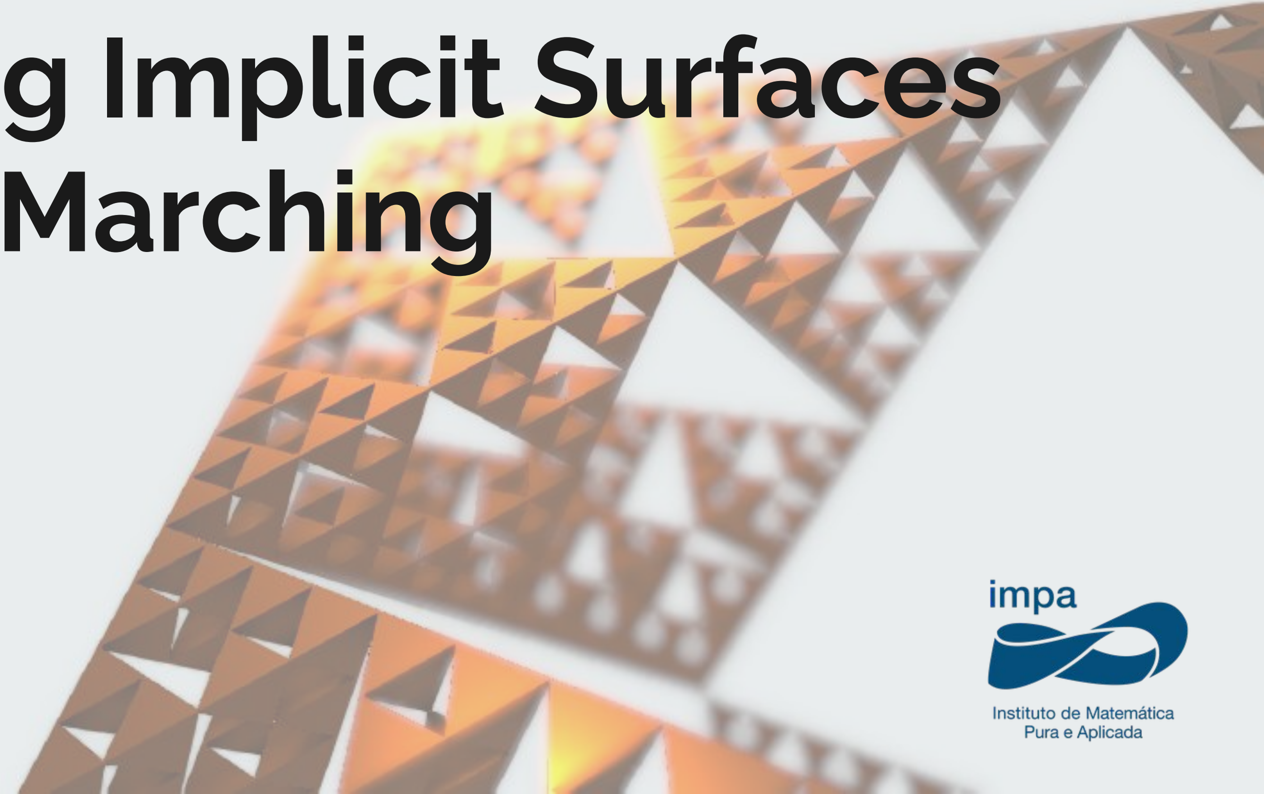
Rendering Implicit Surfaces with Ray Marching

Thales Magalhães



Background: Sierpinski pyramid

impa
Instituto de Matemática Pura e Aplicada

Our Objective

To implement a real-time visualizer for implicit surfaces using a technique known as “ray marching” in Unity.

Background: Mandelbulb detail

Rendering

Ray Tracing

Direct computation of ray intersections

Pros:

- Supports acceleration structures

Cons:

- Requires explicit intersection formulas
- Doesn't work for implicit surfaces in the general case

