Ray/Torus Intersection

Author: Max Wagner, mwagner@digipen.edu
Class: CS400, Summer 2004, Professor Jason Hanson

Finding the polynomial representing intersection of Ray and Torus

Let's start off with the equation for a 3D Torus, centered at the origin, lying flat in the xy-plane of a standard right-handed coordinate system:

$$f(x,y,z) = (x^2 + y^2 + z^2 - r^2 - R^2)^2 + 4R^2(z^2 - r^2) \quad \text{Eq 1}$$

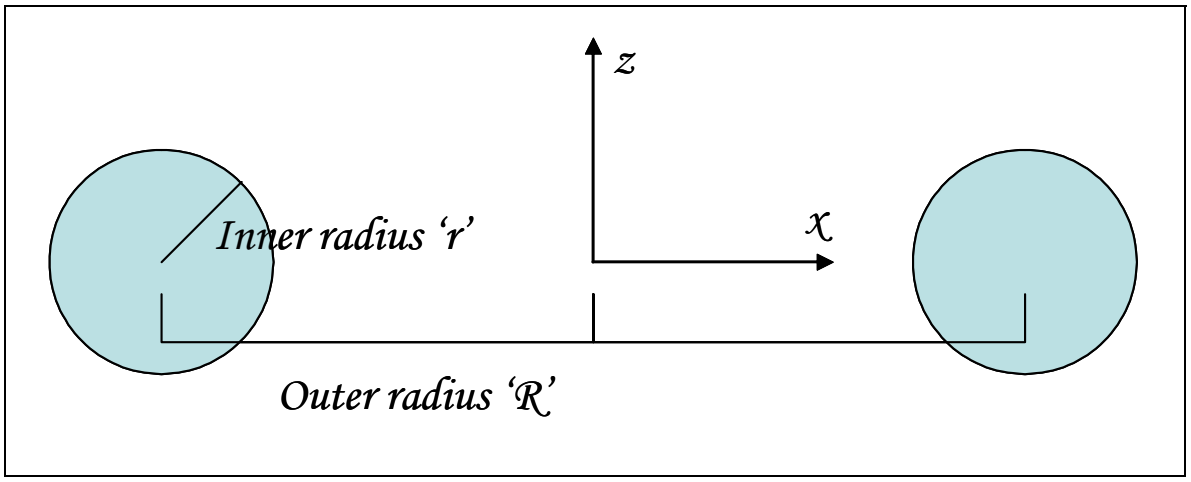

Fig 1: Cross Section of Torus lying in xy-plane

Any point on this surface satisfies the implicit equation $f(x,y,z)=0$. To compute the intersection with a ray, we substitute the ray equation for x,y,z. Defining the equation for the ray with origin $\boldsymbol{p}$ and direction $\boldsymbol{d}$ as

$$\boldsymbol{r}(t) = \boldsymbol{p} + t\boldsymbol{d} \quad \text{Eq 2}$$

or, component-wise as

$$r_x(t) = p_x + td_x$$
$$r_y(t) = p_y + td_y$$
$$r_z(t) = p_z + td_z$$

we can replace the x, y, and z from Eq 1 with $p_x + td_x$, $p_y + td_y$, and $p_z + td_z$ and set the whole thing equal to zero:

$$0 = ((p_x + td_x)^2 + (p_y + td_y)^2 + (p_z + td_z)^2 - r^2 - R^2)^2 + 4R^2((p_z + td_z)^2 - r^2) \quad \text{Eq 3}$$

We are now in the unfortunate situation of having to expand this equation, which requires some tedious arithmetic. We can already see, however, that this equation will yield a polynomial in t, as all the other quantities are known. The solution to the ray-torus intersection problem will then require solving for the roots of the polynomial and finding the smallest root greater than zero. Taking this root of t, we then plug it back into our ray

equation (Eq 2), and this yields the point of intersection. We must of course be aware that there may not be any real roots, or that they will all be less than zero; either of these situations constitutes a situation where the ray does not intersect the torus.

Continuing on then, we crank out the expansion of Eq 3.

$$0 = ((p_x + td_x)^2 + (p_y + td_y)^2 + (p_z + td_z)^2 - r^2 - \mathcal{R}^2)^2 + 4\mathcal{R}^2((p_z + td_z)^2 - r^2)$$

$$= (p_x^2 + 2tp_xd_x + t^2d_x^2 + p_y^2 + 2tp_yd_y + t^2d_y^2 + p_z^2 + 2tp_zd_z + t^2d_z^2 - r^2 - \mathcal{R}^2)^2 + 4\mathcal{R}^2(p_z^2 + 2tp_zd_z + t^2d_z^2 - r^2)$$

$$= (t^2d_x^2 + t^2d_y^2 + t^2d_z^2 + 2tp_xd_x + 2tp_yd_y + 2tp_zd_z + p_z^2 + p_y^2 + p_z^2 - r^2 - \mathcal{R}^2)^2 + 4\mathcal{R}^2(p_z^2 + 2tp_zd_z + t^2d_z^2 - r^2)$$

