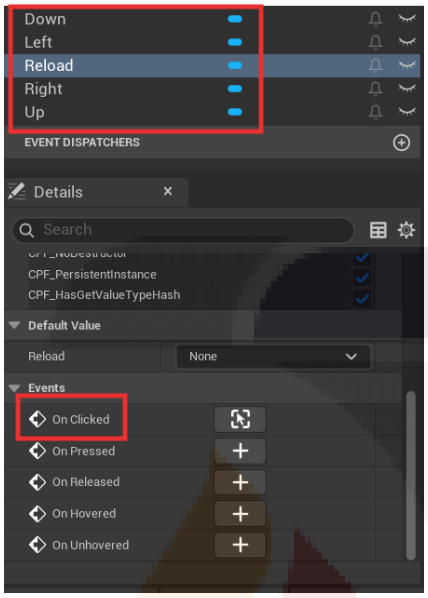


Figura 112 – Añadiendo evento "On Clicked" de cada botón del widget

Con los nodos añadidos, se procede a crear la siguiente lógica: primero se almacena en una variable el valor de desplazamiento que se quiere añadir a las coordenadas del objeto, dicho valor estará determinado por el botón que se accione, en este caso, si se acciona el botón "Down" o "Left" se almacena un valor de -5 y si se acciona el botón "Up" o "Right" se almacena un valor de 5. Evidentemente, solo cuando se trabaje con los botones "Up" y "Down", se guardará un valor "True" en la variable "Is_Vertical" para establecer que el desplazamiento no es horizontal y también para enviarla después junto con el valor de desplazamiento al método de desplazamiento que se creó en la clase de Blueprint de cada nube. En la figura 113 se muestra un ejemplo de cómo es que queda configurado esto con los botones "Up" y "Down" para terminar ejecutando el método que hay en la clase de la nube ".xyz", esto se debe de repetir por cada clase de las otras nubes para que todas sean desplazadas al parejo. Cabe mencionar que esta misma lógica se repite con los botones "Left" y "Right", con la diferencia de que ellos manejan un "False" en "Is_Vertical".

En la figura 114 se muestra la programación que se sigue con el botón "Reload", la cual consiste en únicamente establecer la ubicación original de la nube con el método "Set World Location" utilizando tanto la instancia de cada nube como su ubicación que se obtuvieron con el nodo "Event Construct". Nótese que esto se aplica sobre el componente padre de cada clase, en el caso de la nube ".xyz", se aplica sobre el componente "Sphere". En el caso de la nube que sí

utilizó "Nanite" en su importación, se aplica directamente sobre ella. Nuevamente en la figura solo se muestra esto con solo una nube, pero se tiene que ejecutar sobre todas para que la acción sea ejecutada al parejo sobre las tres.

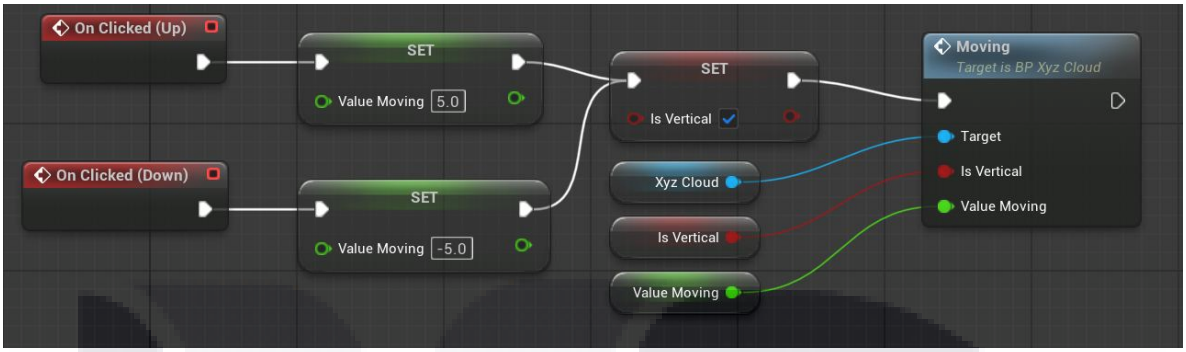


Figura 113 – Programación en los botones del widget

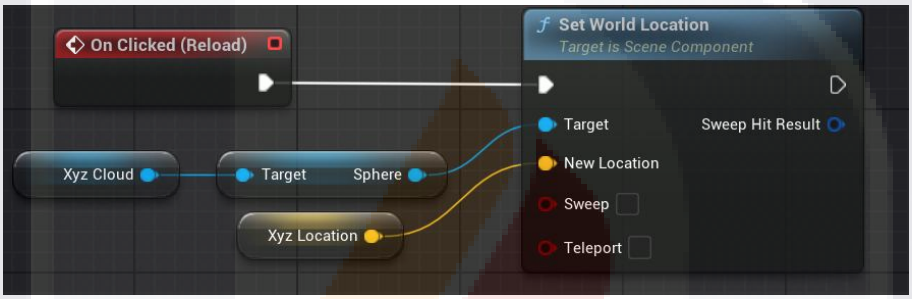


Figura 114 – Programación en el botón "Reload" del widget

Toda la lógica anterior puede repetirse en el widget que ejecutará la rotación: primero obtener la instancia de las nubes de puntos y el valor de su rotación inicial, luego utilizar una variable para identificar si es una rotación vertical u horizontal, para terminar invocando el método que modifica la rotación en las clases de las nubes. Si acaso, el único detalle a considerar es que, en el botón "Reload", se ejecutará el método "Set World Rotation" (figura 115), pero se mantiene la lógica de aplicar el método sobre el componente padre en cada clase.

