2D Optical Simulation

Samuel Davies

August 13, 2025

Abstract

Part of my extension work was creating a program for general purpose 2D optical simulation, by resolving the path of rays of light. This paper details the mathematics involved in this.

My simulation will consist of rays and optical elements. Rays represent the path of light, and optical elements interact with the rays. I first need to resolve collisions with rays and optical elements, then create a new ray at this point with angle decided by the ray-optical element interaction. Therefore, first I discuss resolving the collisions then computing the interaction. Finally, I discuss other requirements, such as wavelength to RGB conversion for drawing and creating lenses.

1 Ray-Optical Element Collisions

1.1 Defining rays

Rays of light are linear. They will be expressed as:

$$p = a + \lambda\hat{b} \quad I, a, \lambda \in \mathbb{R} \quad p, \hat{b} \in \mathbb{R}^2 \quad \lambda \in [0, \lambda_{\max}] \quad I > 0 \tag{1}$$

r is any point on the ray, parameterised by λ. I is the intensity. A ray will be initialised given a and $\hat{b}$, and then resolved to find $\lambda_{\max}$ (the distance between the start and optical element it hits). Therefore, λ is constrained to $[0, \lambda_{\max}]$ as the ray cannot travel past this point or backwards. Note the optical element it hits is the intersections with minimum λ, as the ray hits the first optical element in its path. If it does not hit an optical element, $\lambda_{\max} = \infty$. This is because OpenGL culls off-screen drawing using the GPU, therefore resolving rays using the screen edges is unnecessary and slower. Note practically $\lambda_{\max} = 10^{10}$ to avoid float overflow with IEEE-754. Optical elements can either be linear, arcs, or parabolas.

1.2 Ray-Line Collisions

A linear optical element will also use equation 1. However, to distinguish between lines, they will be as such:

$$L_n | p_n = a_n + \lambda_n \hat{b}_n \quad \text{and} \quad \lambda \in [0, \lambda_{n\max}] \tag{2}$$

Letting a ray be L_n and an optical element be L_m, setting $p_n = p_m$, intersection between them is found: $\begin{bmatrix} \hat{b}_{mx} & -\hat{b}_{nx} \\ \hat{b}_{my} & -\hat{b}_{ny} \end{bmatrix}^{-1} (a_n - a_m) = \begin{bmatrix} \lambda_m \\ \lambda_n \end{bmatrix}$ This could be evaluated using a linear algebra library, however I do without for efficiency as the library allocates dynamic arrays. λ_m is used to check the intersection is between the line endpoints: $\lambda_m \in [0, \lambda_{m\max}]$, and λ_n is used to check if this is the optical element it hits (collision with minimum value > 0).

1.3 Ray-Arc Collisions

1.3.1 Defining Arcs

A circle can be expressed as

$$\|p - c\| = r \quad r \in \mathbb{R} \quad p, c \in \mathbb{R}^2 \tag{3}$$

c is the centre, r is the radius, and p is any point on the circle. Expressing an arc can be done in multiple ways. The most straightforward way is to constrain the angle from the centre, however this will require evaluating

$tan^{-1}(\frac{(p-c)_y}{(p-c)_x})$. Trigonometric functions and division are computationally expensive (section 3.4), so avoiding this is ideal. Instead, an arc can be expressed as a circle sliced by a line. This would only require resolving the intersection then evaluating which side of the line the point is on. An arc will be expressed with circle equation 3 and with line L_m using equation 2. In the intersection with ray $n|p = a_n + \lambda_n \hat{b}_n$, λ_n is found such that p lies on both the line and ray. Letting the ray now be $n|p = a_n + \mu_n \hat{b}_n$, and substituting into equation 3: $\|(a_n + \mu_n \hat{b}_n) - c\| = r$, values of μ_n can be found such that p lies on both the circle and the ray. The value of μ_n can be compared to λ_n to determine which side of the line the circle collision is on.