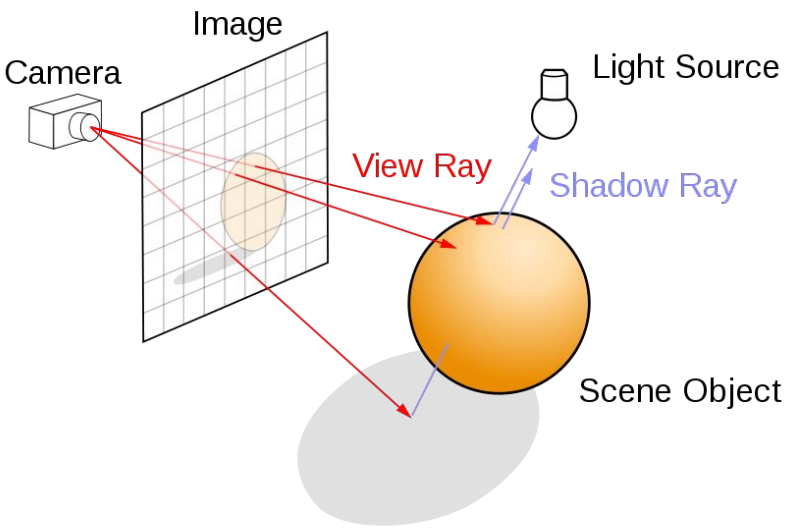


Figure 1: Diagram of the ray tracing algorithm

Ray Marching

Function is sampled along the ray

Pros:

- Easily supports transparency and participating media
- Works for implicits!

Cons:

- Resource intensive for high resolutions

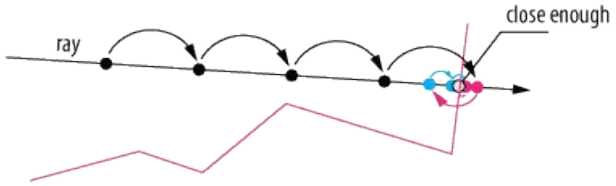


Figure 2: Diagram of the ray marching algorithm

Sphere Tracing

A variant of ray marching where the step size depends on the distance to the surface

Cons:

- Relies on signed distance functions (SDFs)

Pros:

- Much faster than uniform sampling
- SDFs enable a wide range of lighting techniques (as we'll see later)

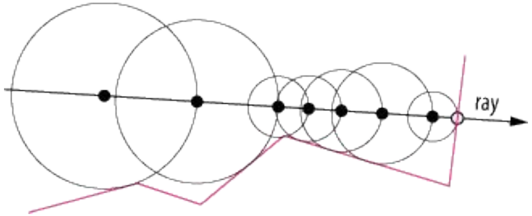


Figure 3: Diagram of the sphere tracing algorithm

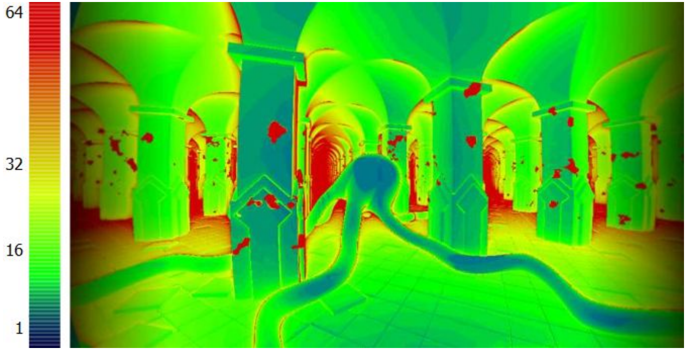


Figure 4: Number of steps taken encoded as colors for an example scene

Our Approach

- Sphere tracing via fragment shaders
- Mesh acts as domain
- Ray-marching performed independently on each fragment
- Fragments are discarded in case of a “miss”
- Front-face culling allows camera to also render from inside the domain

