

---

This is the **published version** of the bachelor thesis:

Velázquez Gómez, Raúl. *Entorn gràfic amb Vulkan i Ray Marching*.. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Enginyeria Informàtica)

---

This version is available at <https://ddd.uab.cat/record/334420>

under the terms of the  license.

# Entorn gràfic amb Vulkan i Ray Marching

Raúl Velázquez Gómez

28 de juny de 2026

**Resum**– Aquest projecte desenvolupa un entorn gràfic simple amb C++ i l'API gràfica Vulkan, amb una aproximació més moderna al món dels gràfics per ordinador, per tal que els estudiants interessats en aquesta llibreria puguin introduir-se amb coneixements gràfics avançats. Vulkan ens permet tenir més control sobre la targeta gràfica que OpenGL, però la seva complexitat és molt més elevada, donat que treballa a més baix nivell. D'altra banda, s'exploraran les possibilitats que ofereix l'algoritme “ray marching”, doncs permet treballar amb un enfocament basat en raigs que posen a prova les capacitats de Vulkan. Els raigs surten de cada píxel en pantalla fins a col·lidir amb l'entorn 3D. En aquest cas, es farà servir per treballar amb fractals, encara que és aplicable en altres àmbits. A més, és una oportunitat perfecta per testear una implementació que pugui aprofitar les característiques de Vulkan, doncs s'utilitza el potencial de la part gràfica i la de computació.

**Paraules clau**– Gràfics per ordinador, Vulkan, Ray Marching, Pipeline gràfica, Pipeline de computació, Fractals.

**Abstract**– This project showcases a simple graphics engine in C++ and Vulkan, using modern approaches to computer graphics, so that students interested in this field can prepare themselves for the world of work, as well as foster research through new projects. Vulkan is a graphics API that gives us more control over the graphics card than OpenGL, but its complexity is much higher, as it operates at a lower level. Furthermore, the possibilities offered by the ‘ray marching’ algorithm will be explored, as it allows me to work with a ray-based approach. In this algorithm, rays will be cast from each pixel on the screen until they collide with the 3D scene. In this use case, I will use it to work with fractals, although it is applicable in other areas. Moreover, it will be a perfect opportunity to test an implementation that can take advantage of the features provided by Vulkan, as I will be utilising the potential of both the graphics and compute components.

**Keywords**– Computer Graphics, Vulkan, Ray Marching, Graphics Pipeline, Compute Pipeline, Fractals.



## 1 Introducció

**D**esprés de cursar Visualització Gràfica Interactiva (VGI) [1], una assignatura de tercer curs del grau d'enginyeria informàtica, la proposta d'un entorn de programació amb C++ [2] i l'API [3] OpenGL [4] em va semblar adient per l'àmbit acadèmic i perfecte per una primera aproximació al món dels gràfics per ordinador [5]. No obstant això, m'hauria agradat aprofundir més i tenir recursos per continuar la meua formació en l'àmbit dels gràfics per computador, ja que OpenGL [6] s'està quedant cada

vegada més enrere per no aprofitar tot el potencial de les targetes gràfiques [7] actuals. Khronos Group [8, 9], el consorci oficial que va crear OpenGL, està enfocat en el desenvolupament de Vulkan [10, 11] i l'última versió que existeix d'OpenGL és de 2017 [6]. Per tant, la manca de suport d'aquest darrer suposa un problema important. Una manera per solucionar aquesta problemàtica en l'assignatura seria definir un entorn gràfic amb Vulkan [12] per deixar pels estudiants. Crec que resultaria interessant fer aplicacions gràfiques utilitzant aquesta API. A més, seria una molt bona opció per a ajustar-se a tecnologies més modernes [11]. Vulkan no és exclusiu per als gràfics per computador i, per tant, es pot usar pel càlcul intensiu amb dades [13] fent servir el potencial de càlcul de les targetes gràfiques [7]. Vulkan ha anat guanyant importància en els darrers anys per la seva contínua evolució [14, 15]. Això és degut a les eines de control que ofereix sobre la targeta gràfica

• E-mail de contacte: raulvelazquezgomez2@gmail.com

• Menció: Computació

• Treball tutoritzat per: Enric Martí Gòdia (Departament de Ciències de la Computació)

• Curs 2025/26

[16], adquirint més llibertat i maniobrabilitat, formes d'optimització i també més responsabilitat. Diferents experts afirmen que no és una llibreria per iniciar-se als gràfics per ordinador donada la seva complexitat si s'aborda des de zero. Per això, una idea interessant és disposar d'un entorn bàsic de programació gràfica amb les característiques més essencials de Vulkan i anar escalant en aquest entorn amb funcionalitats més complexes per adaptar-se a les tecnologies més modernes. D'altra banda, l'exploració de l'algoritme "ray marching" [17] permet aprendre sobre diverses tècniques per renderitzar dades a partir de raigs i visualitzar-les en temps real. El seu funcionament es basa en la definició dels raigs fins a col·lidir amb les dades en escena per poder fer una visualització basada en la distància des de la càmera. Això és útil per visualitzar objectes amb molt de detall (com poden ser fractals [18]) sense haver d'utilitzar models geomètrics modelats en 3D [19] o per crear volums mitjançant la incisió dels raigs en objectes amb densitat o capes [20, 21], entre moltes altres funcionalitats. Es pretén que aquest coneixement estigui disponible pels estudiants com a exemple de l'entorn i aprofundir en l'àmbit dels gràfics per computador [17, 21].

Per tot això, el principal objectiu del projecte és el de desenvolupar un entorn gràfic bàsic per renderitzar amb Vulkan i C++. A més a més, poder utilitzar aquest entorn per crear una aplicació de fractals 3D amb l'algoritme "ray marching", per a provar les capacitats de l'entorn.

Les funcionalitats o objectius més concrets de l'entorn gràfic amb Vulkan:

- Implementar una pipeline gràfica i una pipeline de computació per a càlcul intensiu de dades amb Vulkan.
- Sistema de càrrega d'objectes 3D amb la llibreria Assimp.
- Implementar la llibreria Dear ImGui per tal d'oferir una interfície gràfica de manera simple.

Funcionalitats o objectius més concrets de l'algoritme de "ray marching":

- Fer una implementació en els compute shaders de l'algoritme.
- Crear i visualitzar fractals tridimensionals.

Pel que fa a la metodologia del treball, després d'haver fet un estudi de les metodologies "Agile" [22], penso que Kanban [23, 24] és la més adient per al projecte. A més, he pogut fer ús de les meves aplicacions i em trobo còmode treballant d'aquesta manera. És una metodologia que m'aporta flexibilitat i una bona organització, a més de la simplicitat i l'aspecte visual per no perdre l'enfocament en les tasques.

Les eines de desenvolupament utilitzades són:

- Vulkan SDK 1.4.304.1 [25] com a llibreria gràfica.
- Visual Studio 2026 [26] per programació i compilació de codi C++.

- Blender 4.5.1 [27] per la modelització d'objectes tridimensionals.
- Notepad++ [28] per editar arxius.
- Github [29] com a eina de control de versions.
- Taskesy [30] i Taskesy 2 [31] per l'organització i gestió de tasques.
- Google Translate [32] i DeepL [33] com a eines per la traducció.
- SoftCatalà [34] i Reverso [35] com a eines per la correcció de text.
- FREE IEEE Citation Generator [36] per tal de generar de forma més o menys automàtica les referències en format IEEE.
- Canva [37] per a crear recursos visuals i infografia.
- Visual Studio Code [38] amb l'extensió LaTeX Workshop, Strawberry Perl [39] i MiKTeX [40] com a framework [41] per treballar amb documents .latex.

## 2 Marc teòric

OpenGL [4, 6] és l'opció perfecta i recomanada per iniciar-se en el món dels gràfics per ordinador [5], ja que és una API que treballa a baix nivell i ofereix les possibilitats necessàries per a la majoria de projectes. És ideal tenir un entorn amb OpenGL, perquè permet que els estudiants puguin treballar d'una manera més còmoda i aprendre les bases dels gràfics per ordinador. No obstant això, avui en dia és difícil trobar aquesta API en motors gràfics més moderns i l'alternativa és Vulkan per la seva capacitat d'oferir més llibertat i control sobre les targetes gràfiques [16]. Vulkan [10, 11] és el successor directe d'OpenGL, doncs totes dues són APIs gràfiques obertes creades per Khronos Group [8, 9]. OpenGL va aparèixer a 1992 i l'última versió estable va sortir en 2017 (amb la versió 4.6) [6]. Vulkan s'origina l'any 2016 i segueix en desenvolupament continu [11] per aquest mateix consorci oficial. Vulkan es pot utilitzar per fer tota mena d'aplicacions que vulguin aprofitar la capacitat de càlcul de les targetes gràfiques [16], tant de manera gràfica com per la part de computació. El gran avantatge respecte al seu competidor principal (DirectX 12 [42]), és que és multiplataforma [43]. Cada vegada sorgeixen més projectes on es necessita un enfocament en temps-real i, per tant, l'optimització passa a ser un factor molt important. Un exemple pot ser el programari "Yasara" [44, 45] (estructures amb milions o milers de milions d'àtoms). Vulkan segueix en evolució contínua i s'utilitza en projectes de renom, com pot ser el motor gràfic "Godot Engine" [46] o l'ús en videojocs com "Baldur's Gate III" [47, 48].