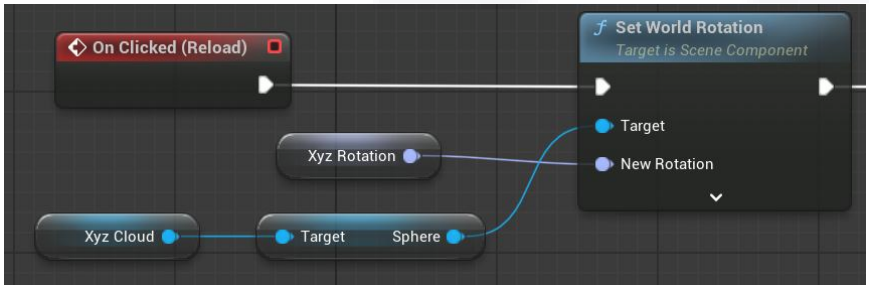


Figura 115 – Botón "Reload" del widget de rotación

El widget del escalamiento sigue una lógica un tanto diferente. Para empezar, este widget se conforma de menos botones (figura 116) ya que al escalar solo se considera la opción de aumentar o disminuir generalmente el tamaño de las nubes. Además, como ya se utilizan variables en las clases de las nubes que almacenan tanto el valor de la escala que tiene cada nube como un booleano para determinar si se restablece el valor original, ya no es necesario almacenar ni la escala original ni el booleano de decisión en la clase de este widget. Por ello, el nodo "Event Construct" solo se usa para obtener la instancia de la clase de cada nube y los botones ejecutan directamente el método de escalamiento de las nubes ajustando los valores en el mismo nodo del método como se muestra en la figura 117.

Figura 116 – Widget para escalar las nubes de puntos

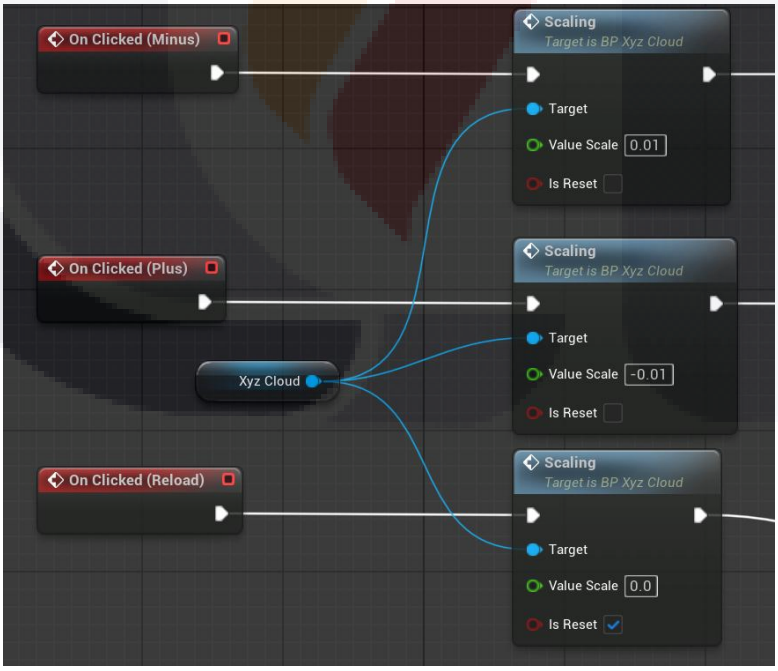


Figura 117 – Programación de los botones en el widget de escalamiento

Con los widgets diseñados y programados, solo faltan tres pasos por hacer:

- Crear una nueva clase de Blueprint en la que se añadirá cada widget como un componente y se acomodarán los tres en el "Viewport" (figura 118).

Figura 118 – Acomodando widgets en una clase de Blueprint

- Acomodar el Blueprint con los widgets creados en el editor de nivel junto con los Blueprints de las nubes (figura 119).

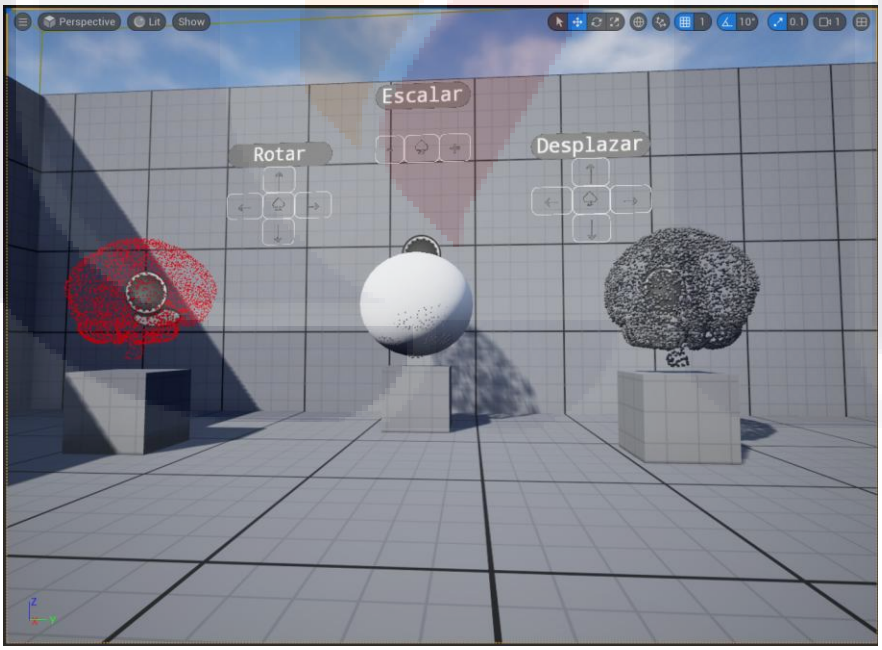


Figura 119 – Añadiendo widgets y nubes de puntos al editor del nivel

- Añadir el componente "Widget Interaction" al Blueprint del jugador como hijo de la mano que podrá accionar los widgets (figura 120). Este complemento trae un apuntador por defecto que se puede mostrar en todo tiempo durante la ejecución al marcar "Show Debug" de "Details" en verdadero y, de esta manera, se podrá ir ajustando la inclinación del apuntador para que coincida con la inclinación de la mano.
- Conectar métodos "Press Pointer Key" y "Release Pointer Key" al botón que podrá accionar los widgets, en este caso se asignan al gatillo derecho accionado con el índice a través de la conexión "Started" y "Completed" del nodo del botón (figura 121).