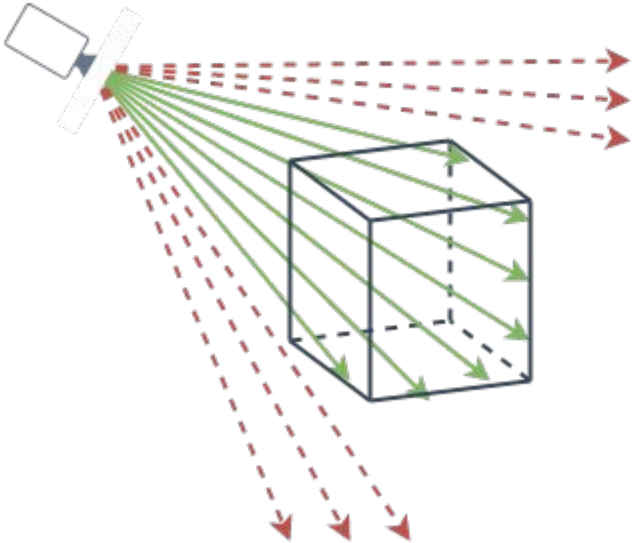


Figure 5: Diagram of our approach

Lighting

Normals

- A necessity for implementing pretty much any lighting model
- Obtained by taking the gradient of the SDF
- This can be done either analytically or numerically
- Visible aliasing for distant surfaces: can be reduced by taking finite difference steps proportional to pixel footprint (a.k.a. filtering)

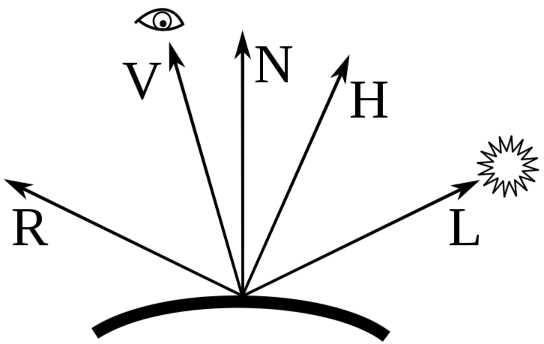


Figure 6: Phong reflection model

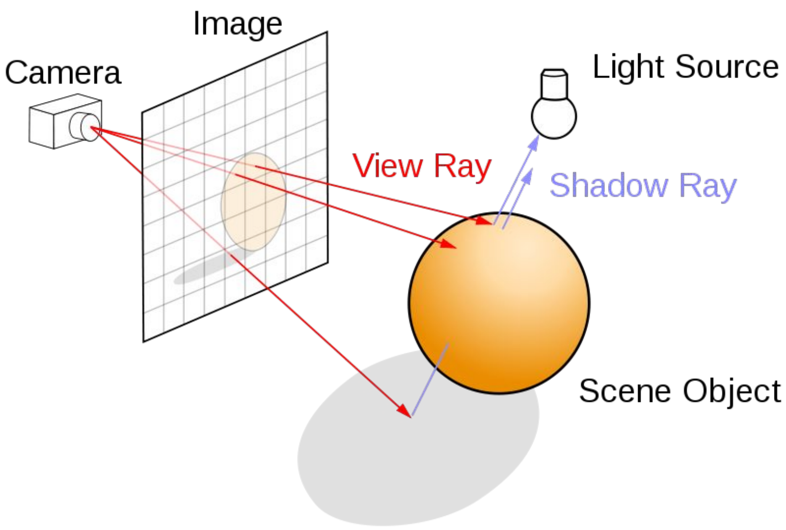
Shadows

Shadow mapping:

- Limited resolution
- Already implemented by Unity

Ray-marched shadows:

- Unlimited resolution
- More expensive than shadow mapping
- Does not account non ray-marched objects



Soft Shadows

- Effect based on distances sampled along the ray to the light source
- Very fast if already using ray-marched shadows

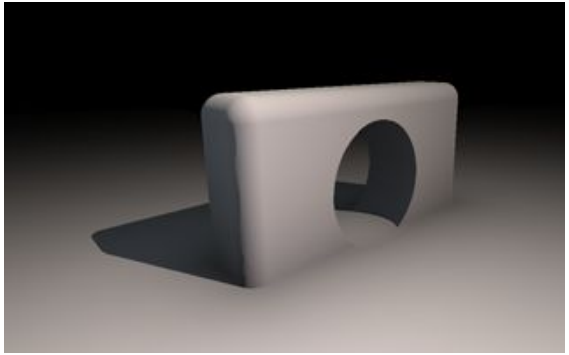


Figure 7: Hard shadows

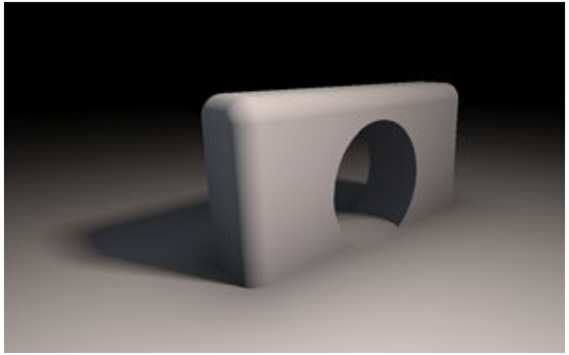


Figure 8: Soft shadows

Ambient Occlusion

- Calculated by taking a few samples along the surface normal
- Not a screen-space effect
- Very fast when compared to physically based simulations