1.3.2 Constraining to Major or Minor

However, $\mu_n > \lambda_n$(or vice versa) does not determine if p is in the major or minor arc. This is because it depends on the direction of L_n. Instead, $\mu_n > \lambda_n \oplus \theta > 0$, where θ is the angle between $\hat{b}_n$ and $\hat{b}_m$, gives whether p is in the minor or major arc. This is because the xor will flip the result of the comparison based on the sign of the angle (whether $\hat{b}_n$ is clockwise of $\hat{b}_m$). Selection of major or minor can be done by changing the sign of $\hat{b}_m$. Letting $m \in \mathbb{B}$ be true if minor is desired,

$$\begin{vmatrix} \hat{b}_{nx} & \hat{b}_{mx} \\ \hat{b}_{ny} & \hat{b}_{my} \end{vmatrix} > 0 \Leftrightarrow \theta > 0 \tag{4}$$

$$\begin{vmatrix} (c-a_m)_x & \hat{b}_{mx} \\ (c-a_m)_y & \hat{b}_{my} \end{vmatrix} > 0 \oplus m \Rightarrow \text{flip sign of } \hat{b}_m \tag{5}$$

Equation 4 is true by definition of the determinant, so this can be used instead. Yet again this can be computed with a linear algebra library, however I do without. Using the same principle, equation 5 is used to determine if $\hat{b}_m$'s sign should be flipped. As the arc used is always pointing anticlockwise of $\hat{b}_m$, whether this arc is major or minor can be determined: If $\hat{b}_m$ is anticlockwise of $(c-a)$, the arc used is minor. Therefore, I xor this with if minor is desired. First letting $\alpha = a_n - c$ to simplify then solving $\|(\alpha + \mu_n \hat{b}_n)\| = r$ through Pythagerous and expanding, $(\|\hat{b}_n\|^2){\mu_n}^2 + 2(\alpha \cdot \hat{b}_n)\mu_n + (\|\alpha\|^2 - r^2)$. Note r^2 is evaluated once and stored, as the simulation is optimised for real-time rays and fixed optics. To simplify, let $a = \|\hat{b}_n\|^2, b = 2(\alpha \cdot \hat{b}_n), c = \|\alpha\|^2 - r^2$. Through quadratic formula, $a{\mu_n}^2 + b\mu_n + c = 0, a \neq 0 \Rightarrow \mu_n = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$

1.3.3 Finding Minimum Valid Root

Because a collision with a ray and optical element is the one with minimum λ, I need to find which root to return to achieve this. Note $a = 0 \Rightarrow \mu_n = \frac{c}{-b}$, and if $a = b = 0$, any μ_n is valid. This means $\mu_n = 0$ is valid and rays cannot end where they start, so there is no collision. This is because this is where the previous ray ended, and created the new ray, and letting this happen would make an infinite loop. Rays can also only travel forwards so $\mu_n > 0$. The collision for a quadratic is as follows (where 0 means no collision):

$$f|\mathbb{R},\mathbb{R},\mathbb{R} \to \mathbb{R}|f(a,b,c) = \begin{cases} \frac{c}{-b} & a = 0 \wedge ((b < 0 \wedge c > 0) \vee (b > 0 \wedge c < 0)) \\ \frac{-b-\sqrt{b^2-4ac}}{2a} & (a > 0 \wedge -b - \sqrt{b^2-4ac} > 0) \vee (a < 0 \wedge -b - \sqrt{b^2-4ac} < 0) \\ \frac{-b+\sqrt{b^2-4ac}}{2a} & (a > 0 \wedge -b + \sqrt{b^2-4ac} > 0) \vee (a < 0 \wedge -b + \sqrt{b^2-4ac} < 0) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