D'altra banda, l'algoritme "ray marching" es basa en la iteració de raigs que surten de cada píxel i van avançant fins a col·lidir amb les dades en escena [17]. A la figura 1 (1) podem veure un exemple del seu funcionament amb l'ús d'una funció de distància amb signe

(SDF) [17, 49]. Aquesta funció permet que quan cada píxel emeti un raig que va avançant en una direcció, aquest depengui de la distància mínima fins a l'objecte més proper. D'aquesta manera, es pot garantir una col·lisió en cas de trobar qualsevol obstrucció al seu pas i és un enfocament més eficient que el de tenir un pas fix definit. Això és exactament el que veiem a la figura 1 (1), on un raig (el seu origen és cada píxel en pantalla) va avançant un pas mínim calculat a partir de la distància fins a l'objecte més proper en la posició actual del raig. Veiem que el raig, en primera instància, s'apropa a una superfície, fet que augmenta el nombre de passos, ja que són més petits. Com no col·lideix amb la superfície, continua avançant fins que trobem una superfície amb la qual sí que col·lidim. Podem esgotar el nombre de passos i perdre l'objecte o, simplement, no trobar-lo en escena. Si no trobem res, no fa falta que pintem res en aquell fragment (no visualitzem res per pantalla o tractem aquest cas de manera adequada). No hem de confondre aquest algoritme amb l'algoritme "ray tracing" [50], ja que aquest últim calcula les interseccions amb els objectes en escena i acostuma a ser recursiu (no iteratiu).

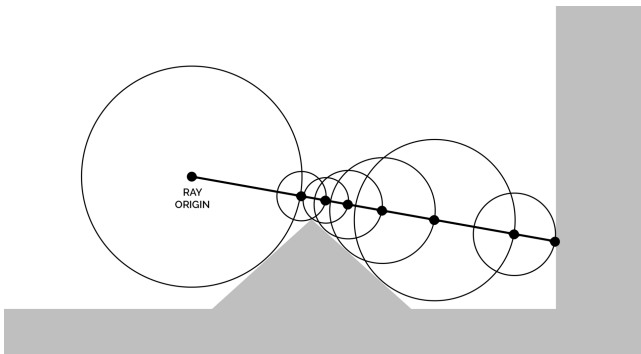


Fig. 1: Sphere Tracing (Ray marching utilitzant funcions de distància amb signe (SDF)). [17]

Podem veure un possible codi de l'algoritme "ray marching", on necessitem l'origen del raig i la seva direcció. Fem un bucle amb un màxim de passos (els raigs han de finalitzar la seva cerca si no troben res en escena) i anem calculant la posició actual del raig amb la distància fins a la superfície més propera (funció SDF) fins que acabem el nombre de passos permès o estiguem tan a prop de la superfície que determinem que hem col·lidit i ja podem retornar la posició de col·lisió.

Listing 1: Possible codi de Ray Marching

```
float rayMarch(vec3 rayOrigin,
vec3 rayDirection)
{
    float distOrigin = 0., distScene;
    vec3 pos;

    for (int i = 0; i < MAX_STEPS; i++)
    {
        pos = rayOrigin +
            rayDirection*distOrigin;
        distScene = SDF(pos);
        distOrigin += distScene;
    }
}
```

```
if (distOrigin > MAX_DIST ||
    distScene < SURF_DIST) break;
}
return distOrigin;
}
```

L'exploració de l'algoritme "ray marching" em permet aprendre sobre aquesta mena de funcions, que ajuden a renderitzar [51] dades de manera volumètrica gràcies a les col·lisions d'aquests raigs a l'escena. Els camps d'aplicació són amplis, ja sigui d'una manera més artística com a les figures 2 i 4 (2, 4) [52, 53, 54] amb l'elaboració de shaders [55], fins a aplicacions mèdiques com a la figura 3 (3) [21] per visualitzar dades tomogràfiques [56] o simulacions utilitzant volums com núvols o fluids [20, 57]. A més, és útil per fer ús d'operacions matemàtiques com les unions o interseccions, visibles a la figura 4 (4) [58].

A la figura 2 (2) podem veure com es poden fer animacions amb l'algoritme ray marching i del potencial creatiu que té per crear escenes de manera matemàtica.

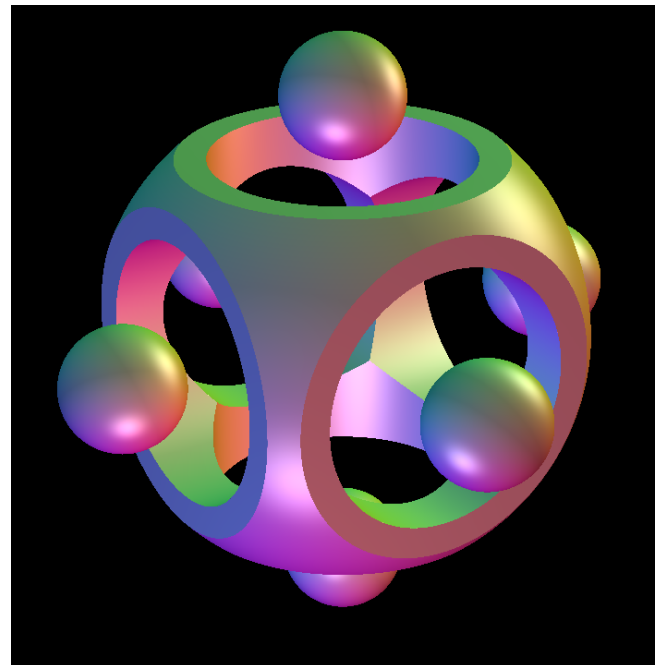


Fig. 2: Una animació feta amb Ray Marching. [59, 60]

A la figura 3 (3) podem veure la visualització de dades mèdiques gràcies a l'algoritme "ray marching", ja que podem representar un conjunt de dades mèdiques i el seu volum gràcies a raigs que puguin travessar les dades i mostrar la seva densitat.

A la figura 4 (4) podem apreciar la característica propietat de "ray marching" per representar les dades de manera matemàtica i poder fer ús d'operadors com les unions o interseccions algebraiques.

Cal esmentar un projecte en el qual s'està fent un motor gràfic amb SDFs com els que hem pogut visualitzar a la figura 1 (1) [64] i es pot veure, per tant, el potencial que té l'algoritme "ray marching". S'utilitza una pipeline gràfica [65] amb pipeline de computació [13], amb l'ús de compute shaders [66] per tal d'operar amb les dades abans de visualitzar-les. Els compute shaders serveixen per aprofitar els recursos de càlcul



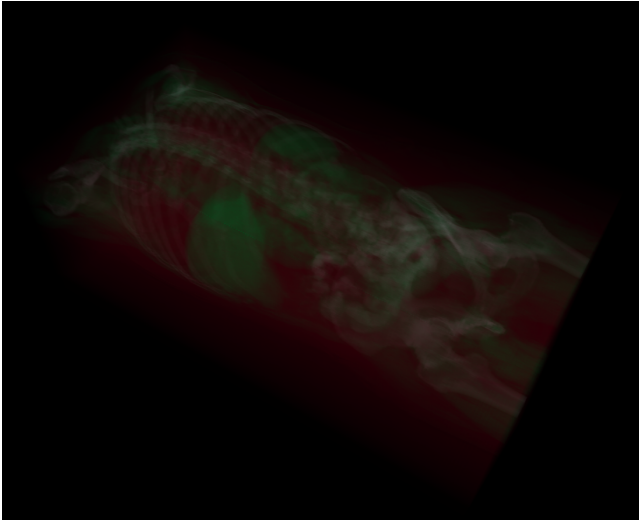


Fig. 3: Visualització de dades mèdiques amb Ray Marching. [61, 62]

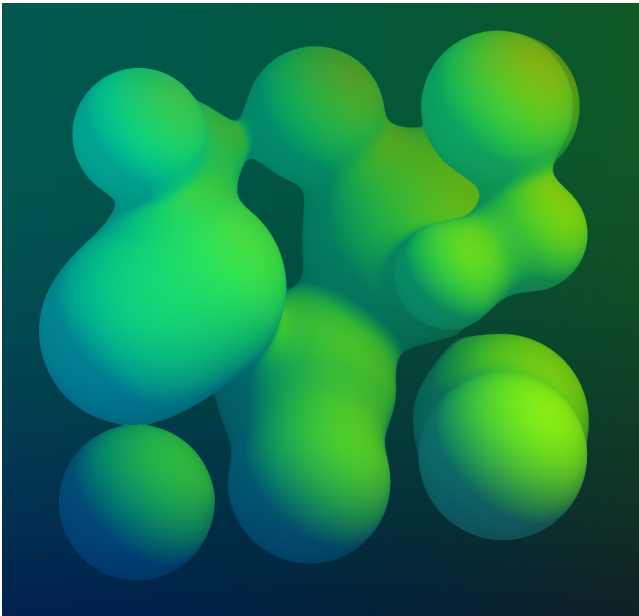


Fig. 4: Animació de "Lava" amb Ray Marching. [63]

que ens ofereix la targeta gràfica per tal de fer operacions que no estan necessàriament lligades a aplicacions gràfiques, és a dir, que no tenen relació amb la pipeline gràfica. Això ho podem veure a la figura 5 (5). En aquesta darrera figura, tenim a la part esquerra la gestió de la part gràfica i, a la dreta, la part de computació. Podem observar que no hi ha dependències (o que són opcionals) entre aquestes fases o pipelines.

L'entorn amb Vulkan serà, per tant, una eina ideal per poder treballar les dades amb la part de computació i fer les modificacions adients en la part gràfica per tal de visualitzar aquestes dades prèviament calculades. El llibre "The Godot Shaders Bible" [68] és interessant per a la formació en la programació de shaders [55], doncs l'algoritme de "ray marching" s'acostuma a implementar d'aquesta manera. Aquest llibre s'actualitza periòdicament, el que permet utilitzar tècniques actuals al motor "Godot Engine" [46] i treballar de manera pràctica i teòrica. De fet, en aquest entorn

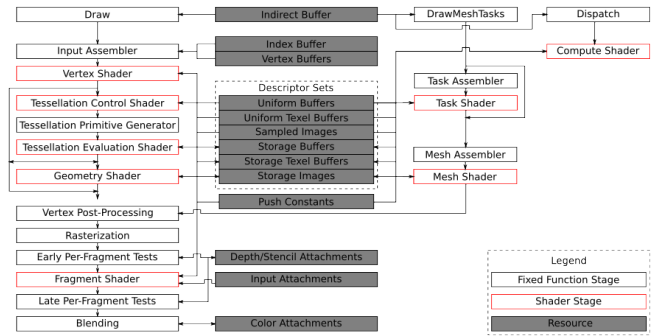


Fig. 5: Diagrama de blocs que representa les relacions entre les pipelines a Vulkan. [67]

he pogut treballar en una implementació de "ray marching" amb un dels exemples del llibre per a comprendre la visualització d'objectes amb volum. En aquest cas els raigs avaluen la densitat respecte a una textura tridimensional [69] d'un núvol a mesura que avancen amb un pas fix. Es pot veure el resultat a la figura 6 (6).

