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, seleccion 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 en- gine implemented in C++ using the WebGPU API through Dawn. The engi- ne 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 ex- pressions. Furthermore, unlike traditional mesh- or polygon-based methods, SDF allows for smooth Boolean operations, such as smooth union, in a vir- tually 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 proyec- to, a mis amigos por las discusiones y las ideas, y a mi familia por su apoyo in- condicional.

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

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

Como cualquier persona nacida en los dos mil, he crecido rodeado de videojuegos y el avance en la tecnología de gráficos 3D ha sido un aspecto fascinante de esta industria. Durante la carrera de informática, estudié diversas asignaturas relacionadas con gráficos por computadora, cuyos proyectos despertaron y consolidaron mi interés en este campo.

El desarrollo de motores gráficos y técnicas de renderizado siempre me ha resultado un área especialmente atractiva, no solo por su complejidad técnica, sino también por el impacto directo que tienen en sectores como el entretenimiento, la simulación o la realidad virtual. A lo largo de mis estudios me encontré con herramientas muy potentes, pero también con la dificultad que implica dominarlas o adaptarlas a entornos experimentales. Esto me llevó a plantearme la posibilidad de crear una aplicación propia que sirviera como espacio de exploración.

La motivación principal de este trabajo es profundizar en tecnologías emergentes, en concreto **WebGPU**, un estándar reciente que promete unificar el desarrollo gráfico multiplataforma con un acceso eficiente a las GPU modernas. Asimismo, me interesaba experimentar con el modelado mediante **funciones de distancia (SDF)**, que representan una alternativa flexible al modelado poligonal clásico. Considero que la combinación de ambas tecnologías constituye un terreno de investigación con un gran potencial, tanto en aplicaciones prácticas como en entornos educativos.

Finalmente, este proyecto me ofrece la oportunidad de afianzar mis conocimientos en programación gráfica, shaders y arquitecturas modernas de GPU, a la vez que desarrollo un software propio que pueda servir de base para futuros trabajos de investigación o aplicaciones más complejas en el ámbito del diseño 3D.

1.2. Descripción del problema

El campo del modelado y renderizado 3D ha estado tradicionalmente dominado por herramientas complejas y de gran envergadura, como Blender, Maya o 3ds Max. Si bien estas aplicaciones ofrecen una gran potencia

y versatilidad, presentan también limitaciones importantes: requieren elevados recursos de hardware, poseen curvas de aprendizaje pronunciadas y no siempre resultan adecuadas para entornos de experimentación ligera o proyectos educativos.

Por otro lado, las API gráficas más extendidas, como OpenGL o DirectX, han demostrado su eficacia a lo largo de los años, pero presentan restricciones en cuanto a eficiencia y portabilidad en plataformas modernas. El reciente estándar **WebGPU** surge como respuesta a estas carencias, ofreciendo un modelo de programación más cercano al hardware y multiplataforma, con el objetivo de unificar el desarrollo gráfico en navegadores y aplicaciones nativas.

En el ámbito del modelado, el paradigma poligonal sigue siendo el más utilizado, pero alternativas como las **funciones de distancia (SDF)** permiten representar geometrías complejas de manera más compacta y flexible, facilitando la combinación de primitivas y operaciones booleanas. Sin embargo, la integración de estas técnicas en aplicaciones prácticas todavía es limitada, especialmente en combinación con tecnologías emergentes como WebGPU.

El problema que aborda este trabajo consiste en la falta de herramientas ligeras que sirvan como demostración y entorno de experimentación para el modelado y renderizado basados en SDF sobre WebGPU.

1.3. Objetivos

Objetivo general

El objetivo principal de este Trabajo de Fin de Grado es el desarrollo de una aplicación de diseño 3D basada en **WebGPU** y en técnicas de **modelado mediante funciones de distancia (SDF)**, que permita explorar y demostrar el potencial de estas tecnologías como alternativa al modelado poligonal clásico y como herramienta de experimentación en el ámbito de los gráficos por computadora.

Objetivos específicos

Para alcanzar este objetivo general, se plantean los siguientes objetivos específicos:

- Investigar y comprender en profundidad el funcionamiento de la API WebGPU y su integración en aplicaciones nativas mediante la librería Dawn.
- Diseñar e implementar un motor de renderizado basado en *ray marching* sobre funciones de distancia, capaz de representar primitivas y combinaciones mediante operaciones booleanas y suaves.

- Incorporar técnicas de sombreado y efectos visuales (iluminación, sombras suaves, antialiasing) que mejoren la calidad del renderizado.
- Desarrollar una interfaz gráfica sencilla que permita al usuario interactuar con la escena y manipular las primitivas.

Capítulo 2

Fundamentos teóricos

En este capítulo se presentan los conceptos fundamentales necesarios para comprender el desarrollo de la aplicación *Copper*. Se revisan las técnicas de modelado y renderizado en gráficos por computador, con especial énfasis en las funciones de distancia y el algoritmo de *ray marching*, así como las tecnologías utilizadas en la implementación: WebGPU, Dawn, WGSL, GLFW e ImGui.

