Mandelbulb:
Rendering 3D Fractals Using Raytracing with Distance Estimation

6.837 Computer Graphics Final Project

14 December 2016

Jimmy Zeng
Catherine Li

1 Introduction

1.1 Fractals

Fractals are self-similar mathematical structures that exhibit infinite recursive detail. One of the most interesting classes of fractals is escape time fractals, where the fractal consists of the set of points for which the orbit of a function is bounded. An example is the famous Mandelbrot set, which consists of all complex numbers c in the complex plane for which the orbit of $f(z) = z^2 + c$ does not diverge.

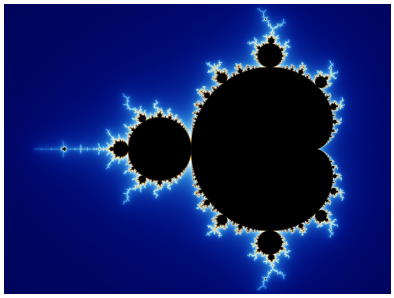

Figure 1: Mandelbrot Set

We can render the 2D fractals like the Mandelbrot set by simply iterating the function for each pixel, but this simple approach will not work in higher dimensions.

1.2 Mandelbulb

There has long been a quest to find a 3D version of the Mandelbrot set. This is difficult because there is no 3D analogue of the complex numbers (there's quaternions for 4D). Instead, note that squaring a complex number in polar form is simply squaring the magnitude and doubling the angle. With this observation came the Mandelbulb: In spherical coordinates, we can "square" a coordinate by doubling both angles and squaring the magnitude. Strangely, the power 2 formula produces a whipped cream like fractal, so the standard Mandelbulb actually uses a power 8 formula: $f(z) = z^8 + c$.

The spherical coordinates are defined as follows (for tangent we need to take the quadrant into account to get $[0, 2\pi)$ range):

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$\theta = \cos^{-1} \frac{z}{r}$$

$$\phi = \tan^{-1} \frac{y}{x}$$

We can transform from spherical coordinates back to Cartesian coordinates by using:

$$x = r \sin\theta \cos\phi$$

$$y = r \sin\theta \sin\phi$$

$$z = r \cos\theta$$

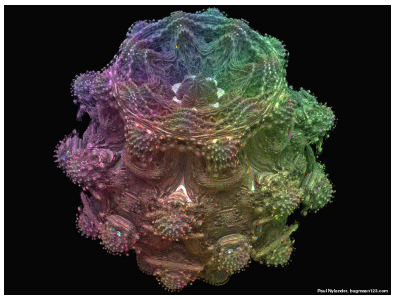

Figure 2: Mandelbulb

Since this is in three dimensions, rendering each voxel separately is too expensive (although it has been done with an optimized octree voxel engine called GigaVoxels). The typical way to render 3D fractals is ray tracing with distance estimation.

2 Background

2.1 Distance Estimation

Distance Estimation is a very general concept. A Distance Estimator provides a conservative estimate of the distance between a given position and the closest object in the scene. Distance Estimators for simple objects can be constructed easily; for a sphere it is simply distance to the center minus the sphere's radius. Distance Estimation naturally lends itself to Constructive Solid Geometry. We can compute the distance estimation for the union, intersection, and subtraction for two objects as follows:

$$DE(A \cup B) = \min(DE(A), DE(B))$$

$$DE(A \cap B) = \max(DE(A), DE(B))$$

$$DE(A \setminus B) = \max(DE(A), -DE(B))$$

The technique of rendering an implicitly defined surface in terms of its distance field by using ray tracing with a distance estimator was first developed in [Har94]. Hart, the author of this technique, called it sphere tracing because the distance estimator returns the radius of the largest sphere, centered at a give position, that does not intersect the surface.

2.2 Mandelbulb Distance Estimator

We present the following bound from [Wik] that yields a lower bound on the distance to the Mandelbrot Set. Although the bound was originally derived for the 2D Mandelbrot Set, it also holds for the 3D Mandelbulb.

Theorem 2.1 *Let $f_c(z) = z^2 + c$. Then, a lower bound of the distance from c to the Mandelbrot set is given by:*

$$\lim_{n\to\infty} \frac{1}{2} \frac{f_c^{(n)}(z) \log f_c^{(n)}(z)}{\frac{\partial}{\partial c} f_c^{(n)}(z)}$$

3 Approach

In this section we present our approach to rendering the Mandelbulb, and some pseudocode for the algorithms that we implement. Our work is based upon the results of [Chr11], which gives a thorough exploration of distance estimation for 3D fractals.