$$= (t^2(d_x^2 + d_y^2 + d_z^2) + 2t(p_xd_x + p_yd_y + p_zd_z) + p_z^2 + p_y^2 + p_z^2 - r^2 - \mathcal{R}^2)^2 + 4\mathcal{R}^2(p_z^2 + 2tp_zd_z + t^2d_z^2 - r^2)$$

$$= (t^2(\mathbf{d} \bullet \mathbf{d}) + 2t(\mathbf{p} \bullet \mathbf{d}) + (\mathbf{p} \bullet \mathbf{p}) - r^2 - \mathcal{R}^2)^2 + t^2 4\mathcal{R}^2 d_z^2 + t8\mathcal{R}^2 p_z d_z + 4\mathcal{R}^2 p_z^2 - 4\mathcal{R}^2 r^2$$

Letting

$$\alpha = \mathbf{d} \bullet \mathbf{d}$$
$$\beta = 2(\mathbf{p} \bullet \mathbf{d})$$
$$\gamma = (\mathbf{p} \bullet \mathbf{p}) - r^2 - \mathcal{R}^2$$

and substituting, we have

$$0 = (\alpha t^2 + \beta t + \gamma)^2 + t^2 4\mathcal{R}^2 d_z^2 + t8\mathcal{R}^2 p_z d_z + 4\mathcal{R}^2 p_z^2 - 4\mathcal{R}^2 r^2$$

$$= \alpha^2 t^4 + 2\alpha\beta t^3 + (\beta^2 + 2\alpha\gamma)t^2 + 2\beta\gamma t + \gamma^2 + t^2 4\mathcal{R}^2 d_z^2 + t8\mathcal{R}^2 p_z d_z + 4\mathcal{R}^2 p_z^2 - 4\mathcal{R}^2 r^2$$

Combining like terms, we arrive at

$$0 = \alpha^2 t^4 + 2\alpha\beta t^3 + (\beta^2 + 2\alpha\gamma)t^2 + 2\beta\gamma t + \gamma^2 + t^2 4\mathcal{R}^2 d_z^2 + t8\mathcal{R}^2 p_z d_z + 4\mathcal{R}^2 p_z^2 - 4\mathcal{R}^2 r^2$$

$$= \alpha^2 t^4 + 2\alpha\beta t^3 + (\beta^2 + 2\alpha\gamma + 4\mathcal{R}^2 d_z^2)t^2 + (2\beta\gamma + 8\mathcal{R}^2 p_z d_z)t + \gamma^2 + 4\mathcal{R}^2 p_z^2 - 4\mathcal{R}^2 r^2$$

Thus, our polynomial is of the form

$$a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0 = 0$$

Where

$$a_4 = \alpha^2 = (\mathbf{d} \bullet \mathbf{d})^2$$
$$a_3 = 2\alpha\beta = 4(\mathbf{d} \bullet \mathbf{d})(\mathbf{p} \bullet \mathbf{d})$$
$$a_2 = \beta^2 + 2\alpha\gamma + 4\mathcal{R}^2 d_z^2 = 4(\mathbf{p} \bullet \mathbf{d})^2 + 2(\mathbf{d} \bullet \mathbf{d})((\mathbf{p} \bullet \mathbf{p}) - r^2 - \mathcal{R}^2) + 4\mathcal{R}^2 d_z^2$$
$$a_1 = 2\beta\gamma + 8\mathcal{R}^2 p_z d_z = 4(\mathbf{p} \bullet \mathbf{d})((\mathbf{p} \bullet \mathbf{p}) - r^2 - \mathcal{R}^2) + 8\mathcal{R}^2 p_z d_z$$
$$a_0 = \gamma^2 + 4\mathcal{R}^2 p_z^2 - 4\mathcal{R}^2 r^2 = ((\mathbf{p} \bullet \mathbf{p}) - r^2 - \mathcal{R}^2)^2 + 4\mathcal{R}^2 p_z^2 - 4\mathcal{R}^2 r^2$$

Root Finding

To find the roots for this polynomial, we could use various techniques, including the quartic formula, which is an analytic solution for the roots of a 4th degree polynomial. However, that didn't sound like much fun to code, and nor does it extend to higher-degree polynomials. Instead, I will briefly outline my implementation of the technique suggested in class by Professor Hanson, aptly named the "Natural Method" for polynomial root finding.