However only collisions where $\mu_n > \lambda_n \oplus \begin{vmatrix} \hat{b}_{nx} & \hat{b}_{mx} \\ \hat{b}_{ny} & \hat{b}_{my} \end{vmatrix} > 0$ are valid (equation 4). Defining a new function,

$$g|\mathbb{R},\mathbb{R},\mathbb{R},\mathbb{R},\mathbb{B} \to \mathbb{R}|g(a,b,c,d,e) =$$

$$\begin{cases} \frac{c}{-b} & a = 0 \wedge ((b < 0 \wedge c > 0) \vee (b > 0 \wedge c < 0)) \wedge (\frac{c}{-b} > d \oplus e) \\ \frac{-b-\sqrt{b^2-4ac}}{2a} & ((a > 0 \wedge -b - \sqrt{b^2-4ac} > 0) \vee (a < 0 \wedge -b - \sqrt{b^2-4ac} < 0)) \wedge (\frac{-b-\sqrt{b^2-4ac}}{2a} > d \oplus e) \\ \frac{-b+\sqrt{b^2-4ac}}{2a} & ((a > 0 \wedge -b + \sqrt{b^2-4ac} > 0) \vee (a < 0 \wedge -b + \sqrt{b^2-4ac} < 0)) \wedge (\frac{-b+\sqrt{b^2-4ac}}{2a} > d \oplus e) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where $\mu_n = g(a,b,c,d,e), d = \lambda_n, e = \begin{vmatrix} \hat{b}_{nx} & \hat{b}_{mx} \\ \hat{b}_{ny} & \hat{b}_{my} \end{vmatrix} > 0$.

1.4 Ray-Parabola Collisions

A parabola is a quadratic in terms of x and y, for example $y = kx^2 + bx + c$. However, this does not allow rotation. A better way to express a parabola is with centre, angle, and k. To simplify, I let my translated and rotated point equal α. Therefore, my parabola is simply:

$$\theta, k \in \mathbb{R} \quad p, c \in \mathbb{R}^2$$

$$\alpha = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} (p - c) \tag{8}$$

$$\alpha_y = k\alpha_x^2 \tag{9}$$

Note constants here can yet again be evaluated once and stored, such as $\cos(\theta)$ and $\sin(\theta)$, to optimise for real-time rays and fixed optics. Letting the ray again be L_n(equation 2) and letting $\alpha = p_n$ to find the intersection, plugging into equation 9: $a_{ny} + \lambda_n \hat{b}_{ny} = k(a_{nx} + \lambda_n \hat{b}_{nx})^2$ is a quadratic I can solve for λ_n: $(k\hat{b}_{nx}^{\ 2})\lambda_n^2 + (2ka_{nx}\hat{b}_{nx} - \hat{b}_{ny})\lambda_n + (ka_{nx}^2 - a_{ny}^2)$. This could be solved using equation 6, however it is important to constrain the parabola's endpoints. This could be done using equation 1.3.3, although it is more efficient to simply constrain the x: $\alpha_x \in [x_{\min}, x_{\max}]$. Hence, I defined a new function:

$$h|\mathbb{R},\mathbb{R},\mathbb{R},\mathbb{R},\mathbb{R} \to \mathbb{R}|h(a,b,c,d,e) =$$

$$\begin{cases} \frac{c}{-b} & a = 0 \wedge ((b < 0 \wedge c > 0) \vee (b > 0 \wedge c < 0)) \wedge \frac{c}{-b} \in [d,e] \\ \frac{-b-\sqrt{b^2-4ac}}{2a} & (a > 0 \wedge -b - \sqrt{b^2-4ac} > 0) \vee (a < 0 \wedge -b - \sqrt{b^2-4ac} < 0) \wedge \frac{-b-\sqrt{b^2-4ac}}{2a} \in [d,e] \\ \frac{-b+\sqrt{b^2-4ac}}{2a} & (a > 0 \wedge -b + \sqrt{b^2-4ac} > 0) \vee (a < 0 \wedge -b + \sqrt{b^2-4ac} < 0) \wedge \frac{-b+\sqrt{b^2-4ac}}{2a} \in [d,e] \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $\lambda_n = h((k\hat{b}_{nx}^{\ 2}), (2ka_{nx}\hat{b}_{nx} - \hat{b}_{ny}), (ka_{nx}^2 - a_{ny}^2), x_{\min}, x_{\max})$

