CHAPTER 7

Precision Improvements for Ray/Sphere Intersection

Eric Haines,[1] Johannes Günther,[2] and Tomas Akenine-Möller[1]
[1]NVIDIA
[2]Intel

ABSTRACT

The traditional quadratic formula is often presented as the way to compute the intersection of a ray with a sphere. While mathematically correct, this factorization can be numerically unstable when using floating-point arithmetic. We give two little-known reformulations and show how each can improve precision.

7.1 BASIC RAY/SPHERE INTERSECTION

One of the simplest objects to ray trace is the sphere—no wonder that many early ray traced images featured spheres. See Figure 7-1.

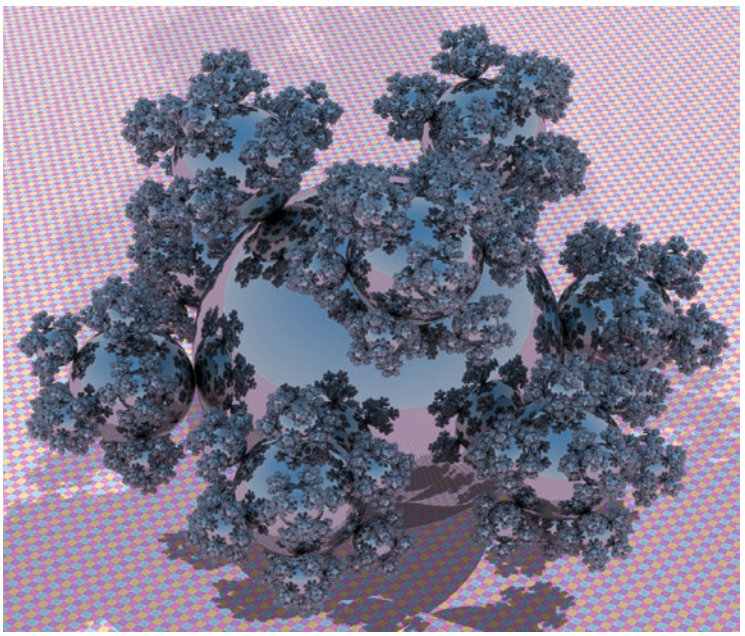

Figure 7-1. *A fractal sphereflake test scene. The ground plane is actually a large sphere. The scene contains 48 million spheres, most of subpixel size [9].*

E. Haines, T. Akenine-Möller (eds.), *Ray Tracing Gems*, https://doi.org/10.1007/978-1-4842-4427-2_7

A sphere can be defined by a center G and a radius r. For all points P at the surface of the sphere, the following equation holds:

$$(P-G)\cdot(P-G)=r^2. \tag{1}$$

To find the intersection between the sphere and the ray we can replace P by $R(t) = O + t\mathbf{d}$ (see Chapter 2). After simplification and using $\mathbf{f} = O - G$, we arrive at

$$\underbrace{(\mathbf{d}\cdot\mathbf{d})}_{a}t^2+2\underbrace{(\mathbf{f}\cdot\mathbf{d})}_{b}t+\underbrace{\mathbf{f}\cdot\mathbf{f}-r^2}_{c}=at^2+bt+c=0. \tag{2}$$

The solutions to this second-degree polynomial are

$$t_{0,1}=\frac{-b\pm\sqrt{b^2-4ac}}{2a}. \tag{3}$$

If the discriminant $\Delta = b^2 - 4ac < 0$, the ray misses the sphere, and if $\Delta = 0$, then the ray just touches the sphere, i.e., both intersections are the same. Otherwise, there will be two t-values that correspond to different intersection points; see Figure 7-2.