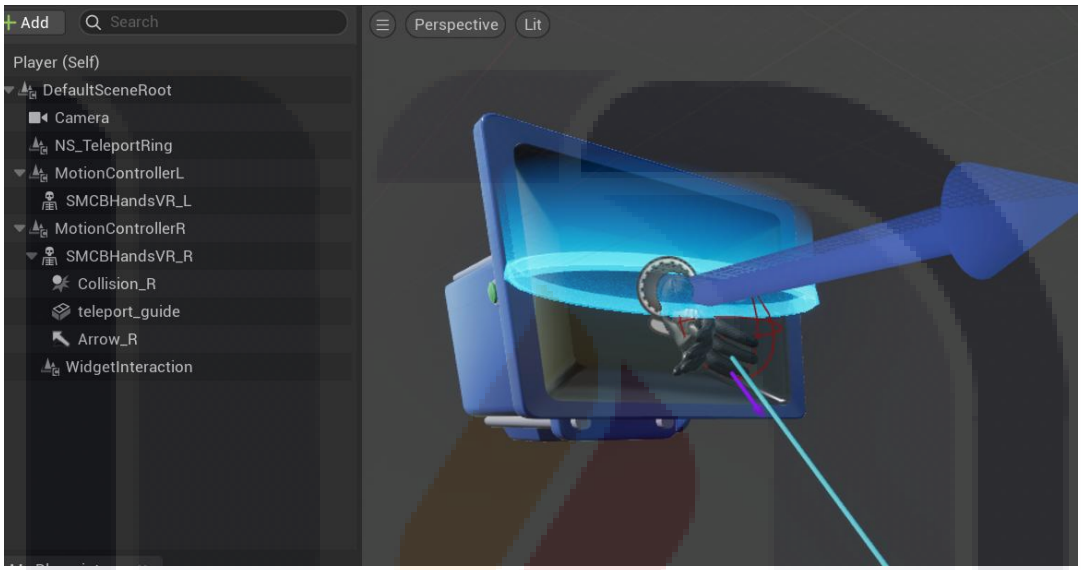


Figura 120 – Añadiendo y acomodando "Widget Interaction" al Blueprint del jugador

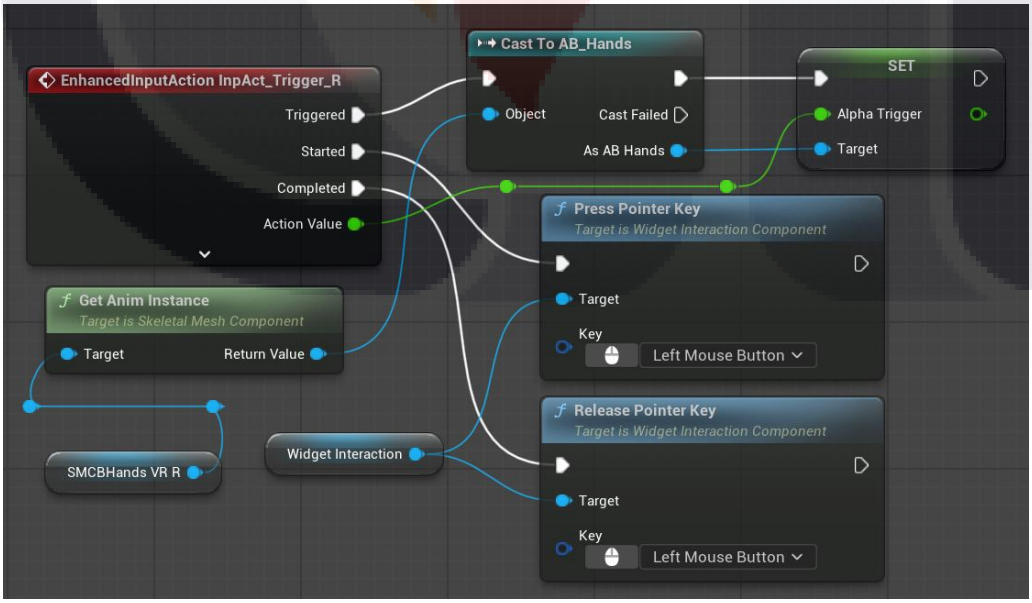


Figura 121 – Programación del botón que accionará los widgets

Ejecución del proyecto

Después de haber seguido los pasos expuestos en este capítulo, se pudo obtener un entorno virtual programado de manera que las manos virtuales tienen animaciones propias (figura 122), el usuario puede moverse mediante un sistema de teletransportación (figura 123) y las nubes de puntos pudieron ser manipuladas tanto directamente con las manos virtuales (figura 124) como con widgets que tienen botones virtuales (figura 125).