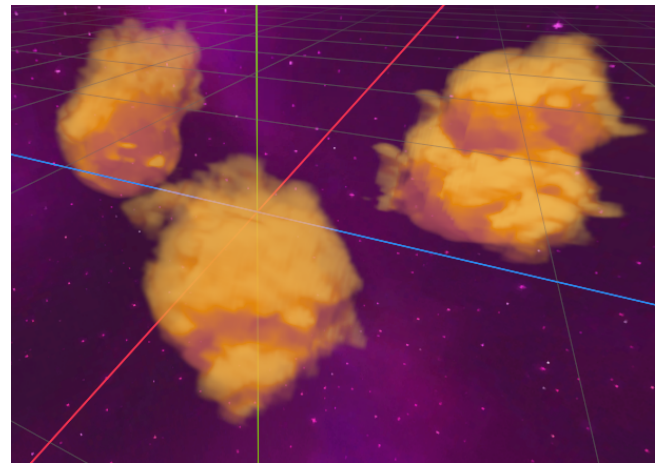


Fig. 6: Ray Marching and 3D Textures. [68]

### 3 Desenvolupament

Gràcies a la realització dels cursos de formació en Vulkan [70, 71], he aconseguit muntar un entorn gràfic amb part gràfica i de computació, preparat per fer aplicacions gràfiques. A més a més, he afegit la llibreria "Dear ImGui" [72] per a poder tenir una interfície gràfica [73]. Podem veure a la figura 7 (7) les relacions entre les classes (VulkanRenderer, VulkanValidation, MeshModel, Mesh, Utilities, Random) i els mòduls (Vulkan, GLFW, GLM, Assimp, imgui, stb\_image) externs utilitzats.

En aquest diagrama es veu el protagonisme de la classe VulkanRenderer en fer ús de la resta de sistemes. Encara que aquesta classe és molt extensa, s'ha creat d'aquesta manera per tal de simplificar la implementació d'aquest entorn gràfic simple. En tenir molta diversitat de funcions en aquesta classe, ens centrem en els procediments i no desviem l'atenció cap a la definició i divisió dels diversos submòduls que trobem, doncs un entorn gràfic està compost per diverses fases i procediments.

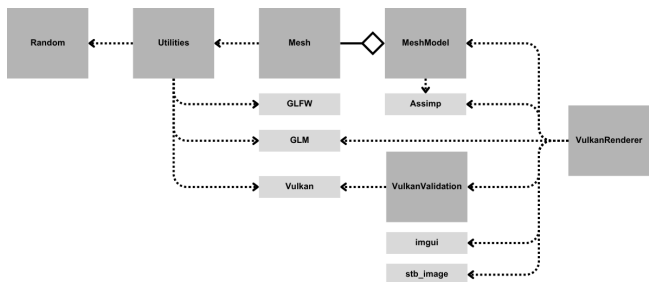


Fig. 7: Diagrama de mòduls.

En primera instància, tenim la classe "VulkanRendrer". Aquesta classe és la més important, donat que és on es defineix l'entorn gràfic de manera directa.

Després, tenim la classe "VulkanValidation", que és la que s'encarrega del tractament dels possibles errors que puguem obtenir amb Vulkan. Aquesta classe és crucial, donat que sense aquesta validació, no podríem detectar els errors a la implementació de Vulkan.

En tercera instància, tenim la classe "Utilities". És la que conté tota mena de variables i funcions auxiliars que serveixen de suport per a l'entorn gràfic.

La classe "Mesh" és la que defineix de manera mínima els objectes 3D. S'utilitza per tractar la càrrega d'objectes 3D a l'escena.

A més, la classe "MeshModel" és la que defineix els objectes 3D mitjançant la classe anterior ("Mesh"). La raó és que un mateix model 3D pot estar format de diversos objectes. D'aquesta manera, tractem la càrrega de models complexes en 3D.

En últim lloc, tenim una classe petita anomenada "Random". S'utilitza aquesta classe per generar nombres aleatoris, encara que només s'han fet servir nombres amb precisió "float".

A la figura 8 (8) podem observar les relacions dels shaders de l'entorn. S'utilitza la pipeline de computació pel càlcul intensiu de dades (com posicions de les partícules en l'exemple trivial) i es passen aquestes dades a la pipeline gràfica, on el primer vertex shader treballa amb els models i el fragment shader treballa sobre aquests models (com per exemple, per calcular textures). Després, tenim un altre vertex i fragment shader, on el segon vertex shader només fa un triangle que ocupa la pantalla (per dibuixar en la totalitat dels píxels en pantalla) i el segon fragment shader s'utilitza per poder aplicar processament al que rebem del primer fragment shader, com es pot apreciar al diagrama.

Aquest entorn ha sigut derivat en una aplicació anomenada "Explorador de Fractals Tridimensionals" per fer una sèrie de demostracions amb "ray marching" [17] (però és un projecte independent de l'entorn gràfic simple, és a dir, una nova aplicació). He fet una primera implementació de "ray marching" amb un exemple trivial [74] a un dels fragment shaders de l'entorn, ja que la implementació s'acostuma a fer d'aquesta manera. Una vegada s'ha vist la correctesa, s'ha implementat l'algoritme al compute shader, pel fet que puc dividir el treball entre els "Work Groups" [75] per tal d'utilitzar el paral·lelisme que ofereix la targeta gràfica. Podem observar a la figura 9 (9) el diagrama de relacions dels shaders de l'aplicació "Explorador de Fractals

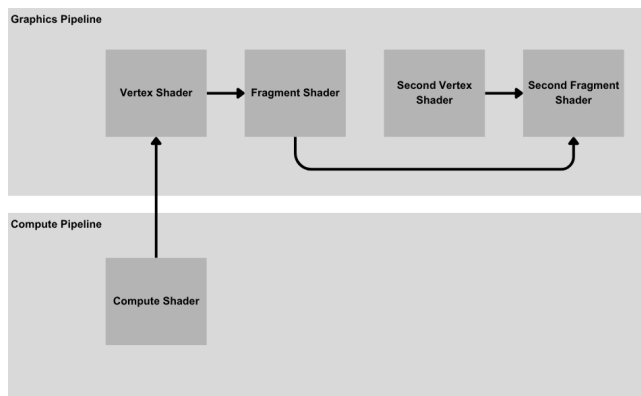


Fig. 8: Diagrama de relacions dels shaders de l'entorn.

Tridimensionals". Com a diferències principals amb l'entorn, s'han modificat les relacions entre la pipeline gràfica i la de computació. Podem observar que el processament de les dades arriba directament al primer fragment shader (no fem ús de models, encara que es podrien utilitzar), per tant, la principal diferència amb l'entorn és que no es fa ús de la càrrega de models 3D.

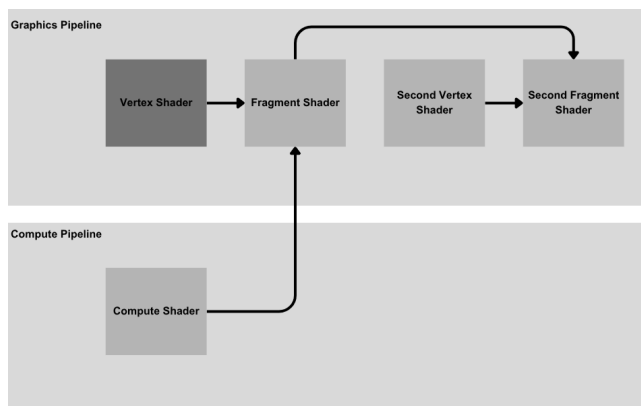


Fig. 9: Diagrama de relacions dels shaders de l'aplicació "Explorador de Fractals Tridimensionals".

## 4 Resultats

Podem veure a l'exemple de demostració trivial de l'entorn gràfic simple de la figura 10 (10) un exemple de partícules on s'utilitza el potencial de l'entorn (part gràfica i de computació).

Finalment, s'ha creat una aplicació per explorar fractals tridimensionals, amb un seguit de tests i demostracions incloent fractals propis. Gràcies a les referències utilitzades ([54, 77]), s'ha seguit el model del fractal "Menger Sponge" [78, 79] com a base per crear la resta de tests (en total, 28 tests). S'han creat les funcions de distància personalitzades per obtenir els meus propis fractals.

S'ha pogut identificar els fractals donat que tenen la propietat de detall infinit [18], ja que en algunes proves fetes no s'apreciava aquesta característica i s'han hagut de descartar. Podem veure aquesta propietat al camp "Fractal Iterations" de l'aplicació. A les següents figures 11, 12, 13 i 14 (11, 12, 13 i 14) podem

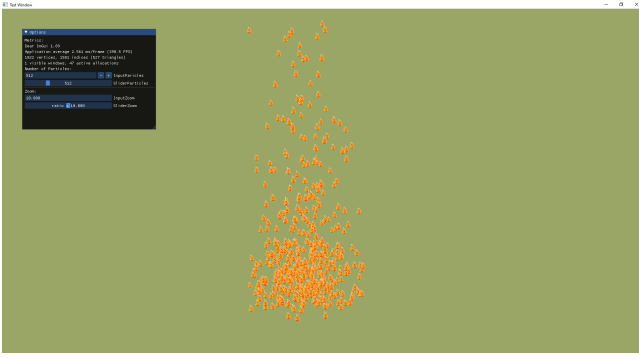


Fig. 10: Demostració predeterminada i trivial amb partícules de l'entorn de Vulkan. [76]

veure aquesta propietat de manera més clara en un dels fractals creats:

De manera inicial, podem veure la forma base del patró (figura 11 (11)). Després, podem veure com el patró es divideix de manera recursiva amb dos iteracions (figura 12 (12)). Amb tres iteracions, podem veure com el patró es divideix de manera recursiva una segona vegada en la forma inicial (figura 13 (13)). Amb quatre iteracions (figura 14 (14)) (o més), podem observar que els patrons cada vegada són més petits i és difícil identificar-los, com hem pogut veure en aquest procés de subdivisió.