Figure 9: Mandelbulb detail with (left) and without (right) ambient occlusion

Our Approach

- Approximated normals with filtering
- Shadow maps + ray-marched shadows for direct lighting
- Ambient occlusion applied to environment mapping

Signed Distance Functions

Primitives

- Distance functions for simple primitives can be derived analytically
- Some provide exact distances while others give a lower bound

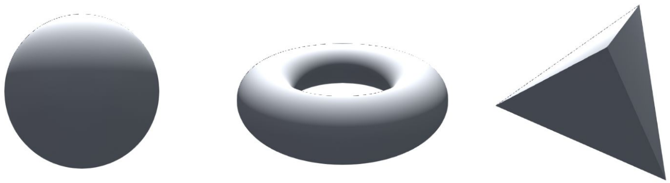


Figure 10: Examples of primitives

```
float Sphere(float3 p, float3 c, float r)
{
    return length(p - c) - r;
}
```

```
float Torus(float3 p, float r1, float r2)
{
  float2 q = float2(length(p.xz) - r1, p.y);
  return length(q) - r2;
}
```

```
float Tetrahedron(float3 p, float3 o, float s)
{
    p = (p - o) / s;
    float d = max(
        max(-p.x - p.y - p.z, p.x + p.y - p.z),
        max(-p.x + p.y + p.z, p.x - p.y + p.z));
    return s * (d - 1.0) / sqrt(3.0);
}
```

Listing 1: Corresponding signed distance functions

Constructive Solid Geometry

- Distance functions can be combined via union, intersection, difference, etc. by using ***min*** and ***max*** operations
- Additionally, it is also possible to perform smooth unions

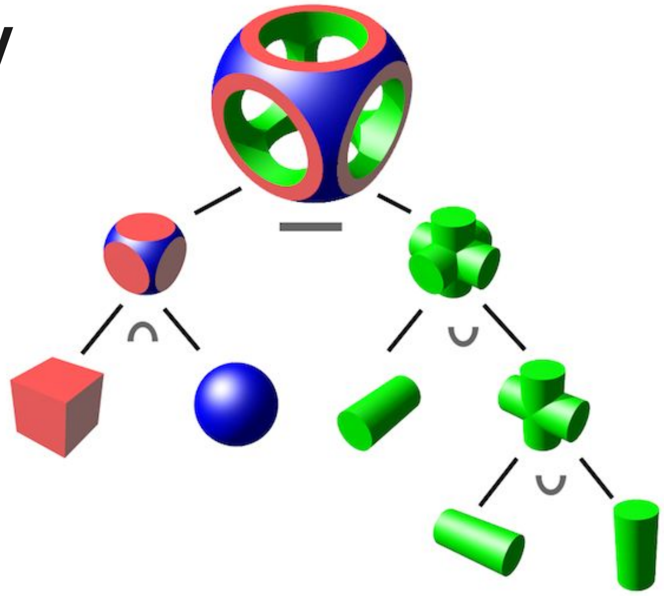


Figure 11: CSG operations

Other Operations

Deformations:

- Displacement
- Twisting
- Bending
- Etc.

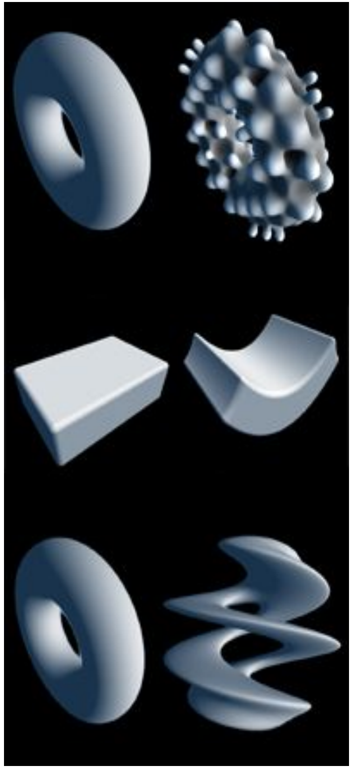


Figure 12: Example of deformations

Other Operations

Domain alterations:

- Symmetry
- Finite domain repetition
- Infinite domain repetition

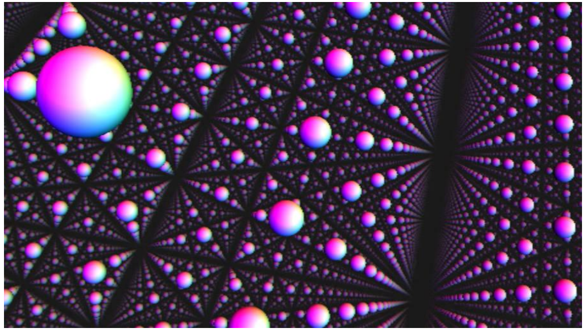


Figure 13: Infinite domain repetition

Figure 14: Example using multiple effects (Ladybug, by Inigo Quilez)

Animations

- Can be easily created by making the distance function time-dependent

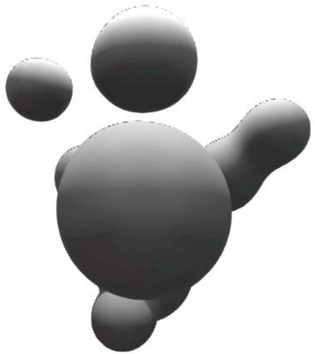


Figure 15: A simple animation created by changing parameters over time

Distance Estimated Fractals

3D Fractals

Some 3D fractals can be described by using primitive shapes and operations, e.g.:

- Sierpinski pyramid
- Menger sponge

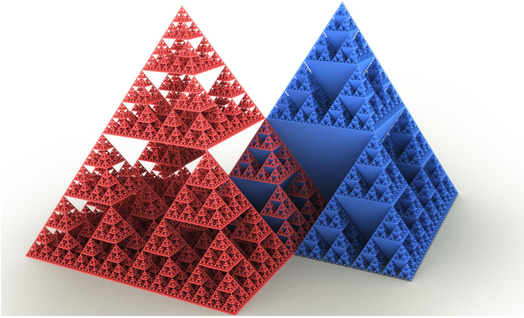


Figure 16: Sierpinski pyramid rendering

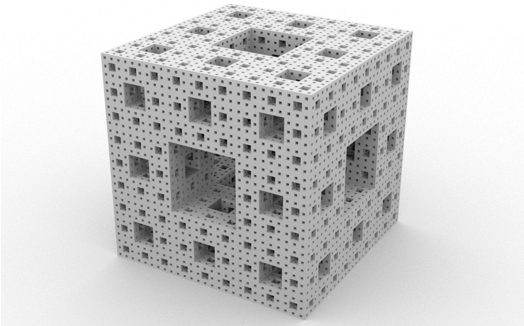


Figure 17: Menger sponge rendering

3D Fractals

Many others cannot:

- Quaternion Julia sets
- Mandelbulb
- Mandelbox
- Kaleidoscopic IFS
- Hybrid systems

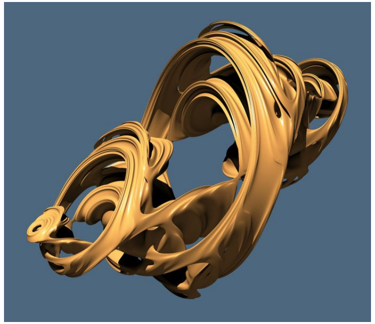


Figure 18: Quaternion Julia set rendering

Figure 19: Example of a Kaleidoscopic IFS

Distance Estimation

- These are so-called “escape time fractals”
- They are described by the convergence properties of iterative functions
- Similar to the Julia and Mandelbrot sets from 2D
- Not distance functions!