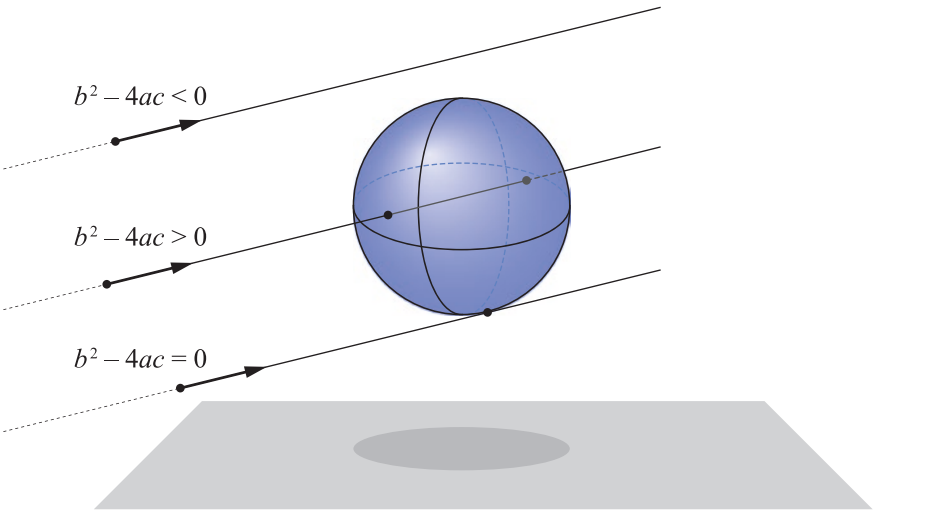

Figure 7-2. *Ray/sphere intersection test. The three different types of intersections are, from top to bottom, no hit, two intersection points, and a single hit (when the two intersections are the same).*

These t-values can be plugged into the ray equation, which will generate two intersection points, $P_{0,1} = R(t_{0,1}) = O + t_{0,1}\mathbf{d}$. After you have computed an intersection point, say, P_0, the normalized normal at the point is

$$\hat{\mathbf{n}} = \frac{P_0 - G}{r}. \tag{4}$$

7.2 FLOATING-POINT PRECISION CONSIDERATIONS

Floating-point arithmetic can break down surprisingly quickly, in particular when using 32-bit single-precision numbers to implement Equation 3. We will provide remedies for two common cases: if the sphere is small in relation to the distance to the ray origin (Figure 7-3), and if the ray is close to a huge sphere (Figure 7-4).

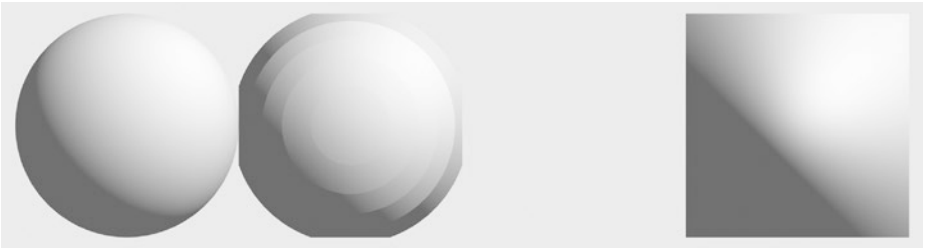

Figure 7-3. *Four unit spheres (r = 1) placed at distances of (from left to right) 100, 2000, 4100, and 8000 from an orthographic camera. Directly implementing Equation 3 can result in severe floating-point precision artifacts, up to missing intersections altogether, as for the 4100 case.*