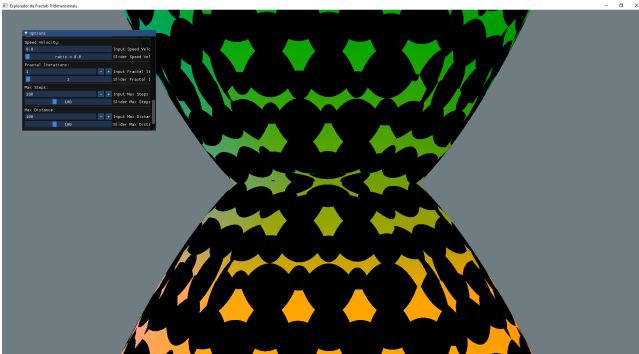


Fig. 11: Test 18: Original SDF 12, Fractal Iterations = 1. [80]

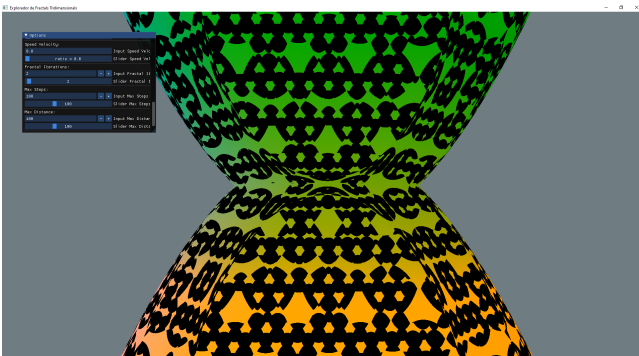


Fig. 12: Test 18: Original SDF 12, Fractal Iterations = 2. [80]

Podem veure, per tant, que s'han creat fractals tridimensionals amb formes i patrons infinits mai vistos anteriorment.

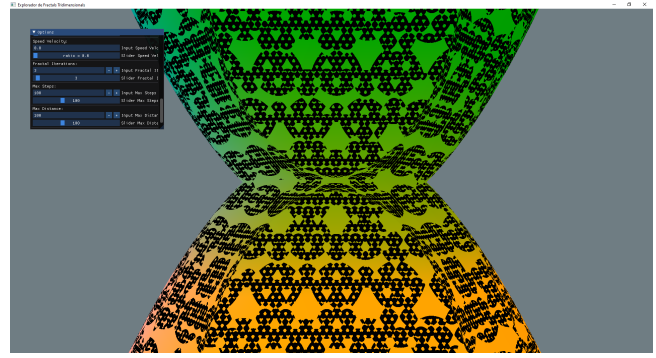


Fig. 13: Test 18: Original SDF 12, Fractal Iterations = 3. [80]

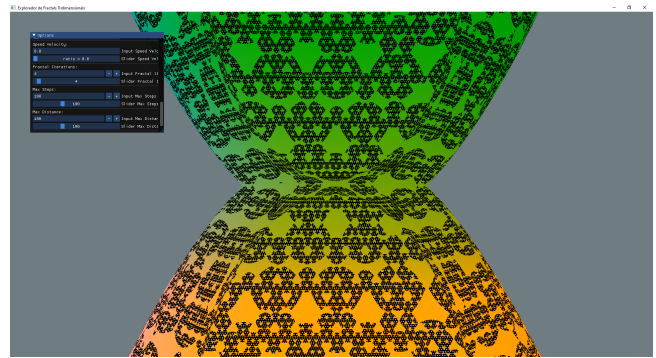


Fig. 14: Test 18: Original SDF 12, Fractal Iterations = 4. [80]

La SDF d'aquest fractal és la següent. S'utilitza una primitiva de 'creus' per Connor A. Haskins (sdCross() [79]) i una primitiva d'esfera creada per Iñigo Quilez (sdSphere() [77]):

Listing 2: Test 18: Original SDF - SDF

```
float test18SDF(vec3 p) {
    p = abs(p);
    return abs(sqrt(sdCross(
        vec3(sdSphere(p, p.y)))));
}
```

Aquestes primitives s'utilitzen per crear formes geomètriques definides com a funcions de distància amb signe (SDF). Per tal de crear els meus fractals tridimensionals (A.1) s'han hagut de definir les seves SDFs, que funcionen dins del sistema referenciat de l'exemple "Menger Sponge". Per tal d'aconseguir-ho, s'han utilitzat operacions matemàtiques (com la funció abs(), que permet crear certa simetria pel funcionament d'aquest operador matemàtic) i l'ús d'aquesta mena de primitives gràfiques.

S'ha creat un sistema de coloració per tal de donar color als fractals depenent de l'orientació de la càmera, com es pot observar en el codi següent.

Listing 3: Coloració fractal

```
vec3 rayMarching( in vec3 ro, in vec3 rd )
{
    ...
    vec3 color;

    if (hit)
```

```

{
    color = vec3(1.0 -
        (float(iterations)
        / float(ubo.MAX_STEPS)));
    color += (((pos.yxz * 0.5)
        - vec3(0.45)) * 0.8) *
        ubo.CAMERA_ORIENTATION.xyz
        - vec3(0.45) * 0.8;
    color.x += clamp
        (ubo.CAMERA_ORIENTATION.x
        * pos.y * 0.4, 0.2, 0.6);
}
else
    color = vec3(0.434, 0.491,
        0.515);

return color;
}

```

Com s'ha comentat, s'ha utilitzat la base del sistema fractal "Menger Sponge" per tal d'incloure les SDFs per cada test fractal, com es pot observar en el codi següent.

Listing 4: Estructura de "Menger Sponge" utilitzada [79]

```

float sdMengerSpongeSystem(vec3 rayPos) {
    const float cubeWidth = 1.0;
    const float oneThird = 1.0 / 3.0;
    float spongeCube = testSDF(rayPos);
    float mengerSpongeDist = spongeCube;

    float scale = 3.0;
    for(int i = 0; i < ubo.FRACT_ITER; ++i) {
        // #1 determine repeated
        box width
        float boxedWidth =
            cubeWidth / scale;

        float translation =
            -boxedWidth / 2.0;
        vec3 ray = rayPos
            - translation;
        vec3 repeatedPos =
            mod(ray, boxedWidth);
        repeatedPos +=
            translation;

        // #2 scale coordinate
        systems from
        // [-1/scale, 1/scale) ->
        to [-1.0, 1.0)
        repeatedPos *= scale;

        float crossesDist = testSDF(
            repeatedPos / oneThird)
            * oneThird;

        // #3 Acquire actual
        distance by un-stretching
        crossesDist /= scale;

        mengerSpongeDist = max(
            mengerSpongeDist,
            -crossesDist);

        scale *= 3.0;
    }
}

```

```

}
return mengerSpongeDist;
}

```

## 5 Conclusions

S'han aconseguit tots els objectius proposats, per una part, s'ha creat un entorn gràfic simple fet amb Vulkan, amb la possibilitat de carregar i visualitzar objectes tridimensionals, fer ús de les pipelines gràfiques, de computació i d'una interfície gràfica. D'altra banda, s'ha creat una aplicació gràfica independent que sorgeix de la modificació de l'entorn gràfic amb Vulkan, anomenada "Explorador de Fractals Tridimensionals". En aquesta aplicació, he pogut desenvolupar i crear diferents fractals tridimensionals a partir de l'estructura pròpia del fractal 3D "Menger Sponge". S'han afegit paràmetres modificables des de l'interfície gràfica de l'aplicació per tal de permetre i ajudar amb l'estudi tant de l'algoritme "ray marching", com del funcionament dels fractals de manera pràctica i en temps real.

Com a millores a futur del treball, cal destacar que a partir d'aquest projecte, la idea principal seria la divisió de l'entorn gràfic simple desenvolupat en dos projectes nous.

- Un entorn gràfic amb les mínimes funcionalitats per a poder usar-lo com a base per crear aplicacions gràfiques potents i específiques (un exemple podria ser, per exemple, la creació de l'aplicació "Explorador de Fractals Tridimensionals", ja que és un producte derivat de l'entorn). Es podria prioritzar l'optimització sense tenir en compte la compatibilitat amb màquines més antigues, perquè per exemple, en el meu cas, la vaig fer compatible amb una versió de Vulkan anterior per a donar suport a més ordinadors.
- Crear un entorn gràfic més desenvolupat per poder aprendre sobre algorismes i conceptes bàsics i avançats. Aquest projecte sí que estaria pensat per passar les funcionalitats de l'entorn gràfic en OpenGL de l'assignatura VGI a aquesta base de Vulkan. Aquest entorn gràfic sí que estaria pensat per treballar amb compatibilitat i poder permetre's tenir menys optimitzacions per tal de propiciar més el coneixement sobre els conceptes gràfics i arribar a més estudiants.
- Per la part de l'"Explorador de Fractals Tridimensionals" amb "ray marching", estaria bé deixar l'aplicació perquè altres estudiants puguin utilitzar-lo. Pot arribar a ser un incentiu per animar-los a aprendre sobre el potencial de l'algoritme i de les seves funcionalitats, donat que és àmpliament utilitzat en diferents àmbits (mèdics, artístics, etc) per la visualització de volums.

## Ús de la IA Generativa

La IA generativa va ser utilitzada a l'inici del projecte per tal de fer recerca de documents (papers) científics, però al final no em van resultar útils i no els vaig incloure. Cercadors com Google m'han ofert un resum fet amb IA del que he buscat, però m'he assegurat de contrastar la informació i de no incloure res directament, de fet, he intentat evitar llegir aquesta classe de resums. Algunes de les eines de traducció i correcció de text utilitzades poden fer ús d'aquesta tecnologia.