$$z_{n+1} = z_n^2 + c$$

Distance Estimation

- We need a way to estimate the equivalent distance function
- Approximations exist for the 2D case
- They work by looking at how fast the function converges/diverges
- Calculated by taking a “running derivative,” which is also iterative
- These same approximations can be adapted for 3D

$$DE = 0.5 * ln(r) * r/dr$$

$$r = |f_n(c)| \text{ and } dr = |f_n'(c)|$$

Distance Estimation

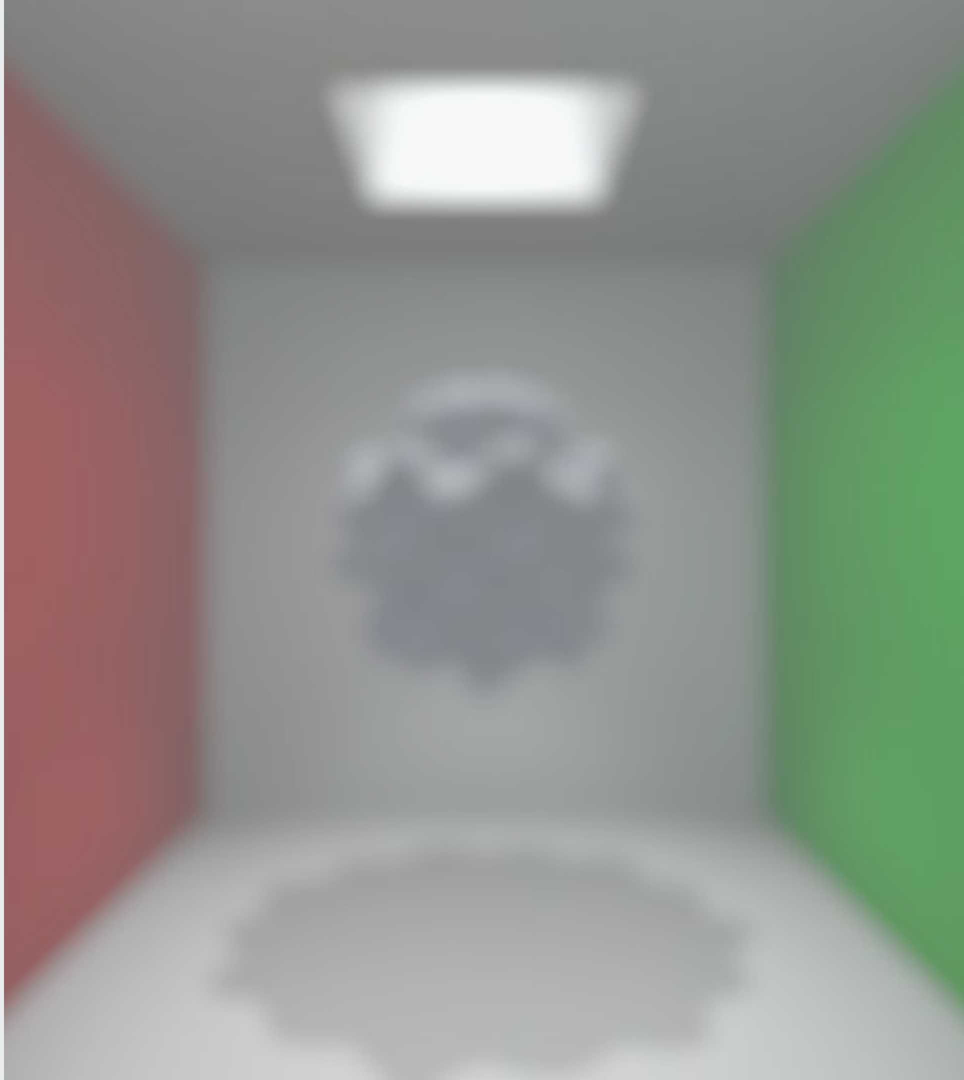
- Not exact!
- Most precise when near the surface
- Possible over-stepping especially when far away
- Solution: encapsulate fractal with a bounding volume

The Mandelbulb

- Based on the Mandelbrot set
- Extension to 3D by taking some artistic liberties
- Multiplication follows the same “geometric” properties as in the complex plane, i.e. magnitudes are multiplied and angles added together in polar coordinates
- Squaring in the original formula becomes exponentiation by an integer constant ***α***

$$z_{n+1} = z_n^{\alpha} + c$$

Demo



Bibliography

- Distance Estimated 3D Fractals
- Rendering Worlds with Two Triangles with raytracing on the GPU in 4096 bytes
- Inigo Quilez' Raymarching Articles
- Catlike Coding Unity Rendering Tutorials

References

- Ray Tracing Deterministic 3-D Fractals
- Exploring the Mandelbrot set