2 Ray-Optical Element Interactions

2.1 Defining the interactions

When light hits an optical element, it refracts and/or reflects. Diffraction is a result of these and interference, however it is impractical to simulate diffraction this way, so I will also use equations for diffraction. These interactions can be implemented by terminating a ray upon collision and making a new ray at its end. The position of the new ray is the end of the first ray, and direction is determined by the refraction or reflection. Letting the first ray be L_n and new ray be L_m (equation 2), and the endpoint be α,

$$\beta = a_n + \lambda_{n\max}\hat{b}_n \tag{11}$$

$$f|\mathbb{R}^2,\mathbb{R}^2 \to \mathbb{R}^2 \quad g|\mathbb{R}^2 \to \mathbb{R}^2$$

$$a_m = \beta \quad \hat{b}_m = f(\hat{b}_n, g(\beta)) \tag{12}$$

Here f is doing the refraction, reflection, or diffraction, given the previous direction($\hat{b}_n$), and the normalized tangent to the optical element (a function, g of the position of the position of intersection(α)).

2.2 Finding Tangents

2.2.1 Line Tangents

Tangents for lines are very easy, Letting the line be L_o (equation 2):

$$g(\beta) = \hat{b}_o \tag{13}$$

2.2.2 Arc Tangents

Tangents for arcs can be found by first finding the normalized normal, then rotating by $\frac{\pi}{2}$ (multiply by $\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$). As the point will lie on the arc, dividing the vector from the centre to the point by the radius gives this tangent,

therefore(using equation 3):

$$g(\beta) = \frac{1}{r}\begin{pmatrix} -(\beta - c)_y \\ (\beta - c)_x \end{pmatrix} \tag{14}$$

This can be optimised by storing $\frac{1}{r}$ because I am designing for real-time rays and static optics(mentioned earlier), and division is computationally expensive (section 3.4).

2.2.3 Parabola Tangents

Tangents for parabolas can be found through use of the derivative: I can find a vector tangent to the parabola at the intersection point: $\begin{pmatrix} 1 \\ \frac{dy}{dx}(\beta_x) \end{pmatrix}$, then normalize. Using equation 8, 9 and letting $\beta = \alpha$:

$$g(\beta) = \frac{1}{\sqrt{1 + 4k^2\beta_x^2}}\begin{pmatrix} 1 \\ 2k\beta_x \end{pmatrix} \tag{15}$$

2.3 Reflection

Reflection in a line can be achieved using the dot product of the point with the line's normal, to find the perpendicular distance to the line. The point can then be moved twice this distance the opposite direction. Note for reflection of direction, I reflect using the line's tangent, and then negate. Also, here a is ray direction and b is tangent (equation 12)

$$f(a, b) = -(a - 2(a \cdot b)b) \tag{16}$$

2.4 Refraction

2.4.1 Refraction Equations

The most known refraction equation is $n_1 \sin(\theta_1) = n_2 \sin(\theta_2) \quad n_1, n_2, \theta_1, \theta_2 \in \mathbb{R}$. This relates incident angle between ray and normal(θ_1) to refracted angle between ray and normal(θ_2) using corresponding refractive indexes (n_1, n_2). However, this can also be written using cosine: By using the identity

$$cos(\frac{\pi}{2} - x) = \sin(x) \tag{17}$$

$n_1 \cos(\phi_1) = n_2 \cos(\phi_2)$, where ϕ is the angle between the ray and the surface. This is more useful as cosine can be found efficiently using dot product, as both tangent and direction are normalized ($\cos(\phi_1) = (a \cdot b)$). I also use the Pythagorean identity to find $\sin(\phi_2)$ as square root evaluates faster than inverse cosine and sine (section 3.4) (equation 19). Finally, the wavelength is also changed through refraction(equation 20).

$$\cos(\phi_2) = \frac{n_1}{n_2}(a \cdot b) \tag{18}$$

$$\sin(\phi_2) = \sqrt{1 - \cos^2(\phi_2)} \quad \{\|\cos(\phi_2)\| \leqq 1\} \tag{19}$$

$$\lambda_2 = \frac{n_1}{n_2}\lambda_1 \tag{20}$$

$$f(a, b) = \begin{bmatrix} \cos(\phi_2) & -\sin(\phi_2) \\ \sin(\phi_2) & \cos(\phi_2) \end{bmatrix} b \tag{21}$$

I then rotate the tangent by ϕ_2, as this is the angle between the tangent and refracted ray.