## Agraïments

No podria haver fet aquest treball si no hagués cursat l'assignatura Visualització Gràfica Interactiva de tercer d'enginyeria informàtica a la Universitat Autònoma de Barcelona. Aquest treball m'ha suposat un gran repte, però he treballat fent el que més m'agrada i he gaudit cada minut treballat.

Aquest treball no podria haver estat possible si no hagués sigut pel meu tutor i professor Enric Martí Gòdia, a qui li he d'agrair especialment pel naixement d'aquest treball.

Cal mencionar la col·laboració del meu company Isaac Sánchez Amat per utilitzar aquest treball com a base per a desenvolupar el seu propi treball de fi de grau "Simulació de cossos celestes en Vulkan".

Finalment, dono gràcies a Jordi Pons Aroztegui per oferir uns estudis d'alta qualitat i d'intentar millorar-los dia a dia.

## Referències

- [1] E. Martí. "AGD :: Aplicatiu de Guies Docents v2.1," Uab.cat, 2025. <https://apps.uab.cat/guies/public/portal/html/2025/assignatura/102753/ca> (data últim accés: març 2026).
- [2] Wikipedia Contributors, "C++," Wikipedia, Mar. 25, 2019. <https://en.wikipedia.org/wiki/C%2B%2B> (data últim accés: març 2026).
- [3] Wikipedia Contributors, "API," Wikipedia, Jun. 30, 2023. <https://es.wikipedia.org/wiki/API> (data últim accés: març 2026).
- [4] K. Group, "OpenGL - The Industry Standard for High Performance Graphics," Opengl.org, 2019. <https://www.opengl.org/> (data últim accés: març 2026).
- [5] Wikipedia Contributors, "Computació gràfica," Wikipedia.org, Jul. 16, 2010. [https://ca.wikipedia.org/wiki/Computaci%C3%B3\\_gr%C3%A0fica](https://ca.wikipedia.org/wiki/Computaci%C3%B3_gr%C3%A0fica) (data últim accés: març 2026).
- [6] Wikipedia Contributors, "OpenGL," Wikipedia, Jan. 20, 2020. <https://en.wikipedia.org/wiki/OpenGL> (data últim accés: març 2026).
- [7] Wikipedia Contributors, "Graphics card," Wikipedia, Sep. 11, 2021. [https://en.wikipedia.org/wiki/Graphics\\_card](https://en.wikipedia.org/wiki/Graphics_card) (data últim accés: març 2026).
- [8] "The Khronos Group," The Khronos Group, Feb. 14, 2023. <https://www.khronos.org/> (data últim accés: març 2026).
- [9] Wikipedia Contributors, "Khronos Group," Wikipedia, Dec. 30, 2024. [https://en.wikipedia.org/wiki/Khronos\\_Group](https://en.wikipedia.org/wiki/Khronos_Group) (data últim accés: març 2026).
- [10] Vulkan, "Home | Vulkan | Cross platform 3D Graphics," www.vulkan.org, May 10, 2021. <https://www.vulkan.org/> (data últim accés: març 2026).
- [11] Wikipedia Contributors, "Vulkan," Wikipedia, Feb. 24, 2022. <https://en.wikipedia.org/wiki/Vulkan> (data últim accés: març 2026).
- [12] Vulkan, "Learn | Vulkan | Cross platform 3D Graphics," Vulkan.org, Feb. 23, 2022. <https://vulkan.org/learn> (data últim accés: març 2026).
- [13] Wikipedia Contributors, "Pipeline (computing)," Wikipedia, May 16, 2019. [https://en.wikipedia.org/wiki/Pipeline\\_\(computing\)](https://en.wikipedia.org/wiki/Pipeline_(computing)) (data últim accés: març 2026).
- [14] S. Willems, "How to Vulkan in 2026 - How to Vulkan," Howtovulkan.com, 2026. <https://www.howtovulkan.com/> (data últim accés: març 2026).
- [15] Vulkan, "Vulkanised 2026 | Vulkan | Cross platform 3D Graphics," Vulkan.org, Feb. 13, 2026. <https://vulkan.org/events/vulkanised-2026> (data últim accés: març 2026).
- [16] "Introduction - Vulkan Tutorial," vulkan-tutorial.com. <https://vulkan-tutorial.com/> (data últim accés: març 2026).
- [17] Wikipedia Contributors, "Ray marching," Wikipedia, Nov. 14, 2022. [https://en.wikipedia.org/wiki/Ray\\_marching](https://en.wikipedia.org/wiki/Ray_marching) (data últim accés: març 2026).
- [18] Wikipedia Contributors, "Fractal," Wikipedia, Jul. 23, 2019. <https://en.wikipedia.org/wiki/Fractal> (data últim accés: març 2026).
- [19] CodeParade, "How to Make 3D Fractals," YouTube, May 19, 2018. <https://youtu.be/svLzmFuSBhk?si=pJdx5zy9OzpHD7Bd> (data últim accés: març 2026).
- [20] Unity, "Volumetrics: Introduction to ray marching | Tutorial," YouTube, Aug. 22, 2023. <https://www.youtube.com/watch?v=hXYOlXVRRl8> (data últim accés: març 2026).
- [21] Wikipedia Contributors, "Volume rendering," Wikipedia, Sep. 25, 2022. [https://en.wikipedia.org/wiki/Volume\\_rendering](https://en.wikipedia.org/wiki/Volume_rendering) (data últim accés: març 2026).
- [22] Wikipedia Contributors, "Desarrollo ágil del software," Wikipedia.org, Jun. 14, 2004. [https://es.wikipedia.org/wiki/Desarrollo\\_%C3%A1gil\\_de\\_software](https://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software) (data últim accés: març 2026).

- [23] Wikipedia Contributors, “Kanban (development),” Wikipedia, Mar. 22, 2019. [https://en.wikipedia.org/wiki/Kanban\\_\(development\)](https://en.wikipedia.org/wiki/Kanban_(development)) (data últim accés: març 2026).
- [24] GeeksforGeeks, “Kanban Agile Methodology,” GeeksforGeeks, Mar. 24, 2022. <https://www.geeksforgeeks.org/software-engineering/kanban-agile-methodology/> (data últim accés: març 2026).
- [25] “LunarXchange,” Lunarg.com, 2026. <https://vulkan.lunarg.com/home/welcome> (data últim accés: març 2026).
- [26] “IDE de Visual Studio, editor de código, Azure DevOps y App Center - Visual Studio,” visualstudio.microsoft.com. <https://visualstudio.microsoft.com/es/> (data últim accés: març 2026).
- [27] Blender Foundation, “Home of the Blender Project - Free and Open 3D Creation Software,” blender.org, 2019. <https://www.blender.org/> (data últim accés: març 2026).
- [28] D. Ho, “Notepad++ Home,” Notepad-plus-plus.org, 2019. <https://notepad-plus-plus.org/> (data últim accés: març 2026).
- [29] “GitHub: Let’s build from here,” GitHub, 2024. <https://github.com/home> (data últim accés: març 2026).
- [30] R. Velázquez “Taskesy,” itch.io, 2026. <https://barometz10.itch.io/taskesy> (data últim accés: març 2026).
- [31] R. Velázquez “Taskesy 2,” itch.io, 2026. <https://barometz10.itch.io/taskesy-2> (data últim accés: juny 2026).
- [32] Google, “Google Translate,” <https://translate.google.es/>(dataúltimaccés:juny2026). (data últim accés: juny 2026).
- [33] J. Kutylowski, “DeepL Translate: The world’s most accurate translator,” www.deepl.com. <https://www.deepl.com> (data últim accés: juny 2026).
- [34] Softcatalà, “Softcatalà | Informàtica i programari en català,” Softcatalà. <https://www.softcatala.org> (data últim accés: juny 2026).
- [35] Reverso, “Reverso | Free translation, dictionary,” www.reverso.net. <https://www.reverso.net> (data últim accés: juny 2026).
- [36] D. Elias, “FREE IEEE Citation Generator [Updated for 2020],” MyBib. <https://www.mybib.com/tools/ieee-citation-generator> (data últim accés: juny 2026).
- [37] Canva, “Canva,” Canva, 2025. <https://www.canva.com/>
- [38] Microsoft, “Visual Studio Code,” Visualstudio.com, 2025. <https://code.visualstudio.com/> (data últim accés: juny 2026).
- [39] Perl.org, “Strawberry Perl for Windows,” strawberryperl.com. <https://strawberryperl.com/> (data últim accés: juny 2026).
- [40] C. Schenk, “MiKTeX,” miktex.org. <https://miktex.org/> (data últim accés: juny 2026).
- [41] Ayudantías, “Cómo instalar LaTeX en VSCode,” YouTube, Aug. 26, 2022. <https://www.youtube.com/watch?v=9w7eb56bF7Y> (data últim accés: juny 2026).
- [42] “Tecnologías de DirectX 12 Ultimate | NVIDIA,” Nvidia.com, 2026. <https://www.nvidia.com/es-es/geforce/technologies/dx12/> (data últim accés: març 2026).
- [43] Wikipedia Contributors, “Multiplataforma,” Wikipedia, Feb. 02, 2021. <https://es.wikipedia.org/wiki/Multiplataforma> (data últim accés: març 2026).
- [44] “YASARA - Yet Another Scientific Artificial Reality Application,” Yasara.org, 2023. <https://www.yasara.org/index.html> (data últim accés: març 2026).
- [45] YASARA Biosciences GmbH, “Building and visualizing gigastructures with YASARA,” Yasara.org, 2021. <https://www.yasara.org/gigastructures.htm> (data últim accés: març 2026).
- [46] Godot Engine, “Godot Engine - Motor de videojuegos 2D y 3D gratuito y de código abierto,” Godot Engine, 2025. <https://godotengine.org/es/> (data últim accés: març 2026).
- [47] Larian Studios, “Baldur’s Gate 3,” baldursgate3.game, 2023. <https://baldursgate3.game/> (data últim accés: març 2026).
- [48] “Support - Larian Studios | What is Vulkan? The difference between Vulkan and DX11,” Larian.com, 2020. [https://larian.com/support/faqs/what-is-vulkan-the-difference-between-vulkan-and-dx11\\_48](https://larian.com/support/faqs/what-is-vulkan-the-difference-between-vulkan-and-dx11_48) (data últim accés: març 2026).
- [49] Wikipedia Contributors, “Signed distance function,” Wikipedia, Dec. 14, 2023. [https://en.wikipedia.org/wiki/Signed\\_distance\\_function](https://en.wikipedia.org/wiki/Signed_distance_function) (data últim accés: març 2026).
- [50] Wikipedia Contributors, “algoritmo para síntesis de imágenes,” Wikipedia.org, Jan. 05, 2005. [https://es.wikipedia.org/wiki/Trazado\\_de\\_rayos](https://es.wikipedia.org/wiki/Trazado_de_rayos) (data últim accés: juny 2026).
- [51] Wikipedia Contributors, “proceso de generación de imágenes por computadora,” Wikipedia.org, Feb. 12, 2002. <https://es.wikipedia.org/wiki/Renderizaci%C3%B3n> (data últim accés: març 2026).
- [52] “Shadertoy BETA,” www.shadertoy.com. <https://www.shadertoy.com/> (data últim accés: març 2026).