2.1. Gráficos por computador

Los gráficos por computador constituyen el conjunto de técnicas y algoritmos destinados a generar imágenes a partir de descripciones matemáticas de escenas tridimensionales. Existen principalmente dos paradigmas de renderizado:

- **Rasterización**: método utilizado tradicionalmente en APIs como OpenGL o DirectX. Consiste en proyectar los triángulos que forman la geometría sobre la pantalla y aplicar transformaciones y sombreados mediante un *pipeline* gráfico fijo o programable.
- **Renderizado basado en rayos**: consiste en simular la trayectoria de rayos de luz en la escena. Incluye técnicas como el *ray tracing*, que calcula intersecciones exactas con geometría, y el *ray marching*, que recorre iterativamente el espacio definido por funciones de distancia.

2.2. Modelado 3D

2.2.1. Modelado poligonal

El modelado poligonal se basa en representar los objetos mediante mallas de triángulos. Es el enfoque dominante en la industria por su compatibilidad con el hardware gráfico actual. Sin embargo, presenta algunas limitaciones:

elevada complejidad para representar operaciones booleanas entre objetos, necesidad de gestionar grandes volúmenes de datos y dificultades para representar superficies suaves sin subdivisión intensiva.

2.2.2. Funciones de distancia (SDF)

Una **Signed Distance Function (SDF)** es una función escalar que, dado un punto $p \in R^3$, devuelve la distancia mínima hasta la superficie de un objeto. El signo indica si el punto está en el interior ($d(p) < 0$), sobre ($d(p) = 0$) o en el exterior ($d(p) > 0$) de la superficie.

Por ejemplo:

- Esfera de radio r y centro c:

$$d(p) = \|p - c\| - r$$

- Caja de dimensiones (x, y, z) y centro c:

$$d(p) = \| \max(|p - c| - (x, y, z), 0)\|$$

Las SDF permiten combinar objetos fácilmente mediante operaciones booleanas:

- Unión: $d(p) = \min(d_1(p), d_2(p))$
- Intersección: $d(p) = \max(d_1(p), d_2(p))$
- Resta: $d(p) = \max(d_1(p), -d_2(p))$

Estas operaciones pueden suavizarse introduciendo funciones de interpolación continua, lo que facilita la creación de formas orgánicas.

2.3. Renderizado con *Ray Marching*

El *ray marching* es un método de renderizado que utiliza funciones de distancia para determinar las intersecciones de rayos con la geometría. El algoritmo se basa en los siguientes pasos:

1. Se lanza un rayo desde la cámara en una dirección determinada.
2. Se evalúa la SDF en la posición actual para obtener la distancia mínima a la superficie.
3. El punto se avanza a lo largo del rayo esa distancia.
4. El proceso se repite hasta que la distancia es menor que un umbral (intersección) o se supera una distancia máxima (sin colisión).

Las normales se calculan mediante el gradiente numérico de la SDF, y a partir de ellas se aplican modelos de iluminación. El método permite añadir sombras suaves mediante la evaluación adicional de distancias a lo largo de los rayos secundarios.

Entre sus limitaciones destacan el elevado coste computacional y la aparición de artefactos cuando el umbral de convergencia no es adecuado.

2.4. APIs gráficas y WebGPU

Las APIs gráficas tradicionales como OpenGL, DirectX o Vulkan permiten acceder al hardware gráfico, pero presentan problemas de portabilidad o complejidad. WebGPU surge como un estándar moderno que busca unificar el desarrollo gráfico en navegadores y aplicaciones nativas, ofreciendo:

- Acceso de bajo nivel a la GPU, similar a Vulkan o Metal.
- Multiplataforma: soportado en navegadores y entornos nativos.
- Un modelo más seguro y predecible de ejecución de shaders.

En este proyecto se emplea **Dawn**, la implementación nativa de WebGPU desarrollada por Google, que permite ejecutar programas fuera del navegador.

2.5. Lenguaje WGSL

El lenguaje **WGSL** (WebGPU Shading Language) es el estándar asociado a WebGPU para la programación de shaders. Su sintaxis se inspira en lenguajes como Rust y GLSL, con un enfoque en la seguridad y la legibilidad. WGSL permite programar *compute shaders*, *vertex shaders* y *fragment shaders*, siendo en este proyecto esencial para implementar el algoritmo de *ray marching*.

2.6. Herramientas auxiliares

2.6.1. GLFW

GLFW es una librería multiplataforma que gestiona ventanas, entrada por teclado/ratón y el contexto gráfico. Se utiliza en este proyecto para crear la ventana principal y manejar la interacción del usuario.

2.6.2. ImGui

ImGui es una librería de interfaz gráfica inmediata (*Immediate Mode GUI*), ligera y especialmente útil en entornos de desarrollo y prototipado. En este proyecto permite modificar en tiempo real parámetros de la escena, como la posición o dimensiones de las primitivas.

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.

[Quí08] 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..