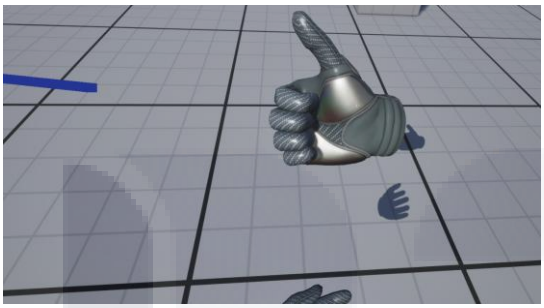


Figura 122 – Manos virtuales con animaciones

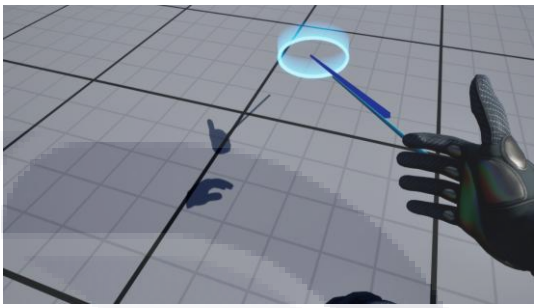


Figura 123 – Sistema de teletransportación

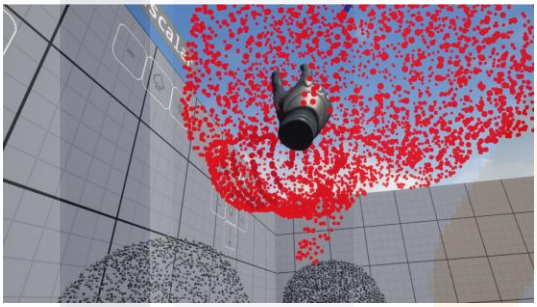


Figura 124 – Manipulando nubes con manos virtuales

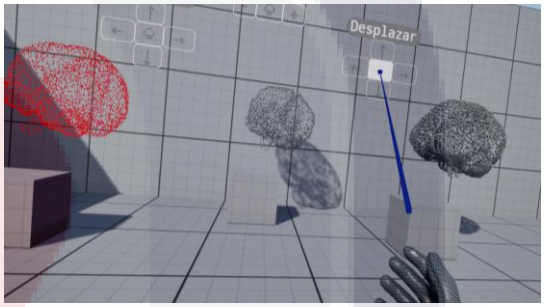


Figura 125 – Manipulando nubes con botones virtuales en widgets

4.4 Análisis del resultado en el entorno virtual

Durante el desarrollo del entorno de realidad virtual propuesto en este capítulo, se puede apreciar que Unreal Engine 5 tiene lo necesario para poder emplear proyectos en los que se visualicen y manipulen nubes de puntos. Sin embargo, cada una de las vías que se exploraron en este trabajo para lograrlo, han demostrado tener áreas de mejora en cada una de ellas.

Para apreciarlas con más claridad, primero se parte de un análisis comparativo de las tres opciones que se usaron para utilizar las nubes de puntos en el entorno de realidad virtual: usando el formato Xyz, el formato Fbx con "Nanite" y el formato Fbx sin "Nanite". Esta comparativa se muestra en forma de tabla considerando los siguientes aspectos:

- Capacidad de detección (Colisiones): Se cumple si la opción utilizada tiene colisiones para poder emplear la función de agarrarse con manos virtuales.

- Capacidad de transformación (Transformación con widgets): Se cumple si la opción utilizada permite hacer transformaciones con botones virtuales sin depender de un elemento padre que le agregue colisiones.
- Capacidad de integración (Ambas funcionalidades): Se cumple si la opción utilizada permite hacer los dos puntos anteriores sin depender de un elemento padre que le agregue colisiones.
- Apariencia de puntos: Se cumple si la opción utilizada permite representar las nubes como puntos o, en su defecto, como esferas.

Cabe mencionar que en este trabajo solo se explicó cómo implementar el sistema de realidad virtual de manera que se pudiera tanto agarrar las nubes de puntos con manos virtuales como modificarlas con los widgets de UE 5, pero en el caso de la capacidad de transformación con widgets, realmente sí es posible implementarlos de manera que permitan modificar directamente las nubes de puntos sin la necesidad de un elemento padre en las clases de Blueprints de las nubes. Simplemente no era conveniente modificar directamente las nubes en este proyecto, ya que no sería funcional alterar las nubes en su tamaño o posición, porque el elemento padre quedaría o desplazado de ellas o no ajustado sus tamaños y la funcionalidad de agarrarlas con manos virtuales sería deficiente. Con lo anterior explicado, los resultados comparativos se muestran en la tabla 6.

Tabla 6 – Análisis comparativo de los diferentes formatos utilizados de las nubes de puntos en el entorno virtual

Formato utilizado	Colisiones	Tansformación con widgets	Ambas funcionalidades	Apariencia de puntos
Xyz	No	Sí	No	Sí
Fbx (con "Nanite")	Sí	Sí	Sí	No
Fbx (sin "Nanite")	No	Sí	No	Sí

Las áreas de mejora se presentan al analizar las causantes de que las colisiones y la apariencia de los puntos no cumplan satisfactoriamente con algunos formatos al intentar implementarse en UE 5. En el caso de las colisiones, las causas son que tanto el formato Xyz como el formato Fbx (sin "Nanite") no tienen colisiones porque no fueron añadidas en su formación y porque no se empleó el sistema de geometría "Nanite" sobre ellos (que genera colisiones sobre todo aquello que renderiza). En el caso de la apariencia de puntos, la causa es que el formato Fbx utilizado fue creado con una baja resolución, ya que Blender no soportaba crear esferas con más caras geométricas por la cantidad de puntos utilizados.

Ante estos panoramas, en la tabla 7 se muestran las áreas de mejora (soluciones) que consideran las causas de los problemas que presenta cada formato de las nubes.

Tabla 7 – Áreas de mejora (soluciones) para cada problema que presentan los formatos utilizados en las nubes

Formato utilizado	Problema	Solución
Xyz	No tiene colisiones	Implementar ya sea un módulo de generación de colisiones en UE 5 que complemente el plugin "LiDAR Point Cloud Support" o encontrar la forma de que se ejecute el sistema "Nanite" sobre las instancias de las nubes que importan con el plugin.
Fbx (con "Nanite")	No tiene apariencia de puntos	Generar las nubes directamente en UE 5 o proponer un módulo en Blender que genere esferas de una manera más eficiente usando más segmentos geométricos.
Fbx (sin "Nanite")	No tiene colisiones	Implementar un módulo independiente de generación de colisiones en UE 5 o en Blender, ya que no será óptimo utilizar formas básicas como se empleó en este trabajo en objetos irregulares.

De esta manera, se puede apreciar que tanto en UE 5 como en Blender, hay bastantes posibilidades por explorar y desarrollar para complementar esas herramientas en el uso de las nubes de puntos.

Capítulo 5: Conclusiones

Después del desarrollo del presente trabajo, se puede llegar a las siguientes conclusiones.

Se logró desarrollar una metodología que mejora la fidelidad que se tiene en una nube de puntos de baja densidad con el objeto 3D que representa. Ya que, como se observó, se redujo el error (Distancia de Hausdorff) que había en las nubes simplificadas que se utilizó con cada uno de los enfoques desarrollados en la metodología propuesta, de manera que las nubes optimizadas resultantes presentan una reducción de al menos el 30% en la distancia de Hausdorff que se calcula sobre ellas; sin mencionar la mejora visual que obtuvieron con respecto a las nubes simplificadas que se tomaron como punto de partida.