- [53] “Shadertoy BETA,” [www.shadertoy.com](http://www.shadertoy.com). <https://www.shadertoy.com/results?query=ray+marching> (data últim accés: març 2026).
- [54] I. Quilez, “Inigo Quilez,” [Iquilezles.org](http://iquilezles.org), 2022. <https://iquilezles.org/articles/raymarchingdf/> (data últim accés: març 2026).
- [55] Wikipedia Contributors, “Shader,” Wikipedia, Apr. 01, 2020. <https://en.wikipedia.org/wiki/Shader> (data últim accés: març 2026).
- [56] Wikipedia Contributors, “Tomografía,” Wikipedia.org, Apr. 07, 2006. <https://es.wikipedia.org/wiki/Tomograf%C3%ADa> (data últim accés: març 2026).
- [57] Kivoxo, “water - raymarching,” YouTube, Mar. 11, 2025. <https://www.youtube.com/watch?v=1AS9jSGf-yE> (data últim accés: març 2026).
- [58] The Art of Code, “RayMarching: Basic Operators,” YouTube, Feb. 08, 2019. <https://www.youtube.com/watch?v=AfKGMUDWfuE> (data últim accés: març 2026).
- [59] J. Wong, “Ray Marching: Part 6”, Shadertoy, Jul. 15, 2016. <https://www.shadertoy.com/view/4tcGDr> (data últim accés: abril 2026).
- [60] J. Wong, “Ray Marching and Signed Distance Functions”, [jamie-wong.com](http://jamie-wong.com), Jul. 15, 2016. <https://jamie-wong.com/2016/07/15/ray-marching-signed-distance-functions/> (data últim accés: abril 2026).
- [61] A. Pérez i S. Pineda, “Advanced Computer Graphics - Volume Rendering”, Github, Jul. 23, 2021. <https://github.com/abelperezd/Volume-Rendering-with-Ray-Marching> (data últim accés: abril 2026).
- [62] A. Pérez, “Volume Rendering with Ray Marching,” YouTube, Jul. 23, 2021. [https://www.youtube.com/watch?v=W\\_5r5V9TeX4](https://www.youtube.com/watch?v=W_5r5V9TeX4) (data últim accés: abril 2026).
- [63] edankwan, “CineShader Lava”, Shadertoy, Nov. 21, 2019. <https://www.shadertoy.com/view/3sySRK> (data últim accés: abril 2026).
- [64] Mike Turitzin, “I’m making a game engine based on dynamic signed distance fields (SDFs),” YouTube, Jan. 06, 2026. <https://www.youtube.com/watch?v=il-TXbn5iMA> (data últim accés: març 2026).
- [65] “Graphics pipeline,” Wikipedia, Jan. 02, 2020. [https://en.wikipedia.org/wiki/Graphics\\_pipeline](https://en.wikipedia.org/wiki/Graphics_pipeline) (data últim accés: març 2026).
- [66] Wikipedia Contributors, “Compute shader,” Wikipedia, Nov. 10, 2025. [https://en.wikipedia.org/?title=Compute\\_shader&redirect=no](https://en.wikipedia.org/?title=Compute_shader&redirect=no) (data últim accés: març 2026).
- [67] Vulkan, “Pipelines - Vulkan Documentation Project,” Vulkan.org, 2026. <https://docs.vulkan.org/spec/latest/chapters/pipelines.html> (data últim accés: abril 2026).
- [68] F. Espíndola, “The Godot Shaders Bible,” Jettelly | Resources for Game Devs, 2024. <https://jettelly.com/store/the-godot-shaders-bible> (data últim accés: abril 2026).
- [69] Wikipedia Contributors, “Textura (gráficos por computadora),” Wikipedia.org, Jan. 15, 2009. [https://es.wikipedia.org/wiki/Textura\\_\(gr%C3%A1ficos\\_por\\_computadora\)](https://es.wikipedia.org/wiki/Textura_(gr%C3%A1ficos_por_computadora)) (data últim accés: abril 2026).
- [70] B. Cook. “Learn the Vulkan API with C++,” Udemy, 2024. <https://www.udemy.com/course/learn-the-vulkan-api-with-cpp/> (data últim accés: març 2026).
- [71] M. Stofa. “GPU computing in Vulkan,” Udemy, 2024. <https://www.udemy.com/course/vulkan-gpu-computing/> (data últim accés: març 2026).
- [72] ocornut, “ocornut/imgui,” GitHub, Dec. 06, 2019. <https://github.com/ocornut/imgui> (data últim accés: març 2026).
- [73] Wikipedia Contributors, “Graphical user interface,” Wikipedia, Aug. 08, 2019. [https://en.wikipedia.org/wiki/Graphical\\_user\\_interface](https://en.wikipedia.org/wiki/Graphical_user_interface) (data últim accés: març 2026).
- [74] The Art of Code, “Ray Marching for Dummies!,” YouTube, Dec. 07, 2018. <https://www.youtube.com/watch?v=PGtv-dBi2wE> (data últim accés: abril 2026).
- [75] V. Blanco, “Compute Shaders,” Vulkan Guide. [https://vkguide.dev/docs/gpudriven/compute\\_shaders/](https://vkguide.dev/docs/gpudriven/compute_shaders/) (data últim accés: abril 2026).
- [76] R. Velázquez, “VGI-Entorn-Vulkan,” GitHub, 2026. <https://github.com/Barometz1o/VGI-Entorn-Vulkan> (data últim accés: juny 2026).
- [77] I. Quilez, “Inigo Quilez,” [iquilezles.org](http://iquilezles.org). <https://iquilezles.org/articles/distfunctions/> (data últim accés: juny 2026).
- [78] I. Quilez, “Inigo Quilez :: computer graphics, maths, shaders, fractals, demoscene,” [Iquilezles.org](http://iquilezles.org), 2026. <https://iquilezles.org/articles/menger/> (data últim accés: juny 2026).
- [79] C. A. Haskins, “Ray Marching: Menger Sponge Breakdown,” Substack.com, Mar. 28, 2024. <https://connorahaskins.substack.com/p/ray-marching-menger-sponge-breakdown> (data últim accés: juny 2026).
- [80] R. Velázquez, “Vulkan-Application,” GitHub, 2026. <https://github.com/Barometz1o/Vulkan-Application> (data últim accés: juny 2026).



## Apèndix

### A.1 Explorador de Fractals Tridimensionals

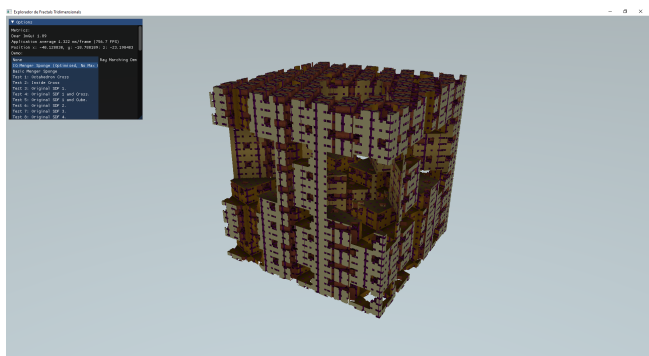


Fig. 15: Resultat obtingut de la prova fractal "IQ Menger Sponge (Optimised, No Max Distance)."

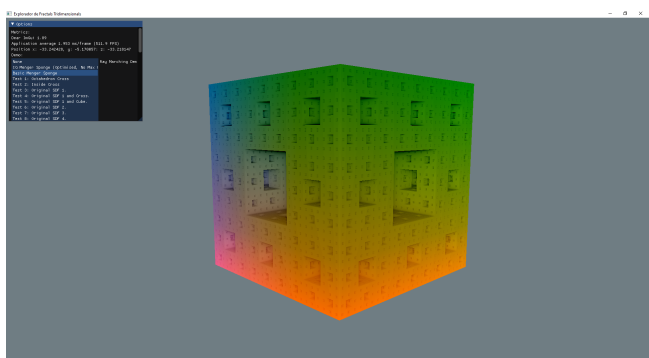


Fig. 16: Resultat obtingut de la prova fractal "Basic Menger Sponge."

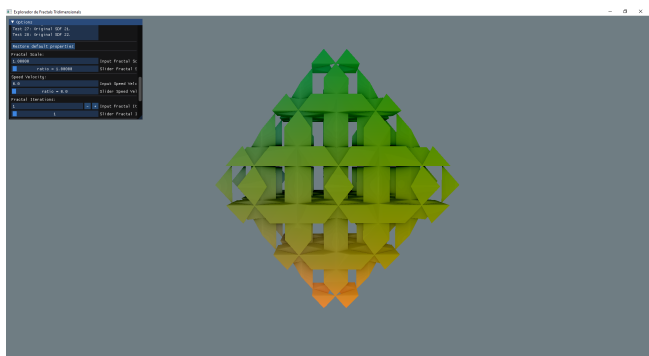


Fig. 17: Resultat obtingut de la prova fractal "Test 1: Octahedron Cross."