The goal is to bracket all the real roots, because we know that once we've done so we can use one or a combination of techniques such as the Newton-Raphson or False Position to find a single root on a given interval. While there are highly-theoretical approaches to bracket roots, including using Sturm sequences, it seems much more intuitive to seek out the extrema of the polynomial and just check if a root lies between the extrema points (or critical points). This check is clearly just

```
if(f(x0)*f(x1) < 0) {
     // a root lies between x0 and x1
}
```

How do we find the critical points of a polynomial? By finding the roots of the derivative of that polynomial. Hence, this algorithm is recursive, and will continue all the way until we end up trying to solve a degree 1 polynomial of the form

$$a_1 t + a_0 = 0$$
$$t = -a_0/a_1,\ t \neq 0$$

This base case root will be propagated back up as the critical point of the degree 2 polynomial. Then, given user-supplied absolute min and max values for x, the degree 2 polynomial will (attempt to) create two brackets, [min, x_0] and [x_0, max], where x_0 is the value returned from solving the degree 1 polynomial. Remember these brackets depend on the above mentioned check, and also that x_0 is really between min and max. Provided there are two valid brackets, the degree 2 polynomial then can use the Newton-Raphson technique to find the roots in each bracket. Then these roots will be propagated back up to the degree 3 polynomial, and the process repeats, up until the calling polynomial is solved.

Surface Normals

Once we've found the intersection between the ray and the torus, we will need to find the surface normal at that point. To do so, we simply take the gradient of the original function for the torus:

$$\nabla f(x,y,z) = \nabla [(x^2 + y^2 + z^2 - r^2 - R^2)^2 + 4R^2(z^2 - r^2)] \quad \text{Eq 4}$$

Which, component-wise, yields:

$$\delta f/\delta x = 4x(x^2 + y^2 + z^2 - r^2 - R^2)$$
$$\delta f/\delta y = 4y(x^2 + y^2 + z^2 - r^2 - R^2)$$
$$\delta f/\delta z = 4z(x^2 + y^2 + z^2 - r^2 - R^2) + 8R^2z$$

With our intersection point in hand, all we have to do is plug it into Eq 4 (its component functions), and out pops our surface normal vector.

Warping the Torus into World Space

While a circular torus centered at the origin of the world is fine and dandy, it'd be nice to be able to move it around and stretch it. Fortunately, this is an easy problem rooted in basic Linear Algebra and Matrix math.

Taking our original torus with outer radius R and inner radius r, we can specify a translation from the world origin $\boldsymbol{s}$, along with a new basis matrix $\mathcal{M} = [\boldsymbol{x}' \mid \boldsymbol{y}' \mid \boldsymbol{z}']$ to which we will map the torus's original basis matrix $\mathcal{W} = [\boldsymbol{x} \mid \boldsymbol{y} \mid \boldsymbol{z}]$ (this is just the original world basis, the standard basis of $\mathbf{R}^3$). This mapping is fortunately very simple: just multiply all points $\{p\}$ on the torus by $\mathcal{M}$ and add $\boldsymbol{s}$. Note that the matrix $\mathcal{M}$ need not be a special-orthogonal or even orthogonal matrix; it can contain rotation, shearing, and scaling. It is important that this matrix be non-singular, however, as we shall see below.

Given that our rendering technique is ray-tracing and not polygon rasterization, we don't have a discrete list of points for our torus, so it doesn't make a lot of sense to perform this transformation on the torus. Denoting the space of our original, un-transformed torus as Object Space and calling the above warping the torus's World Transformation (the transformation taking the torus from its Object Space and placing it in the World), the proper course of action is to transform our ray from World Space to the Object Space of the torus. How do we do this? By the inverse of the torus's World Transformation, i.e., by first subtracting $\boldsymbol{s}$ and then multiplying by $\mathcal{M}^{-1}$.

In summary, before solving for the roots of the polynomial derived above, where we defined our ray as $\boldsymbol{p} + t\boldsymbol{d}$, we first transform this ray as follows:

$$\boldsymbol{p}' = \mathcal{M}^{-1}(\boldsymbol{p} - \boldsymbol{s})$$

$$\mathbf{d}' = \mathcal{M}^{-1}(\mathbf{d})$$

Note that when transforming the direction vector $\mathbf{d}$, we do not first subtract the translation component $\mathbf{s}$, as the vector represents a displacement, not a position. We then use these new quantities $\mathbf{p}'$ and $\mathbf{d}'$ when solving the roots of the polynomial. Note also that when finding the surface normal using the above derivation, this surface normal will be in the torus's Object Space. While deriving the correct transformation for normal vectors is just slightly beyond the scope of this article, you can take it on faith (or research it yourself :) that the correct way to do so is as follows. Letting $\mathbf{n}$ and $\mathbf{n}'$ denote the World and Object Space normals, respectively:

$$\mathbf{n} = (\mathcal{M}^{-1})^{T}(\mathbf{n}')$$

We can then go ahead and use the normal in World Space for further lighting computations.

File Format

Specifying the Torus requires the same information as for the ellipsoid, with the addition of the inner and outer radii. The new Torus specification looks like

TORUS *(cx, cy, cz) r R (ux, uy, uz) (vx, vy, vz) (wx, wy, wz) <surface properties>*
A torus with center *(cx, cy, cz)*, inner radius *r*, outer radius *R*, and semiaxes *(ux, uy, uz)*, *(vx, vy, vz)*, and *(wx, wy, wz)*.