Además, se logró explorar la representación y manipulación de las nubes optimizadas en un entorno de realidad virtual de manera que se pudieron aprovechar tanto las opciones nativas de Unreal Engine 5 (UE 5) como la alternativa de tratar las nubes con un software de modelado 3D (Blender). Como se observó, UE 5 tiene las herramientas básicas para poder combinar el rubro de la realidad virtual con el de las nubes de puntos en proyectos donde el usuario puede desplazarse, visualizar las nubes y manipularlas directamente con los controles de las gafas de realidad virtual, así como también hacerlo con interfaces virtuales mostradas en el entorno de realidad virtual. Sin embargo, haber utilizado Blender para emplear las nubes como un modelo 3D permitió complementar el entorno desarrollado con UE 5 y analizar las fortalezas y las oportunidades de mejora entre ambos, de manera que se pueden idear herramientas que complementen el uso de las nubes para poder utilizarlas como componentes principales en el desarrollo de animaciones, videojuegos e incluso simulaciones.

Por último, el hecho de haber obtenido una premiación a partir de un artículo de congreso confirma que el camino para continuar con este trabajo de investigación es por la vía de buscar más, e incluso mejores, optimizaciones y aplicaciones relevantes con los datos no estructurados tridimensionales.

Futuros trabajos

Se consideran las siguientes opciones como posibles pasos a seguir después de la conclusión de este trabajo:

- Migrar el código del trabajo actual a un lenguaje compilado para mejorar los tiempos de ejecución obtenidos con esta metodología, además de buscar la manera de reducir la complejidad algorítmica.
- Proponer un algoritmo para crear un mayado sobre nubes de puntos de baja densidad.

- Proponer un algoritmo para crear un esqueleto en las nubes de puntos de baja densidad.
- Proponer un módulo que cree nubes de puntos con esferas nativas de Unreal Engine y las unifique (a través de un mayado o incluso de un esqueleto)
- Proponer un módulo independiente en Unreal Engine que sea capaz de generar colisiones por delimitaciones de un usuario en tiempo real de ejecución.
- Proponer un módulo en Blender que cree espacios de colisión en contornos de formas irregulares.
- Explorar las opciones de simulación de Unreal Engine para analizar la implementación de las nubes de puntos en entornos de realidad aumentada.

Capítulo 6: Referencias

[1] A. Kharroubi, R. Hajji, R. Billen, and F. Poux, "CLASSIFICATION AND INTEGRATION OF MASSIVE 3D POINTS CLOUDS IN A VIRTUAL REALITY (VR) ENVIRONMENT," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-2/W17, pp. 165–171, Nov. 2019, doi: 10.5194/isprs-archives-XLII-2-W17-165-2019.

[2] F. Poux and R. Billen, "Voxel-Based 3D Point Cloud Semantic Segmentation: Unsupervised Geometric and Relationship Featuring vs Deep Learning Methods," 2019. doi: 10.3390/IJGI8050213.

[3] Z. Liu *et al.*, "Point Cloud Video Streaming: Challenges and Solutions," *IEEE Netw.*, vol. 35, no. 5, pp. 202–209, 2021, doi: 10.1109/MNET.101.2000364.

[4] L. Li, Z. Li, V. Zakharchenko, J. Chen, and H. Li, "Advanced 3D Motion Prediction for Video-Based Dynamic Point Cloud Compression," *IEEE Transactions on Image Processing*, vol. 29, pp. 289–302, 2020, doi: 10.1109/TIP.2019.2931621.

[5] E. S. Jang *et al.*, "Video-Based Point-Cloud-Compression Standard in MPEG: From Evidence Collection to Committee Draft [Standards in a Nutshell]," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 118–123, May 2019, doi: 10.1109/MSP.2019.2900721.

[6] J. Tang, "Graphic Design of 3D Animation Scenes Based on Deep Learning and Information Security Technology," *Journal of ICT Standardization*, Sep. 2023, doi: 10.13052/jicts2245-800X.1135.

[7] J. Liu, "An adaptive process of reverse engineering from point clouds to CAD models," *Int. J. Comput. Integr. Manuf.*, vol. 33, no. 9, pp. 840–858, Sep. 2020, doi: 10.1080/0951192X.2020.1803501.

[8] J. Fan, L. Ma, and Z. Zou, "A registration method of point cloud to CAD model based on edge matching," *Optik (Stuttg).*, vol. 219, p. 165223, Oct. 2020, doi: 10.1016/j.ijleo.2020.165223.

[9] Y. Liu, A. Obukhov, J. D. Wegner, and K. Schindler, "Point2CAD: Reverse Engineering CAD Models from 3D Point Clouds," Dec. 2023.

[10] A. R. El Sayed, A. El Chakik, H. Alabboud, and A. Yassine, "An efficient simplification method for point cloud based on salient regions detection," *RAIRO - Operations Research*, vol. 53, no. 2, pp. 487–504, Apr. 2019, doi: 10.1051/ro/2018082.

[11] E. Leal, G. Sanchez-Torres, J. W. Branch-Bedoya, F. Abad, and N. Leal, "A Saliency-Based Sparse Representation Method for Point Cloud Simplification," *Sensors*, vol. 21, no. 13, p. 4279, Jun. 2021, doi: 10.3390/s21134279.

[12] O. A. Tapia-Dueñas, H. Sánchez-Cruz, and H. H. López, "3D object simplification using chain code-based point clouds," *Multimed. Tools Appl.*, vol. 82, no. 6, pp. 9491–9515, Mar. 2023, doi: 10.1007/s11042-022-13588-3.

[13] C. Feng, Y. Taguchi, and V. R. Kamat, "Fast plane extraction in organized point clouds using agglomerative hierarchical clustering," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2014, pp. 6218–6225. doi: 10.1109/ICRA.2014.6907776.

[14] Y. Xu, W. Yao, S. Tuttas, L. Hoegner, and U. Stilla, "Unsupervised Segmentation of Point Clouds From Buildings Using Hierarchical Clustering Based on Gestalt Principles," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 11, no. 11, pp. 4270–4286, Nov. 2018, doi: 10.1109/JSTARS.2018.2817227.