2.4.2 Flipping Sides

As equation 21 uses rotation of b, the result can be wrong depending on the sign of b; If b is anticlockwise of a, the refracted direction would be on the wrong side. Moreover, which refractive index n_2 is depends on if b is anticlockwise of a (Note it is more correct to switch n_1 and n_2 however in my code n_1 is decided by the incident

ray). Therefore, using equations 4,18,19 and 21:

$$f(a,\beta) = \begin{bmatrix}\cos(\phi_2) & -\sin(\phi_2)\\ \sin(\phi_2) & \cos(\phi_2)\end{bmatrix} b$$
$$\begin{vmatrix}a_x & b_x\\ a_y & b_y\end{vmatrix} > 0 \Rightarrow n_2 = n_a, b = -\beta$$
$$\begin{vmatrix}a_x & b_x\\ a_y & b_y\end{vmatrix} \leq 0 \Rightarrow n_2 = n_b, b = \beta \tag{22}$$

2.5 Ray Intensity

When light hits a refractive material, it can reflect and refract. Therefore, the incident intensity is split between these two rays (or one depending on angle). These intensities are found with Fresnel equations from [4] (equations 24 and 25). Letting $r = R/I$ and $t = R/I$, where $T, R, I \in \mathbb{R}$ (where $x_\perp^2 + x_\parallel^2 = x^2$ is true for each intensity) are the transmitted (refracted), reflected, and incident intensity respectively:

$$\cos(\theta_2) = \sin(\phi_2) \quad \cos(\theta_1) = \sin(\phi_1) = \sqrt{1-(a\cdot b)^2} \tag{23}$$
$$r_\parallel = \frac{n_2\cos(\theta_1) - n_1\cos(\theta_2)}{n_2\cos(\theta_1) + n_1\cos(\theta_2)}$$
$$r_\perp = \frac{n_1\cos(\theta_1) - n_2\cos(\theta_2)}{n_1\cos(\theta_1) + n_2\cos(\theta_2)} \tag{24}$$
$$t_\perp = r_\perp + 1 \quad t_\parallel = \frac{n_1}{n_2}(r_\parallel + 1) \tag{25}$$
$$I_\perp = I_\parallel = \frac{1}{\sqrt{2}} I \tag{26}$$
$$R = \frac{1}{\sqrt{2}} I \sqrt{r_\parallel^2 + r_\perp^2} \tag{27}$$
$$T = \frac{1}{\sqrt{2}} I \sqrt{(\frac{n_1}{n_2}(r_\parallel + 1))^2 + (r_\perp + 1)^2} \tag{28}$$

Alternatively, Fresnel sine and tangent law could be used [4]. This looks cleaner but is more expensive to compute (section 3.4) so will not be used:

$$r_\perp = -\frac{\sin(\theta_1 - \theta_2)}{\sin(\theta_1 + \theta_2)} \quad r_\parallel = \frac{\tan(\theta_1 - \theta_2)}{\tan(\theta_1 + \theta_2)} \quad \theta_1 \neq 0 \tag{29}$$

2.6 Diffraction (Gratings)

2.7 Finding Direction

The most known diffraction equation is

$$d\sin(\theta) = m\lambda \quad d, \theta, \lambda \in \mathbb{R} \quad m \in \mathbb{Z} \tag{30}$$

Where θ is the angle between normal and diffracted ray, λ is the wavelength, d is the slit distance in the grating, and m is the order. However, the more useful equation is

$$d(\sin(\alpha) - \sin(\beta)) = m\lambda \quad \alpha, \beta, d, \lambda \in \mathbb{R} \quad m \in \mathbb{Z} \tag{31}$$