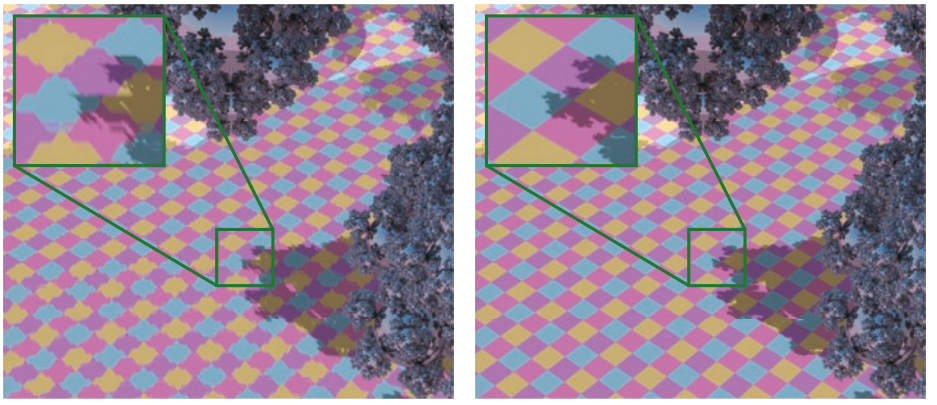

Figure 7-4. *Quadratic equation precision: the zoomed result when using the original, schoolbook test for a huge sphere forming the ground "plane" (left), and the effect of the more stable solver from Press et al. [6] (right).*

To understand why these artifacts are visible, we need a brief introduction to the properties of floating-point numbers. Ignoring the sign bit, floats are internally represented as $s \times 2^e$, with a fixed number of digits for the significand s and the exponent e. For floating-point addition and subtraction, the exponent of both numbers involved needs to match. As such, the bits of the significand of the smaller number are shifted right. The rightmost bits are lost, and thus the accuracy of this number is reduced. Single-precision floats have effectively 24 bits for the significand, which means that adding a number that is more than $2^{24} \approx 10^7$ times smaller in magnitude does not change the result.

This problem of diminished significance is greatly pronounced when calculating the coefficient $c = \mathbf{f} \cdot \mathbf{f} - r^2$ (Equation 2), because terms are squared before subtraction, which effectively halves the available precision. Note that $\mathbf{f} \cdot \mathbf{f} = \| O - G \|^2$ is the squared distance of the sphere to the ray origin. If a sphere is more than $2^{12} r = 4096r$ away from O, then the radius r has no influence on the intersection solution. Artifacts will show at shorter distances, because only a few significant bits of r remain. See Figure 7-3.

A numerically more robust variant for small spheres has been provided by Hearn and Baker [3], used for example by Sony Pictures Imageworks [4]. The idea is to rewrite $b^2 - 4ac$, where we use the convenient notation that $\mathbf{v} \cdot \mathbf{v} = \|\mathbf{v}\|^2 = \mathbf{v}^2$:

$$\begin{aligned} b^2 - 4ac &= 4a\left(\frac{b^2}{4a} - c\right) \\ &= 4\mathbf{d}^2\left(\frac{(\mathbf{f}\cdot\mathbf{d})^2}{\|\mathbf{d}\|^2} - \left(\mathbf{f}^2 - r^2\right)\right) \\ &= 4\mathbf{d}^2(r^2 - (\underbrace{\mathbf{f}^2 - (\mathbf{f}\cdot\hat{\mathbf{d}})^2}_{l^2})) \\ &= 4\mathbf{d}^2\left(r^2 - (\mathbf{f} - (\mathbf{f}\cdot\hat{\mathbf{d}})\hat{\mathbf{d}})^2\right). \end{aligned} \tag{5}$$

The last step deserves an explanation, which is easier to understand if we interpret the terms geometrically. The perpendicular distance l of the center G to the ray can be calculated either by the Pythagorean theorem, $\mathbf{f}^2 = l^2 + \left(\mathbf{f}\cdot\hat{\mathbf{d}}\right)^2$, or as the length of $\mathbf{f}$ minus the vector from the ray origin to the foot of the perpendicular, $S = O + \left(\mathbf{f}\cdot\hat{\mathbf{d}}\right)\hat{\mathbf{d}}$. See Figure 7-5. This second variant is much more precise, because the vector components are subtracted before they are squared in the dot product. The discriminant now becomes $\Delta = r^2 - l^2$. The radius r does not lose significant bits in this subtraction, because $r \geq l$ if there is an intersection. See Figure 7-6.