[15] Y. Xu, W. Yao, S. Tuttas, L. Hoegner, and U. Stilla, "Unsupervised Segmentation of Point Clouds From Buildings Using Hierarchical Clustering Based on Gestalt Principles," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 11, no. 11, pp. 4270–4286, Nov. 2018, doi: 10.1109/JSTARS.2018.2817227.

[16] J. A. Romero-Guerrero, D. Arenas-Islas, C. Reta, G. E. Bautista-Orduña, I. Juárez-Sosa, and J. M. Suarez-Luna, "Evaluación de formatos de exportación e importación de nubes de puntos para su uso en motores gráficos de realidad virtual," *Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI*, vol. 11, no. 22, pp. 11–19, Jan. 2024, doi: 10.29057/icbi.v11i22.11033.

[17] Eda Kavlakoglu and Vanna Winland, "K-Means clustering," IBM | Think. [Online]. Available: https://www.ibm.com/mx-es/think/topics/k-means-clustering

[18] J. N. Callow, S. M. May, and M. Leopold, "Drone photogrammetry and KMeans point cloud filtering to create high resolution topographic and inundation models of coastal sediment archives," *Earth Surf. Process. Landf.*, vol. 43, no. 12, pp. 2603–2615, Sep. 2018, doi: 10.1002/esp.4419.

[19] B.-Q. Shi, J. Liang, and Q. Liu, "Adaptive simplification of point cloud using -means clustering," *Computer-Aided Design*, vol. 43, no. 8, pp. 910–922, Aug. 2011, doi: 10.1016/j.cad.2011.04.001.

[20] J. Zhou, J. Chen, and H. Li, "An optimized fuzzy K-means clustering method for automated rock discontinuities extraction from point clouds," *International Journal of Rock Mechanics and Mining Sciences*, vol. 173, p. 105627, Jan. 2024, doi: 10.1016/j.ijrmms.2023.105627.

[21] Y. Miao, S. Li, L. Wang, H. Li, R. Qiu, and M. Zhang, "A single plant segmentation method of maize point cloud based on Euclidean clustering and K-means clustering," *Comput. Electron. Agric.*, vol. 210, p. 107951, Jul. 2023, doi: 10.1016/j.compag.2023.107951.

[22] Z. Zhang, B. Yang, B. Wang, and B. Li, "Growsp: Unsupervised semantic segmentation of 3d point clouds," in *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, 2023, pp. 17619–17629.

[23] J. Noble, "¿Qué es la agrupación jerárquica?," IBM | Think. [Online]. Available: www.ibm.com/mx-es/think/topics/hierarchical-clustering#2142864943

[24] J. Gabriel Gomila, "Curso completo de Machine Learning: Data Science en Python," Udemy. [Online]. Available: https://www.udemy.com/course/draft/1606018/

[25] S. Xu, R. Wang, H. Wang, and H. Zheng, "An Optimal Hierarchical Clustering Approach to Mobile LiDAR Point Clouds," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 7, pp. 2765–2776, Jul. 2020, doi: 10.1109/TITS.2019.2912455.

[26] C. Feng, Y. Taguchi, and V. R. Kamat, "Fast plane extraction in organized point clouds using agglomerative hierarchical clustering," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2014, pp. 6218–6225. doi: 10.1109/ICRA.2014.6907776.

[27] C. Chen, G. Li, R. Xu, T. Chen, M. Wang, and L. Lin, "ClusterNet: Deep Hierarchical Cluster Network With Rigorously Rotation-Invariant Representation for Point Cloud Analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4994–5002.

[28] F. Wang *et al.*, "An Automatic Hierarchical Clustering Method for the LiDAR Point Cloud Segmentation of Buildings via Shape Classification and Outliers Reassignment," *Remote Sens. (Basel).*, vol. 15, no. 9, p. 2432, May 2023, doi: 10.3390/rs15092432.

[29] G. K. Knopf, "Hierarchical data clustering approach for segmenting colored three-dimensional point clouds of building interiors," *Optical Engineering*, vol. 50, no. 7, p. 077003, Jul. 2011, doi: 10.1117/1.3599868.

[30] J. Nava, "Qué es Unity y para qué sirve este motor gráfico | IMSED," IMSED. [Online]. Available: https://imsed.com/postgrados/tecnologia/que-es-unity-usos/

[31] Domestika, "Unity vs Unreal. ¿Cuál escoger para desarrollar videojuegos?," Domestika Blog. [Online]. Available: https://www.domestika.org/es/blog/11485-unity-vs-unreal-cual-escoger-para-desarrollar-videojuegos

[32] D. Garrido, R. Rodrigues, A. Augusto Sousa, J. Jacob, and D. Castro Silva, "Point Cloud Interaction and Manipulation in Virtual Reality," in *2021 5th International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, New York, NY, USA: ACM, Jul. 2021, pp. 15–20. doi: 10.1145/3480433.3480437.

[33] E. Neuman-Donihue, M. Jarvis, and Y. Zhu, "FastPoints: A State-of-the-Art Point Cloud Renderer for Unity," Feb. 2023.

[34] Y. Wu, H. Vo, J. Gong, and Z. Zhu, "UnityPIC: Unity Point-Cloud Interactive Core ," *Parallel graphics and visualisation*, Jun. 2021, [Online]. Available: https://doi.org/10.2312/pgv20211044

[35] Inc. Epic Games, "Funciones - Unreal Engine," Epic Games. [Online]. Available: https://www.unrealengine.com/features?lang=es-ES

[36] Facultad de Ingeniería Videojuegos, "Evaluación Técnica de Motores: Arquitectura Unreal Engine 5 vs. Unity (Casos de Uso)," Instituto Cardan. Accessed: May 24, 2026. [Online]. Available: https://institutocardan.com/protocolo/diferencia-unity-vs-unreal-engine

[37] D. Martín-Fuentes and P. M. Cabezos-Bernal, "Universal Point Cloud viewer based on Unreal Engine," *DisegnareCon*, 2023, [Online]. Available: https://doi.org/10.20365/disegnarecon.30.2023.13

[38] S. González-Domínguez, J. Balado, A. Novo, and P. Arias, "TREE DIGITISATION FROM POINT CLOUDS WITH UNREAL ENGINE," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. X-1/W1-2023, pp. 555–560, Dec. 2023, doi: 10.5194/isprs-annals-X-1-W1-2023-555-2023.

