



ugr

Universidad  
de Granada

TRABAJO FIN DE GRADO  
GRADO EN INGENIERÍA INFORMÁTICA

# Desarrollo de una aplicación de modelado 3D

---

Motor de renderizado en tiempo real basada en SDFs

**Autor**

Pablo Cantudo Gómez

**Directores**

Juan Carlos Torres Cantero y Luis López Escudero



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE  
TELECOMUNICACIÓN

Granada, Septiembre de 2025







# Desarrollo software de una aplicación de modelado 3D basada en Signed Distance Functions (SDF)

Pablo Cantudo Gómez

**Palabras clave:** WebGPU, C++, ray marching, motor de renderizado, funciones de distancia con signo (SDF)

## Resumen

Este trabajo presenta el desarrollo de Copper, un motor de renderizado 3D en tiempo real implementado en C++ utilizando la API WebGPU a través de Dawn. El motor emplea técnicas de ray marching sobre funciones de distancia con signo (SDF), lo que permite una representación eficiente y flexible de geometría tridimensional. Una de las principales ventajas de utilizar SDF es la capacidad de expresar de forma directa y compacta operaciones booleanas entre primitivas geométricas —como uniones, intersecciones o diferencias— mediante simples expresiones algebraicas. Además, a diferencia de los métodos tradicionales basados en mallas o polígonos, las SDF permiten aplicar operaciones booleanas suaves, como la unión con suavizado (smooth union), de forma prácticamente trivial desde el punto de vista computacional.

Este enfoque simplifica notablemente la construcción de formas complejas, evitando problemas típicos de la computación geométrica como el manejo de vértices, normales o topologías complejas. También se facilita la animación y transformación de objetos mediante funciones continuas. El motor incluye un sistema de sombreado en WGSL, con soporte para sombras suaves, operaciones modulares sobre la escena, selección de objetos, carga y guardado de modelos. Los resultados demuestran que el uso de SDF no solo ofrece un modelo más elegante para definir geometría, sino que permite construir escenas visualmente complejas con menos código y una mayor expresividad gráfica. En conjunto, el sistema demuestra la viabilidad técnica y creativa del uso de SDF en entornos modernos.



# Development of a 3D modeling application based on Signed Distance Functions/Fields (SDF)

Pablo Cantudo Gomez

**Keywords:** WebGPU, C++, ray marching, rendering engine, Signed Distance Functions/Fields (SDF)

## Abstract

This work presents the development of Copper, a real-time 3D rendering engine implemented in C++ using the WebGPU API through Dawn. The engine employs ray marching techniques over Signed Distance Functions/Fields (SDF), enabling an efficient and flexible representation of three-dimensional geometry. One of the main advantages of using SDF is the ability to directly and compactly express Boolean operations between geometric primitives — such as unions, intersections, or differences — through simple algebraic expressions. Furthermore, unlike traditional mesh- or polygon-based methods, SDF allows for smooth Boolean operations, such as smooth union, in a virtually trivial manner from a computational standpoint.

This approach greatly simplifies the construction of complex shapes, avoiding typical problems in geometric computing such as handling vertices, normals, or complex topologies. It also facilitates the animation and transformation of objects through continuous functions. The engine includes a shading system in WGSL, with support for soft shadows, modular scene operations, object selection, and model loading and saving. The results show that the use of SDF not only offers a more elegant model for defining geometry, but also enables the creation of visually complex scenes with less code and greater graphical expressiveness. Overall, the system demonstrates the technical and creative feasibility of using SDF in modern environments.





---

Yo, **Pablo Cantudo Gómez**, alumno de la titulación Grado en ingeniería informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 78243264, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Pablo Cantudo Gómez

Granada a x de Septiembre de 2025.



---

D. **Tutores**, Profesores del Área de x del Departamento x de la Universidad de Granada.

**Informan:**

Que el presente trabajo, titulado ***Nombre***, ha sido realizado bajo su supervisión por **Tutores**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 11 de Noviembre de 2023.

**Los directores:**

**Tutores**



# Agradecimientos

Quiero expresar mi agradecimiento a todas las personas que han contribuido directa o indirectamente a la realización de este trabajo. En primer lugar, a mi tutor Juan Carlos Torres Cantero y cotutor Luis López Escudero por la ayuda durante el desarrollo y la corrección del proyecto, a mis amigos por las discusiones y las ideas, y a mi familia por su apoyo incondicional.

Finalmente, me gustaría mencionar a la comunidad de desarrolladores y recursos abiertos, especialmente los foros, artículos y proyectos relacionados con WebGPU, SDF y ray marching, cuya documentación y ejemplos han sido una fuente de aprendizaje invaluable.



# Índice general

<b>1. Introducción</b>	<b>19</b>
1.1. Motivación . . . . .	20
1.2. Descripción del problema . . . . .	20
1.3. Objetivos . . . . .	20
<b>2. Fundamentos teóricos</b>	<b>21</b>
2.1. Signed Distance Functions (SDF) . . . . .	21
2.2. Ray Marching . . . . .	21
<b>3. Arquitectura</b>	<b>23</b>
3.1. Tecnologías utilizadas . . . . .	23
3.2. Estructura del proyecto . . . . .	23
<b>4. Implementación: Ventana</b>	<b>25</b>
<b>5. Implementación: Controles</b>	<b>27</b>
<b>6. Implementación: Shaders</b>	<b>29</b>
<b>7. Funcionalidades adicionales</b>	<b>31</b>
<b>8. Pruebas</b>	<b>33</b>
<b>9. Conclusiones</b>	<b>35</b>
<b>Bibliografía</b>	<b>35</b>
<b>A. Código</b>	<b>37</b>
<b>B. Escenas</b>	<b>39</b>
<b>Glosario</b>	<b>39</b>





# Índice de figuras



# Capítulo 1

## Introducción

El desarrollo de los gráficos tridimensionales por computadora ha sido un campo de investigación activo y en constante evolución desde sus inicios. La capacidad de crear representaciones visuales de objetos y escenas en tres dimensiones ha revolucionado diversas industrias, desde el entretenimiento hasta la medicina y la ingeniería.