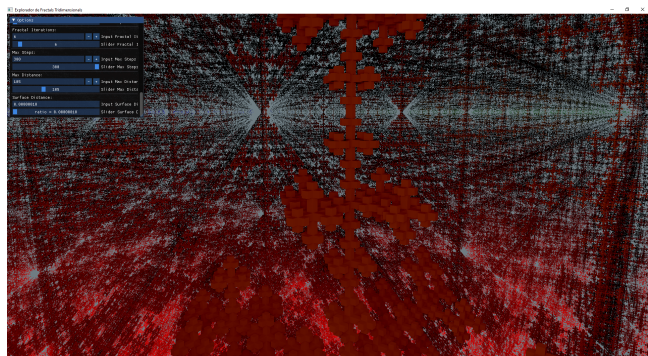


Fig. 18: Resultat obtingut de la prova fractal "Test 2: Inside Cross."

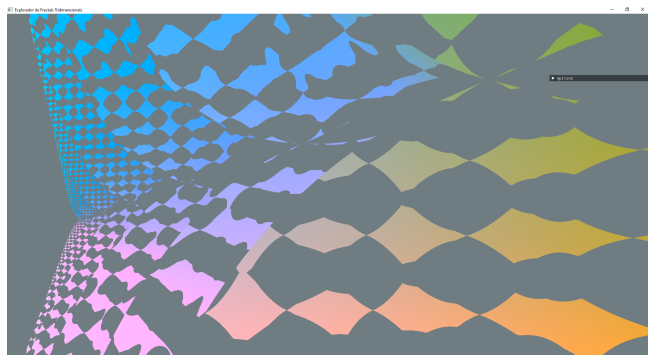


Fig. 19: Resultat obtingut de la prova fractal "Test 3: Original SDF 1."

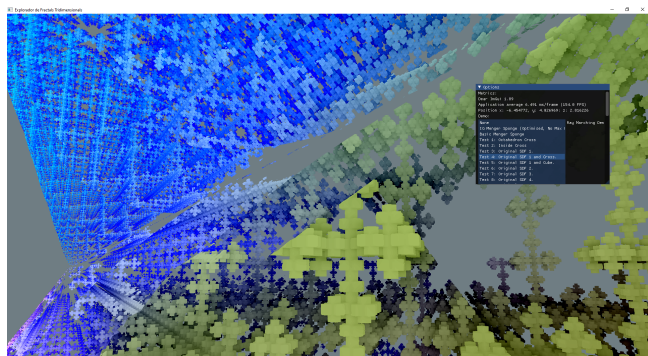


Fig. 20: Resultat obtingut de la prova fractal "Test 4: Original SDF 1 and Cross."

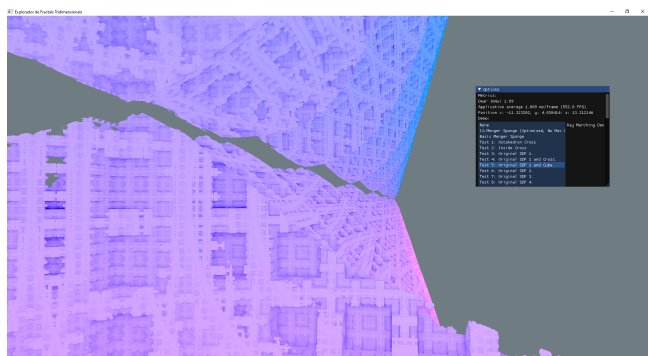


Fig. 21: Resultat obtingut de la prova fractal "Test 5: Original SDF 1 and Cube."



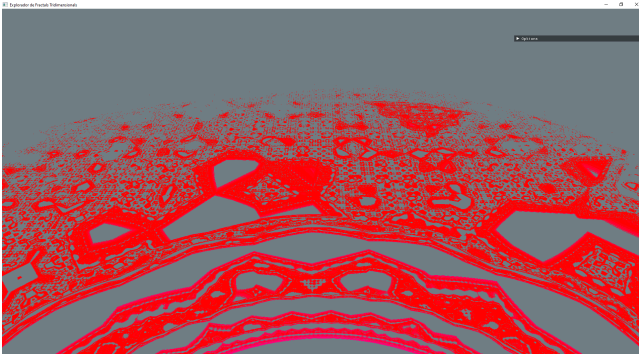


Fig. 22: Resultat obtingut de la prova fractal "Test 6: Original SDF 2."

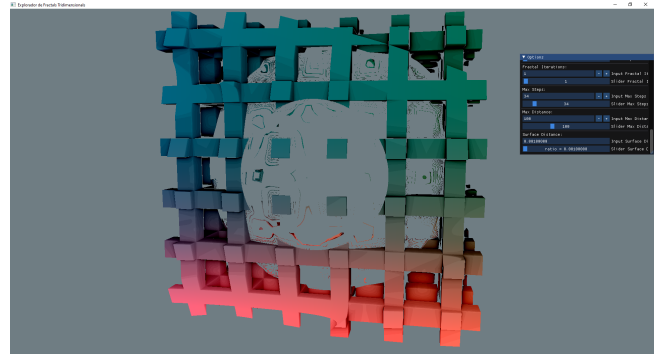


Fig. 26: Resultat obtingut de la prova fractal "Test 10: Original SDF 5 and Cross."

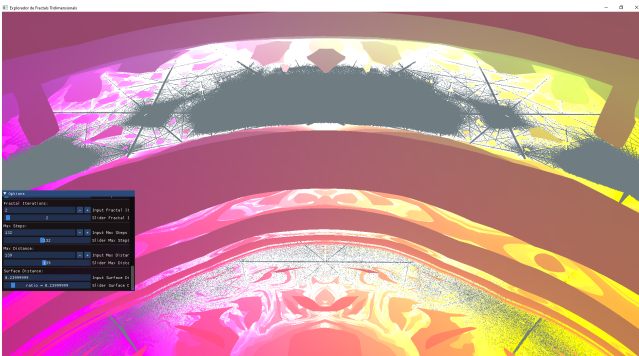


Fig. 23: Resultat obtingut de la prova fractal "Test 7: Original SDF 3."

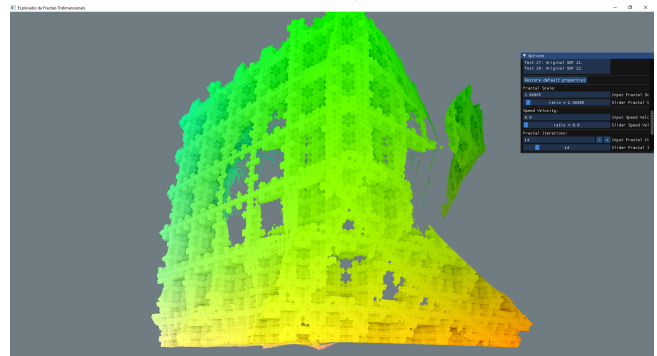


Fig. 27: Resultat obtingut de la prova fractal "Test 11: Original SDF 5 and Cube."

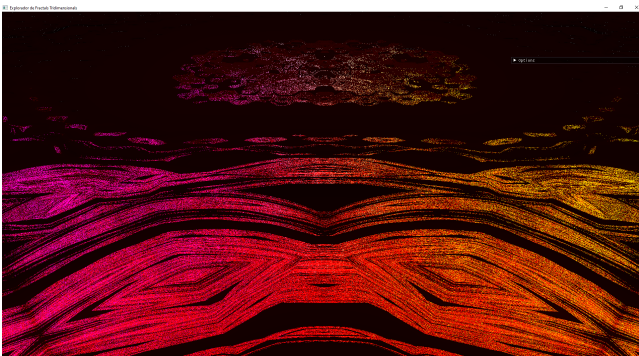


Fig. 24: Resultat obtingut de la prova fractal "Test 8: Original SDF 4."

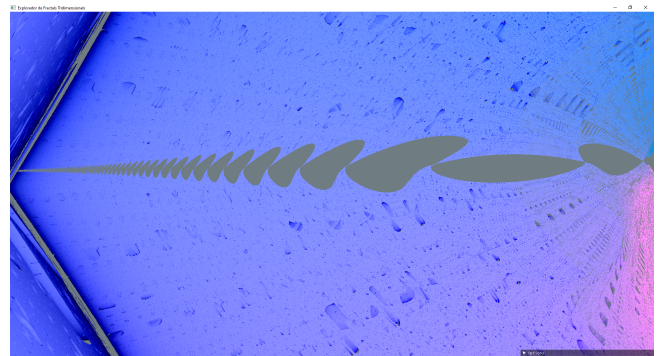


Fig. 28: Resultat obtingut de la prova fractal "Test 12: Original SDF 6."

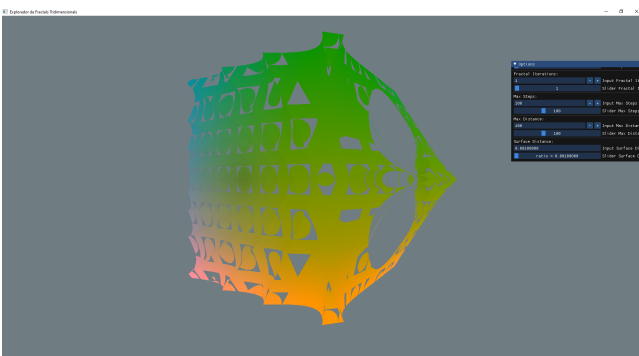


Fig. 25: Resultat obtingut de la prova fractal "Test 9: Original SDF 5."

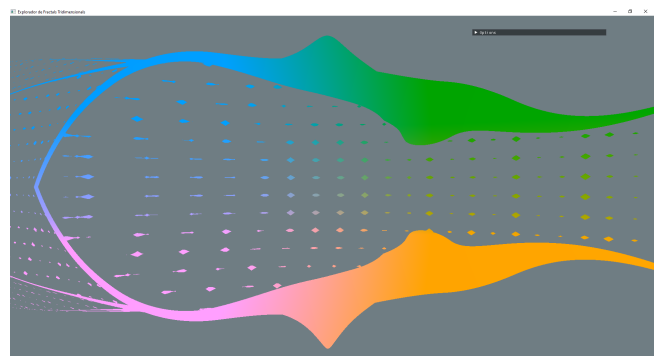


Fig. 29: Resultat obtingut de la prova fractal "Test 13: Original SDF 7."