[39] Z. Liu *et al.*, "Game Engine-based Point Cloud Visualization and Perception for Situation Awareness of Crisis Indoor Environments," *LBS 2021: Proceedings of the 16th International Conference on Location Based Services*, Nov. 2021.

[40] T. Kato, H. Takahashi, M. Yamashita, A. Doi, and T. Imabuchi, "Development of time-series point cloud data changes and automatic structure recognition system using Unreal Engine," *Artif. Life Robot.*, vol. 30, no. 1, pp. 126–135, Feb. 2025, doi: 10.1007/s10015-024-00983-2.

[41] U-Tad, "Autodesk Maya: el Software de modelado 3D," U-tad. [Online]. Available: https://u-tad.com/autodesk-maya-el-software-de-modelado-3d-que-debes-conocer/

[42] S. Rodrigues-Sierra, "Autodesk Maya vs. Blender en el mundo del modelado 3D," TAI ARTS. [Online]. Available: https://taiarts.com/blog/autodesk-maya-vs-blender/

[43] A. Kumar, *Immersive 3D Design Visualization*. Berkeley, CA: Apress, 2021. doi: 10.1007/978-1-4842-6597-0.

[44] M. Fırat and M. E. Kahraman, "Prefrential dynamics between Unreal Engine and Maya: a comparative analysis of CGI production processes," *Journal for the Interdisciplinary Art and Education*, vol. 6, no. 3, pp. 239–249, 2025, doi: 10.5281/zenodo.16948102.

[45] Y.-T. Tsai, W.-Y. Jhu, C.-C. Chen, C.-H. Kao, and C.-Y. Chen, "Unity game engine: interactive software design using digital glove for virtual reality baseball pitch training," *Microsystem Technologies*, vol. 27, no. 4, pp. 1401–1417, Apr. 2021, doi: 10.1007/s00542-019-04302-9.

[46] A. Watkins, "Creating games with Unity and Maya: how to develop fun and marketable 3D games," *Routledge*, 2012.

[47] Blender Foundation, "Features — blender," Blender. [Online]. Available: https://www.blender.org/features/

[48] E. Pérez, A. Sánchez-Hermosell, and P. Merchán, "TLSynth: A Novel Blender Add-On for Real-Time Point Cloud Generation from 3D Models," *Remote Sens. (Basel).*, vol. 17, no. 3, p. 421, Jan. 2025, doi: 10.3390/rs17030421.

[49] G. Gorup, Ž. Lesar, M. Marolt, and C. Bohak, "Procedural Point Cloud and Mesh Editing for Urban Planning Using Blender," *Land (Basel).*, vol. 14, no. 4, p. 815, Apr. 2025, doi: 10.3390/land14040815.

[50] Z. Yu, J. Zheng, X. Yu, K. Lu, and B. Gu, "Simulating 3D fabric drape models with diversity based on point cloud random resampling and blender," *The Journal of The Textile Institute*, vol. 117, no. 3, pp. 532–545, Mar. 2026, doi: 10.1080/00405000.2025.2510932.

[51] M. Chen *et al.*, "3D Modeling and Web Visualization of Cloud Campus: A Technical Route Integrating Python Point Cloud Processing, Unity3D, and Three.js," in *Proceedings of the 2026 3rd International Conference on Informatics Education and Computer Technology Applications*, New York, NY, USA: ACM, Jan. 2026, pp. 501–508. doi: 10.1145/3802133.3802215.

[52] K. Asrorov, " GAME DEVELOPMENT WITH UNITY 3D AND BLENDER 3D," *Интернаука*, vol. 15, no. 144, p. 52, 2020.

[53] J. Plowman, "3D Game Design with Unreal Engine 4 and Blender," *Packt Publishing Ltd*, 2016.

[54] İ. Ç. Gençtürk, B. Erkek, and E. Ayyıldız, "Integrating Photogrammetric 3D City Models into Virtual Reality: A Methodological Approach Using Unreal Engine," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLVIII-M-6–2025, pp. 353–358, May 2025, doi: 10.5194/isprs-archives-XLVIII-M-6-2025-353-2025.

[55] G. Acurero, "Pico 4, los lentes de realidad virtual de ByteDance que compiten con los Meta Quest ," FayerWayer. [Online]. Available: https://www.fayerwayer.com/entretenimiento/2022/12/16/pico-4-los-lentes-de-realidad-virtual-de-bytedance-que-compiten-con-los-meta-quest-2/

[56] PICO, "Blog | PICO Developer." [Online]. Available: https://developer.picoxr.com/blog/

[57] Meta, "Meta Horizon Developer Center," Meta Horizon Blog. [Online]. Available: https://developers.meta.com/horizon/

[58] M. Giammetti, A. A. Cantone, and P. Battistoni, "OpenXR-Based Hand Motion Recording for VR Applications Using Unity and VR Headset," *Meta*, vol. 3, p. 3s, 2025.

[59] M. Pitteloud and A. Widmer, "Comparison of Meta Quest 3 and Pico 4 Enterprise for Mixed Reality Applications," Haute Ecole de Gestion Valais, 2025.

[60] L. Cheng, M. Schreiner, and A. Kunz, "Comparing Tracking Accuracy in Standalone MR-HMDs: Apple Vision Pro, Hololens 2, Meta Quest 3, and Pico 4 Pro," in *30th ACM Symposium*

on Virtual Reality Software and Technology, New York, NY, USA: ACM, Oct. 2024, pp. 1–2. doi: 10.1145/3641825.3689518.

[61] C. Ramos *et al.*, "Enhancing Automotive Usability Testing and User Experience: Insights from Virtual Reality and Eye Tracking Integration," in *2025 IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality (AIxVR)*, IEEE, Jan. 2025, pp. 429–434. doi: 10.1109/AIxVR63409.2025.00081.