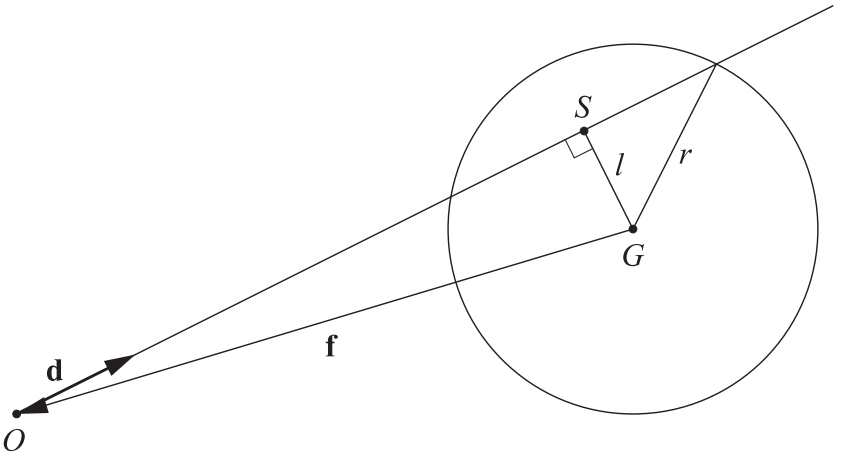

Figure 7-5. *Geometric setting for options to compute l^2. The ray origin O, the sphere center G, and its projection, S, onto the ray form a right-angled triangle.*

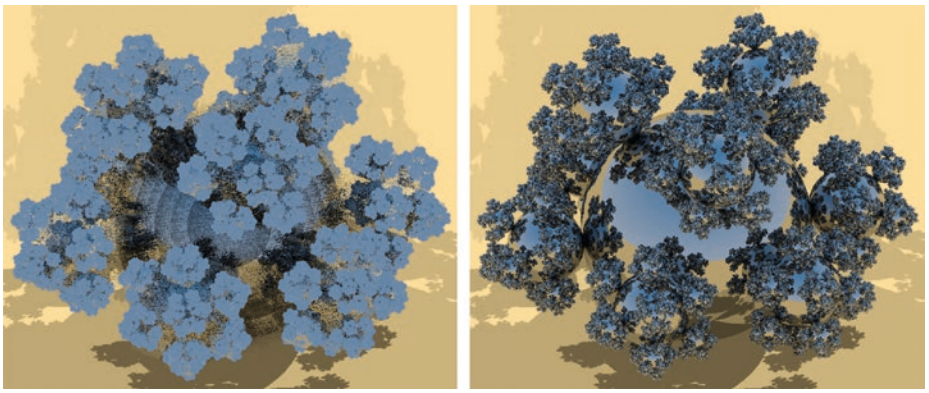

Figure 7-6. *Small sphere precision. The camera is moved 100× farther than the original view in Figure 7-1 and the field of view is narrowed: the result using the traditional quadratic formula (left), and the effect of the more stable solver from Hearn and Baker [3] (right).*

Another way we can lose precision is from subtracting numbers that are close to each other. By doing so, many of the significant bits eliminate each other, and only a few meaningful bits remain. Such a situation, often called *catastrophic cancellation*, can occur in the quadratic equation solution (Equation 3) if $b \approx \sqrt{b^2 - 4ac}$, e.g., if the intersection with a nearby huge sphere is close to the ray's origin. Press et al. [6]

give a more stable version, used in the *pbrt* renderer [5] and other systems. The key observation is that catastrophic cancellation happens only for one of the two quadratic solutions, depending on the sign of b. We can compute that solution with higher precision using the identity $t_0 t_1 = \frac{c}{a}$:

$$\begin{cases} t_0 = \dfrac{c}{q}, \\ t_1 = \dfrac{q}{a}, \end{cases} \quad \text{where} \quad q = -\frac{1}{2}\left(b + \text{sign}(b)\sqrt{b^2 - 4ac}\right). \tag{6}$$

Here, `sign` is the sign function, which returns 1 if the argument is greater than zero and –1 otherwise. See Figure 7-4 for the effect.