This is the same as before where $\alpha = \theta$ and β is the angle between normal and incident ray. Equation 30 is a specific case where $\beta = 0$. Letting a be the ray direction, and b be the tangent, and using equation 17 like before, $(a \cdot b) = \sin(\alpha)$, therefore:

$$\sin(\beta) = (a\cdot b) - \frac{m\lambda}{d}$$
$$\cos(\beta) = \sqrt{1 - \sin^2(\beta)} \tag{32}$$

Because β is the angle between the ray and the normal, I would rotate the tangent by $\frac{\pi}{2} - \beta$ to get the new direction. Using equation 17 for both sine and cosine, and equation 32:

$$f(a,b) = \begin{bmatrix} \sin(\beta) & -\cos(\beta) \\ \cos(\beta) & \sin(\beta) \end{bmatrix} b \tag{33}$$

Finally I need to constrain m. As $\beta \in \mathbb{R} \Rightarrow \sin(\beta) \in [-1,1] \Rightarrow m \in [\frac{d}{\lambda}(-1-(a\cdot b)), \frac{d}{\lambda}(1-(a\cdot b))]$. Like before, optimisation can be done by storing $\frac{1}{d}$ to avoid real-time division. Note depending on the direction of the tangent relative to the ray, diffraction is either transmissive or reflective.

2.8 Ray Intensity

Incident intensity is distributed between each diffraction order. Using the interference intensity equation multiplied by the single slit diffraction equation [6], where $N = 1/d$, and using equation 31:

$$\delta = \frac{2\pi}{\lambda} d \sin(\theta) = 2\pi m \tag{34}$$

$$I_{\text{out}} = I \frac{\sin^2(N\frac{1}{2}\delta)}{(\frac{1}{2}\delta)^2} = I(\frac{\sin(N\pi m)}{\pi m})^2 \tag{35}$$

3 Miscellaneous

Content is section 1 and 2 is sufficient for a functioning simulation. However, the following sections are important for better functionality.

3.1 Wavelength to RGB

3.1.1 Defining Perceived and RGB Colour

The human eye contains three types of cone, each with different spectral sensitivity. Perceived colour is the result of the combination of the response of each cone. Mathematically, $c_n|\mathbb{R} \to \mathbb{R}$ where $c_n(\lambda)$ is the cone response and λ is the wavelength. Perceived colour can then be defined as a function of spectrum (Λ, a set of wavelengths of each photon incident in a period of time):

$$p|\mathbb{R}^n \to \mathbb{R}^3 \quad p(\Lambda) = \sum_{m=0}^{n} \begin{pmatrix} c_1(\Lambda_n) \\ c_2(\Lambda_n) \\ c_3(\Lambda_n) \end{pmatrix} \tag{36}$$

This is because the cone responds to each photon hitting it, so the overall response is proportional to the sum of individual responses over a period of time. Most display technologies use RGB to create perceived colour. By choosing different intensity of red, green, and blue light any perceived colour can be created. Let $R = p(\Lambda_r), G = p(\Lambda_g), B = p(\Lambda_b)$. Provided R, G and B are not parallel(different perceived colour), any $\mathbb{R}^3$ can be represented as a linear combination of R, G and B, therefore so can any perceived colour. An RGB colour can be represented as $\mathbb{R}^3$, with the intensity of each light. Letting α be the RGB colour, and $q(\alpha)$ be the perceived colour,

$$q|\mathbb{R}^3 \to \mathbb{R}^3 \quad q(\alpha) = \begin{bmatrix} R_1 & G_1 & B_1 \\ R_2 & G_2 & B_2 \\ R_3 & G_3 & B_3 \end{bmatrix} \alpha \tag{37}$$

$$q^{-1}(\beta) = \begin{bmatrix} R_1 & G_1 & B_1 \\ R_2 & G_2 & B_2 \\ R_3 & G_3 & B_3 \end{bmatrix}^{-1} \beta \tag{38}$$

Using equation 38, RGB colour can be found from spectrum: $\alpha = q^{-1}(p(\Lambda))$. In practice this requires two things: The spectral response of each cone(c_1, c_2, c_3), and the emitted spectrum of the red, green, and blue light in a display$(\Lambda_r, \Lambda_g, \Lambda_b)$. Both of these could be found online. Alternatively, Λ_r, Λ_g and Λ_b could be found using a spectrometer. In reality, $c_n \geq 0$. This means all perceived colours lie in the positive octant. The intensity of each light in a display is also ≥ 0. This means unless R, G and B are all perpendicular(their perceived colours excite only one cone (impossible due to the cone response overlap)), it is impossible to create every perceived colour using RGB.