[62] R. Padamadan, N. Bogahawatta, T. D. Ranagalage, and K. Thilakarathna, "Fusing Point Clouds and Wireless Signals for Seamless Interaction in Smart Spaces," in *Proceedings of the 3rd ACM Workshop on Mobile Immersive Computing, Networking, and Systems*, New York, NY, USA: ACM, Nov. 2025, pp. 22–28. doi: 10.1145/3742889.3768348.

[63] S. Cantarelli, G. M. Santi, and D. Francia, "Real-Time Incremental Point Cloud Acquisition and Spatial Bounding Box Filtering Using Lidar Sensors in Virtual Reality Environments," 2026, pp. 279–291. doi: 10.1007/978-3-032-14950-3_23.

[64] M. A. Rivas-Juarez, O. A. Tapia-Dueñas, and H. Sánchez-Cruz, "Hausdorff Distance Optimization in Low-Density Point Clouds," in *Pattern Recognition*, A. P. López-Monroy, A. Rosales-Pérez, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, and J. A. Olvera-López, Eds., Cham: Springer Nature Switzerland, 2025, pp. 13–25.

Capítulo 7: Glosario

- **Estadísticos poblacionales**: es un valor numérico que describe un a característica de una población completa.
- **Segmentación**: técnica de dividir un conjunto de elementos en conjuntos más pequeños, normalmente llamados clústeres, que comparten características similares.
- **Clúster**: un clúster en segmentación es un grupo de elementos que comparten características similares, como posición, densidad, entre otras.
- **Centro de masa/ centroide**: es el centro geométrico de un conjunto de datos. Se obtiene al calcular el punto promedio de las posiciones que conforman el conjunto de datos.
- **Rigging**: es el proceso de crear un esqueleto digital para un modelo 3D y asignarle un sistema de controles para otorgar funcionalidad y/o movilidad al esqueleto digital creado.
- **Plugins**: en el entorno de los software de computadora, son programas, herramientas o módulos complementarios que se integran a una aplicación base con el fin de expandir, o mejorar, su funcionalidad para solventar problemas específicos.
- **CGI**: son las siglas en inglés de "Computer-Generated Imagery", que significa "imágenes generadas por computadora". Se refiere a las técnicas de modelado, animación y creación de efectos visuales que se emplean para crear tanto personajes como objetos y efectos especiales en entornos virtuales.
- **Código abierto**: en el rubro de la programación, todos los programas que son de código abierto son gratuitos y de dominio público, lo que significa que cualquier persona puede acceder a ellos, modificarlos y compartirlos libremente.
- **API**: son las siglas en inglés de "Application Programming Interface", que significa "Interfaz de programación de aplicaciones". Es un protocolo que permite a dos o más aplicaciones comunicarse entre sí sin tener que estar escritas en el mismo lenguaje de programación.
- **Modelos Fotogramétricos**: son representaciones 3D obtenidas a partir de diferentes fotografías.
- **Realidad mixta / realidad aumentada**: área de la informática que permite mezclar elementos virtuales (objetos, pantallas, entre otros) en entornos de la vida real.
- **Divide y vencerás**: paradigma para resolución de problemas que consiste en dividir un problema complejo en subprocesos pequeños e independientes, de manera que se pueda resolver de una manera más eficiente y eficaz.
- **Voxel**: es un acrónimo de "volumetric pixel" que significa "píxel volumétrico". Es el equivalente al píxel, pero en un entorno tridimensional. En otras palabras, es una unidad cúbica básica que compone un objeto o un espacio tridimensional.
- **Voxelización**: proceso de representar un objeto 3D en voxeles.

- **Códigos de cadena**: conjunto de símbolos que representan el contorno de la forma de un objeto 2D o 3D. Cada símbolo en estos códigos representa el cambio de dirección que sigue la forma de un objeto con tal de preservar su forma original.
- **IDE**: por sus siglas en inglés "Integrated Development Environment" significa "Entorno de desarrollo integrado". Es un software de desarrollo de código que permite tanto escribir como ejecutar programas sobre él. Usualmente se especializan en un lenguaje de programación específico.
- **Complejidad algorítmica**: es una métrica que indica la eficiencia de un algoritmo. Utiliza la notación "Big O" para indicar cómo crecen los tiempos de ejecución con respecto a los datos que entran en un algoritmo. Por ejemplo, O(n) significa que el tiempo de ejecución es lineal y que crecerá a la par de los datos de entrada y $O(n^2)$ significa que el tiempo de ejecución crecerá exponencialmente con respecto a los datos de entrada.
- **Lenguaje interpretado**: lenguaje de programación que se traduce línea por línea a lenguaje de máquina mientras es ejecutado. Suelen ser más lentos en tiempos de ejecución.
- **Lenguaje compilado**: lenguaje de programación que se traduce completamente a lenguaje de máquina por un compilador antes de ser ejecutado. Suelen ser más rápidos en tiempos de ejecución.
- **Colisiones**: en el ámbito del modelado 3D, las colisiones son un proceso computacional que determina si dos o más objetos se tocan entre sí.

Anexos

- **Listado de artículos premiados en el MCPR2025**

AWARDS

The MCPR-IAPR Best Paper Award and MCPR-IAPR Student Paper Award winners will be invited to send extended papers to the special section devoted to MCPR in the "Pattern Recognition Letters" Journal. Authors must consider that an extended paper means at least a 40% difference in theory content, experimental results, and references. In addition, the extended versions will be peer-reviewed according to the journal's rules.

MCPR 2025 AWARD WINNERS

Best Paper Award

Deep Neural Networks and Log-Mel Spectrogram for Emotion Recognition through Spanish Speech
Juan Alberto Ramirez-Quintana, Ricardo Steve Ang-Foster, Mario Ignacio Chacon-Muguia, Abimael Guzman-Pando and Alma Delia Corral-Saenz

Best Student Paper Award

Hausdorff Distance Optimization in Low-Density Point Clouds
Misael A. Rivas-Juarez *et al.*

Best Poster Award

Crime Prediction Using Computer Vision: Data Selection, Balancing, and Representation
Luis Enrique Morales-Márquez *et al.*

Figura 126 – Lista de artículos premiados e invitados a publicar en "Pattern Recognition Letters"