A medida que la tecnología avanza, también lo hacen las técnicas y herramientas utilizadas para generar gráficos 3D, lo que plantea nuevos desafíos y oportunidades para los investigadores y desarrolladores.

En sus inicios, la generación de gráficos 3D estaba limitada por la capacidad de cómputo y se basaba en *pipelines* gráficos fijos compuestos por etapas de transformación, iluminación y rasterización.

Posteriormente, con la llegada de los *shaders* programables en GPU (Nvidia GeForce 3 en 2001), fue posible sustituir los *pipelines* fijos por *pipelines* programables, lo que abrió un abanico de posibilidades para la creación de efectos visuales complejos y personalizados, como los algoritmos no basados en polígonos.

Esto impulsó la investigación en técnicas de representación más avanzadas, como el *ray tracing* y, en particular, el *ray marching*.

En el contexto del renderizado basado en funciones implícitas, las *Signed Distance Functions* (SDF) no son una invención reciente, sino que tienen sus raíces en trabajos mucho más antiguos.

El concepto de combinar funciones implícitas mediante operaciones booleanas se remonta al trabajo de Ricci en 1972 [Ric73], y fue ampliado en 1989 por B. Wyvill y G. Wyvill con el modelado de *soft objects* [WMW86].

Ese mismo año, Sandin, Hart y Kauffman aplicaron *ray marching* a SDF para renderizar fractales tridimensionales [HSK89].

Posteriormente, en 1995, Hart documentó de nuevo la técnica, aunque la denominó erróneamente *Sphere Tracing* [Har96].

La popularización moderna de las SDF en el ámbito del *renderizado en tiempo real* se debe en gran parte a la comunidad *demoscene*, especialmente

a partir de mediados de la década de 2000.

Trabajos como el de Crane (2005) y Evans (2006) introdujeron la idea de restringir el campo a una distancia euclidiana real, mejorando el rendimiento y la calidad visual.

Sin embargo, el uso de esta técnica no está tan extendido en aplicaciones de renderizado en tiempo real, a pesar de su potencial para crear gráficos visuales y eficientes.

### **1.1. Motivación**

### **1.2. Descripción del problema**

### **1.3. Objetivos**

## Capítulo 2

# Fundamentos teóricos

2.1. Signed Distance Functions (SDF)

2.2. Ray Marching



## Capítulo 3

# Arquitectura

3.1. Tecnologías utilizadas

3.2. Estructura del proyecto





## Capítulo 4

# Implementación: Ventana



## Capítulo 5

# Implementación: Controles



## Capítulo 6

# Implementación: Shaders



## Capítulo 7

# Funcionalidades adicionales





## Capítulo 8

# Pruebas



## Capítulo 9

# Conclusiones



# Bibliografía

- [Ric73] A. Ricci. “A Constructive Geometry for Computer Graphics”. En: *The Computer Journal* 16.2 (mayo de 1973), págs. 157-160. DOI: <http://dx.doi.org/10.1093/comjnl/16.2.157>.
- [WMW86] Geoff Wyvill, Craig McPheeters y Brian Wyvill. “Data Structure for Soft Objects”. En: *The Visual Computer* 2.4 (1986), págs. 227-234. DOI: 10.1007/BF01900346. URL: <https://link.springer.com/article/10.1007/BF01900346>.
- [HSK89] John C. Hart, Daniel J. Sandin y Louis H. Kauffman. “Ray Tracing Deterministic 3-D Fractals”. En: *Computer Graphics* 23.3 (jul. de 1989), págs. 289-296. DOI: 10.1145/74334.74340. URL: <https://www.cs.drexel.edu/~david/Classes/Papers/rtqjs.pdf>.
- [Har96] John C. Hart. “Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces”. En: *The Visual Computer* 12.10 (1996), págs. 527-545. DOI: 10.1007/s003710050084. URL: <https://link.springer.com/article/10.1007/s003710050084>.
- [Qui08] Iñigo Quílez. *Raymarching Distance Fields*. <https://iquilezles.org/articles/raymarchingdf/>. Accessed: 2025-08-11. 2008.



Apéndice A

Código





## Apéndice B

### Escenas



# Glosario

**Constructive Solid Geometry (CSG)** Método de modelado geométrico que combina primitivas mediante operaciones booleanas como unión, intersección y diferencia..

**Demoscene** Comunidad de programadores, artistas y músicos que crean producciones audiovisuales en tiempo real para mostrar destreza técnica y creatividad..

**Función Implícita** Función matemática que describe una superficie como el conjunto de puntos que satisfacen una ecuación dada, sin necesidad de una parametrización explícita..

**Hypertexture** Técnica de Ken Perlin y Louis Hoffert para renderizar volúmenes procedurales, que sirvió de base para el desarrollo del ray marching..

**Ray Marching** Técnica de renderizado que recorre un rayo en pasos discretos para encontrar intersecciones con superficies definidas implícitamente..

**Signed Distance Function (SDF)** Función que, dado un punto en el espacio, devuelve la distancia mínima a la superficie más cercana, con signo positivo si el punto está fuera y negativo si está dentro..

**Smooth Blending** Técnica para suavizar las transiciones entre primitivas geométricas en modelado implícito, evitando uniones abruptas..

**Sphere Tracing** Método propuesto por Hart en 1995 para recorrer campos de distancia en el renderizado de superficies implícitas..