These two methods can be used together, as they are independent of each other. The first computes the discriminant in a more stable way, and the second then decides how best to use this discriminant to find the distances. The quadratic equation can also be solved without need for values such as "4" by reformulating the b value. The unified solution, along with other simplifications, is

$$a = \mathbf{d} \cdot \mathbf{d}, \tag{7}$$

$$b' = -\mathbf{f} \cdot \mathbf{d}, \tag{8}$$

$$\Delta = r^2 - \left(\mathbf{f} + \frac{b'}{a}\mathbf{d}\right)^2, \tag{9}$$

where Δ is the discriminant. If Δ is not negative, the ray hits the sphere, so then b' and Δ are used to find the two distances. We then compute $c = \mathbf{f}^2 - r^2$ as before to get

$$\begin{cases} t_0 = \dfrac{c}{q}, \\ t_1 = \dfrac{q}{a}, \end{cases} \quad \text{where} \quad q = b' + \text{sign}(b')\sqrt{a\Delta}. \tag{10}$$

If we can assume that the ray direction is normalized, then $a = 1$ and the solutions get slightly simpler.

Earlier exits and shortcuts are also possible if the situation warrants. For example, c is positive when the ray starts outside the sphere and negative when inside, which can tell us whether to return t_0 or t_1, respectively. If b' is a negative value, then the center of the sphere is behind the ray, so if it is also the case that c is positive, the ray must miss the sphere [2].

There is, then, no single best way to intersect a sphere with a ray. For example, if you know your application is unlikely to have the camera close to large spheres, you might not want to use the method by Press et al., as it adds a bit of complication.

7.3 RELATED RESOURCES

Code implementing these variant formulations is available on Github [8]. Ray intersectors such as those implemented in shaders in Shadertoy [7] are another way to experiment with various formulations.

ACKNOWLEDGMENTS

Thanks to Stefan Jeschke who pointed out the Hearn and Baker small spheres test, Chris Wyman and the Falcor team [1] for creating the framework on which the sphereflake demo was built, and John Stone for independent confirmation of results.

REFERENCES

[1] Benty, N., Yao, K.-H., Foley, T., Kaplanyan, A. S., Lavelle, C., Wyman, C., and Vijay, A. The Falcor Rendering Framework. https://github.com/NVIDIAGameWorks/Falcor, July 2017.

[2] Haines, E. Essential Ray Tracing Algorithms. In *An Introduction to Ray Tracing*, A. S. Glassner, Ed. Academic Press Ltd., 1989, pp. 33–77.

[3] Hearn, D. D., and Baker, M. P. *Computer Graphics with OpenGL*, third ed. Pearson, 2004.

[4] Kulla, C., Conty, A., Stein, C., and Gritz, L. Sony Pictures Imageworks Arnold. ACM Transactions on Graphics 37, 3 (2018), 29:1–29:18.

[5] Pharr, M., Jakob, W., and Humphreys, G. Physically Based Rendering: From Theory to Implementation, third ed. Morgan Kaufmann, 2016.

[6] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. Numerical Recipes: The Art of Scientific Computing, third ed. Cambridge University Press, 2007.

[7] Quílez, I. Intersectors. http://www.iquilezles.org/www/articles/intersectors/intersectors.htm, 2018.

[8] Wyman, C. A Gentle Introduction to DirectX Raytracing, August 2018. Original code linked from http://cwyman.org/code/dxrTutors/dxr_tutors.md.html; newer code available via https://github.com/NVIDIAGameWorks/GettingStartedWithRTXRayTracing. Last accessed November 12, 2018.

[9] Wyman, C., and Haines, E. Getting Started with RTX Ray Tracing. https://github.com/NVIDIAGameWorks/GettingStartedWithRTXRayTracing, October 2018.