3.1.2 Implementation

I use the CIE-1931 standard to map wavelength to RGB. CIE maps wavelength to intensity of three imaginary perceived colours (X, Y and Z). This was achieved by first experimentally finding intensity of three real lights to create perceived colour the same as a specific wavelength (for many wavelengths), and then transforming to XYZ lights to be a better standard. Note to allow negative intensity, light could be added to the wavelength colour (essentially adding to the other side of the equation). When each intensity is plotted as a function of wavelength, it is reminiscent of a bell curve, allowing use of sums of Gaussian functions to approximate [3]:

$$g(x;\mu,\tau_1,\tau_2) = \begin{cases} \frac{1}{2}\exp(-\tau_1^2(x-\mu)^2) & x < \mu \\ \frac{1}{2}\exp(-\tau_2^2(x-\mu)^2) & x \geq \mu \end{cases} \tag{39}$$

$$\bar{x},\bar{y},\bar{z}|\mathbb{R}\to\mathbb{R}$$

$$\bar{x}(\lambda) = 1.056g(\lambda\times10^{-9};599.8,0.0264,0.0323) + 0.362g(\lambda\times10^{-9};442.0,0.0624,0.0374)$$
$$-0.065g(\lambda\times10^{-9};501.1,0.0490,0.0382)$$
$$\bar{y}(\lambda) = 0.821g(\lambda\times10^{-9};568.8,0.0213,0.0247) + 0.286g(\lambda\times10^{-9};530.9,0.0613,0.0322)$$
$$\bar{z}(\lambda) = 1.217g(\lambda\times10^{-9};437.0,0.0845,0.0278) + 0.681g(\lambda\times10^{-9};459.0,0.0385,0.0725) \tag{40}$$

XYZ can be transformed linearly to intensity of real RGB lights in a display. Most displays use sRGB standard, with a nonlinear function mapping RGB value to RGB light output. This is because the human eye perceives colour more accurately with lower intensity, so has the effect of more resolution at lower intensity. To counter this nonlinear mapping, gamma correction is used (inverse of the mapping, equation 42). Letting α be the RGB colour like before, and v be the sRGB colour [1],[2]:

$$\alpha, v \in \mathbb{R}^3$$

$$\begin{bmatrix} 3.2404542 & -1.5371385 & -0.4985314 \\ -0.9692660 & 1.8760108 & 0.0415560 \\ 0.0556434 & -0.2040259 & 1.0572252 \end{bmatrix} \begin{bmatrix} \bar{x}(\lambda) \\ \bar{y}(\lambda) \\ \bar{z}(\lambda) \end{bmatrix} = \alpha \tag{41}$$

$$v_n = \begin{cases} 12.92\alpha_n & \alpha_n \leq 0.0031308 \\ 1.055\alpha_n^{1/2.4} - 0.055 & \text{otherwise} \end{cases} \cong \alpha_n^{1/2.2} \tag{42}$$

For efficient implementation, I generate a 4096x1 image where $x = (\text{width})(\lambda - \lambda_{\min})/(\lambda_{\max} - \lambda_{\min})$, and the pixel at x is in the RGB of that wavelength. In runtime, the image is in memory and sampling is very fast. For colours to add when overlapping, I use the GPU to perform additive blending with zero overhead. After all the ray drawing, I apply a shader (processes image using GPU in parallel) to convert from RGB to sRGB over the whole screen.