3.1 Scalar Mandelbulb Distance Estimator

We can use the following algorithm to compute the distance estimate for Mandelbulb:

3.2 Raymarching

We can use the following ray marching algorithm to step along the ray until the distance is below a certain threshold:

Algorithm 1 Mandelbulb DE

procedure DE(c)
  $r \leftarrow c$
  $dr \leftarrow 1$
  for $i = 1 \ldots maxIters$ **do**
    if $|r| > maxRadius$ **then**
      break
    end if
    $\theta \leftarrow \cos^{-1} \frac{r.z}{|r|}$
    $\phi = \tan^{-1} \frac{r.y}{r.x}$
    $dr \leftarrow |r|^{Power-1} \cdot Power \cdot dr + 1$
    $\theta \leftarrow \theta \cdot Power$
    $\phi \leftarrow \phi \cdot Power$
    $r \leftarrow |r|^{Power}(\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta) + c$
  end for
  return $\frac{1}{2}\frac{|r|\log|r|}{dr}$
end procedure

Algorithm 2 Raymarching

procedure RAYMARCH(ray, t_{min}, d_{min})
  $t \leftarrow t_{min}$
  for $i = 1 \ldots maxSteps$ **do**
    $d \leftarrow DE(ray.origin + t \cdot ray.dir)$
    $t \leftarrow t + d$
    if $d < d_{min}$ **then**
      break
    end if
  end for
end procedure

3.3 Compute Normal

We can compute the normal as the gradient of the distance field. That is, $Normal(pos) = \nabla DE(pos)$. To calculate this in practice, we can use finite differences. So,

$Normal(pos) \approx (DE(pos + \epsilon\mathbf{i}) - DE(pos - \epsilon\mathbf{i}), DE(pos + \epsilon\mathbf{j}) - DE(pos - \epsilon\mathbf{j}), DE(pos + \epsilon\mathbf{k}) - DE(pos - \epsilon\mathbf{k}))$

3.4 Phong Shading

To do Phong shading, we can use the standard Blinn-Phong model: Given light direction L, reflected eye direction R, shininess s, light intensity I, and specular coefficient k_s, the intensity of the reflected light is

$$max(L \cdot R, 0)^s * I * k_s$$

4 Results

For fractal rendering, there are a lot of adjustable parameters (for example minimum distance, number of iterations, and number of steps along the ray). We consulted [Mom14] for some good choices of parameters. Our favorite image was generated using a minimum distance of 0.0055, 120 maximum steps, and 6 iterations (see Fig 5).

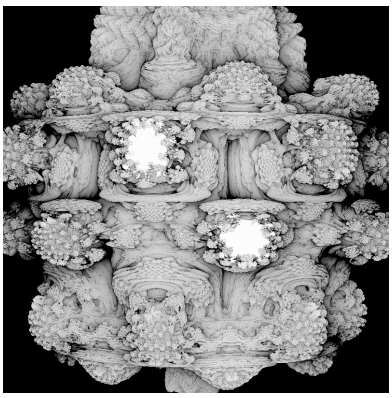

Figure 3: Mandelbulb Ambient Occlusion

The fractals can be rendered fairly well, but with a few artifacts. Fig 3 was colored using ambient occlusion so that the more detailed parts of the fractal appear darker. However, there are a few white splotches that we were unable to remove.

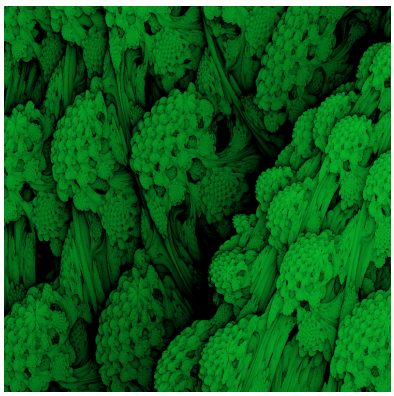

Figure 4: Mandelbulb Zoomed In

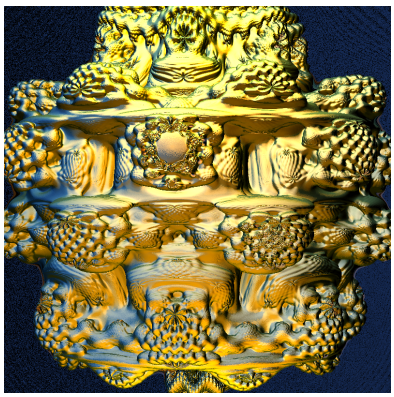

Figure 5: Mandelbulb Phong Shading

Fig. 4 shows an image of the Mandelbulb zoomed in to reveal finer details. The details are noticeably similar to the original image, reaffirming the recursive nature of fractals.

Phong shading also works fairly well (see Fig 5). There are a few artifacts that show up as a circular flat surfaces (e.g. flat spot near the center).

More (precomputed) results can be found in our YouTube video: https://www.youtube.com/watch?v=6bA6m9risQI.

---

5 Conclusion

Overall, we found that fractals are really cool and exciting. Rendering 3D fractals is best done with a distance estimator; other approaches are possible but slower. We hope to explore more with 3D fractals in the future.

Possible future work includes:

- Optimize the distance estimation by using half angle formulas to avoid using expensive trig functions.
- Run the raytracer on the GPU to achieve better performance due to parallelization.
- Render other fractals like Mandelbox.
- Use the interior distance estimation to render scenes *inside* the Mandelbulb. This requires a more complicated formula that requires estimating the periodicity of the orbit.

References

[Chr11] Mikael Hvidtfeldt Christensen. Distance estimated 3d fractals. Blog Posts, 2011. http://blog.hvidtfeldts.net/index.php/2011/06/distance-estimated-3d-fractals-part-i/.

[Har94] John C. Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12:527–545, 1994.

[Mom14] Vera Mommersteeg. Mandelbulb. Technical report, NHTV University of Applied Sciences, 2014.

[Wik] Mandelbrot set. Wikipedia. https://en.wikipedia.org/wiki/Mandelbrot_set.