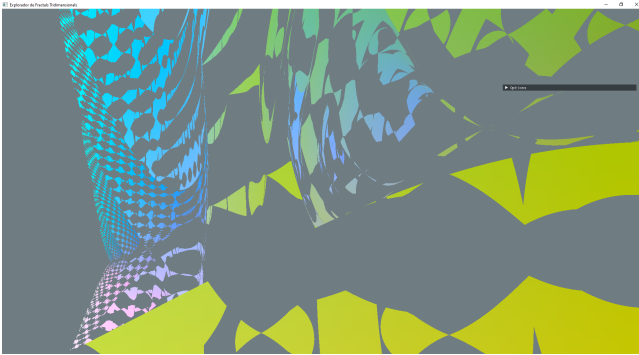


Fig. 30: Resultat obtingut de la prova fractal "Test 14: Original SDF 8."

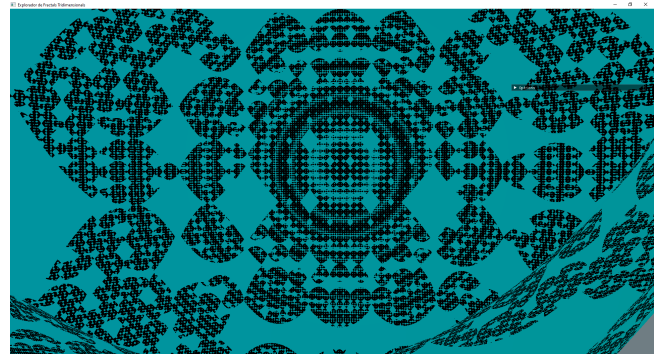


Fig. 34: Resultat obtingut de la prova fractal "Test 18: Original SDF 12."

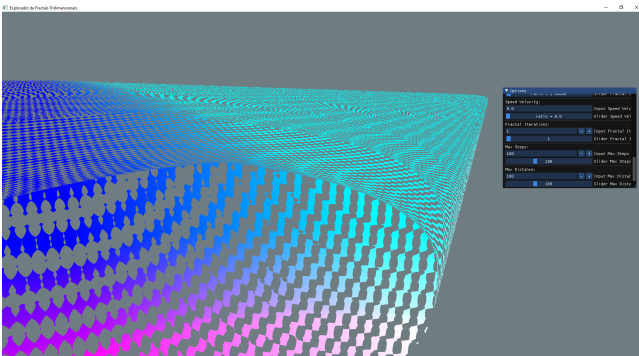


Fig. 31: Resultat obtingut de la prova fractal "Test 15: Original SDF 9."

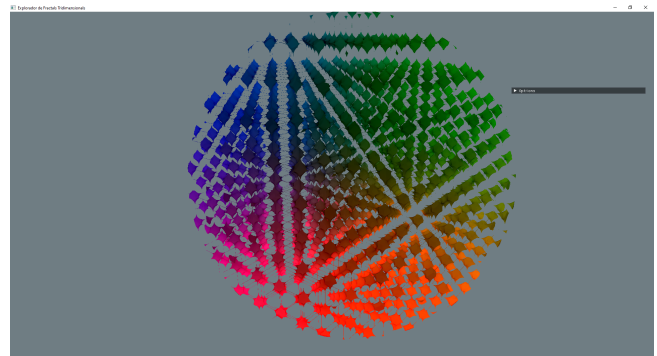


Fig. 35: Resultat obtingut de la prova fractal "Test 19: Original SDF 13 on Sphere."

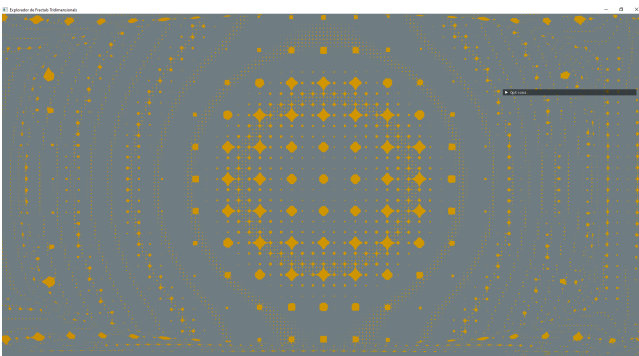


Fig. 32: Resultat obtingut de la prova fractal "Test 16: Original SDF 10."



Fig. 36: Resultat obtingut de la prova fractal "Test 20: Original SDF 14."

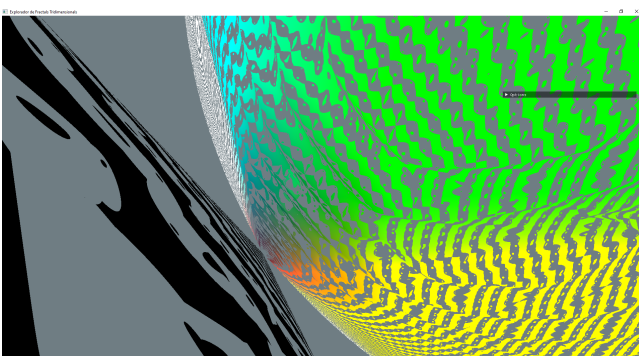


Fig. 33: Resultat obtingut de la prova fractal "Test 17: Original SDF 11."

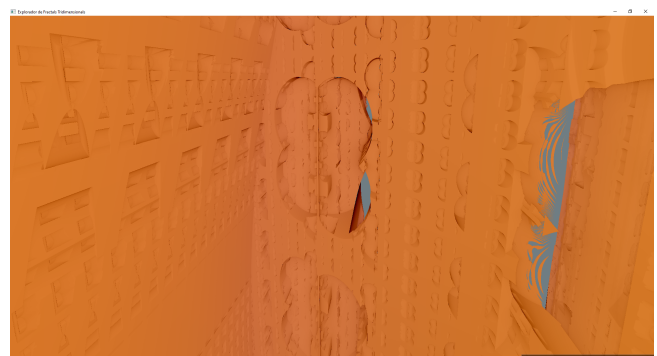


Fig. 37: Resultat obtingut de la prova fractal "Test 21: Original SDF 15."



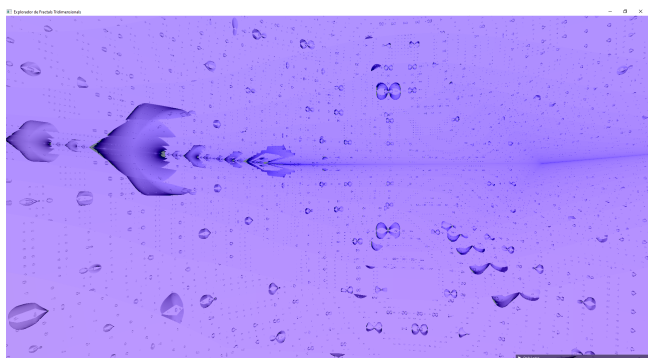


Fig. 38: Resultat obtingut de la prova fractal "Test 22: Original SDF 16."



Fig. 42: Resultat obtingut de la prova fractal "Test 26: Original SDF 20."



Fig. 39: Resultat obtingut de la prova fractal "Test 23: Original SDF 17."



Fig. 43: Resultat obtingut de la prova fractal "Test 27: Original SDF 21."

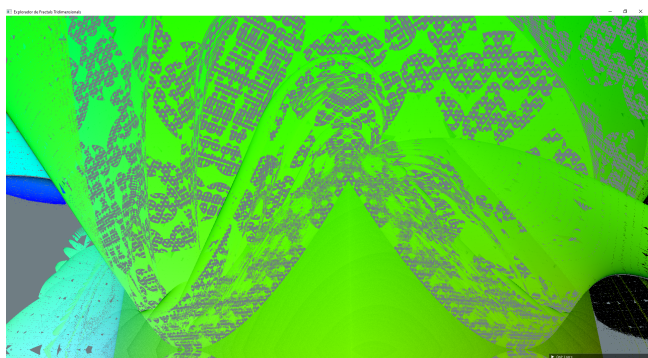


Fig. 40: Resultat obtingut de la prova fractal "Test 24: Original SDF 18."

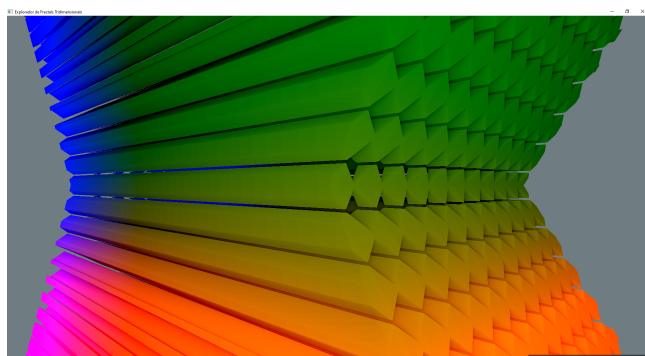


Fig. 44: Resultat obtingut de la prova fractal "Test 28: Original SDF 22."

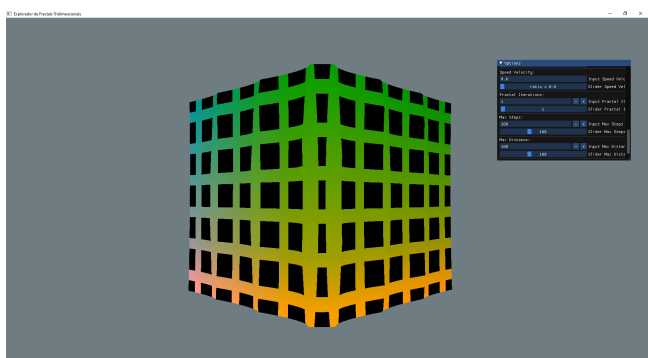


Fig. 41: Resultat obtingut de la prova fractal "Test 25: Original SDF 19."