3.2 Drawing Optics

It is useful to distinguish between each side of an optical element. This can be achieved by drawing two lines next to each other in different colours. This is simple for lines and arcs; The graphics library I use (Raylib) provides functionality to draw arcs and line. I can draw two lines next to each other or two arcs of different radius. Letting g be the gap and L_n be the optical element, using equation 2, L_n can be drawn again translated by $g\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\hat{b}_n$. An arc can be drawn of radius r and $r+g$. However, for parabolas this is more difficult. A parabola is drawn by evaluating a set of points on the parabola, and drawing lines to connect the points. Letting the set of points be $P \subseteq \mathbb{R}^3$, each point would be translated T: $T_n = g\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}(P_{n+1} - P_n)/\|P_{n+1} - P_n\|$. However, these vectors translate each point considering the direction of the line after it only, and not the line before it. This can look bad at a corner. The solution is to translate using a new set of points, $Q|Q_n = \begin{cases} Q_n = T_n & n = 1 \\ \frac{1}{2}(T_n + T_{n-1}) \end{cases}$. Here each point in P is translated by g times the mean of the normal of the previous line segment and the next. Alternatively, tangents could be found using equation 15.

3.3 Lenses

3.3.1 Creating Optical Elements

Lenses can be created by the intersection or subtraction of spheres from a cylinder, where the cylinder face centres and sphere centres lie in the same line; The lens is a cylinder with convex or concave spherical surfaces. Lenses can then be defined by $R_1, R_2, d, h, \theta \in \mathbb{R}$ and $C \in \mathbb{R}^2$, where R_1 and R_2 are the radius of curvature of each face, d is the thickness (distance between each face along the centre line), h is the height (diameter of the cylinder), θ is the angle and C is the centre. By sign convention of Lensmakers equation, a radius is negative if concave, therefore I let $r_n = \|R_n\|$. The height must be constrained; If the spheres intersect, it cannot exceed the height at the intersection, and if they do not(both concave), it cannot exceed the minimum of radius. If both are convex, the intersection height can be found by constructing a triangle with sides r_1, r_2 and $a = r_1 + r_2 - g$, where g is the gap along the centreline in the intersection (equation 43). a can then be simplified (equation 44):

$$g = \begin{cases} d & R_1, R_2 \geq 0 \\ 2r_1 - d & R_1 \geq 0 \wedge R_2 < 0 \\ 2r_2 - d & R_2 \geq 0 \wedge R_1 < 0 \end{cases} \tag{43}$$

$$a = \begin{cases} R_1 + R_2 - d & R_1, R_2 \geq 0 \\ -(R_1 + R_2 - d) & \text{sgn}(R_1) \neq \text{sgn}(R_2) \end{cases} \tag{44}$$

The height perpendicular to c is $h/2$. The maximum height can be found using herons formula to first find the area, then multiply by $4/a$ (equation 45). Note for meniscus lenses (when $\text{sgn}(R_1) \neq \text{sgn}(R_2)$), the true maximum height is the maximum of this and the minimum radius, as it can curve back inward. This is the same for biconvex if $d > \min(r_1, r_2)$. This ensures the faces are completely spherical. Letting $b = (r_1 + r_2 + a)/2$,

$$\alpha = \frac{4\sqrt{b(b-r_1)(b-r_2)(b-a)}}{a} \tag{45}$$

$$\beta = \min(r_1, r_2) \tag{46}$$

$$h_{max} = \begin{cases} \beta & R_1, R_2 < 0 \vee (\beta > \alpha \wedge \text{sgn}(R_1) \neq \text{sgn}(R_2)) \vee (R_1, R_2 \geq 0 \wedge d > \beta) \\ \alpha & \text{otherwise} \end{cases} \qquad h \in [0, h_{\max}] \tag{47}$$

The arc centres will need to be transformed using f. Letting circle 1 always create the face negative distance along the centre line,

$$f|\mathbb{R}^2 \to \mathbb{R}^2 | f(x) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} x + C \tag{48}$$

$$x_{c_1} = R_1 - \frac{1}{2}d \quad x_{c_2} = \frac{1}{2}d - R_2 \tag{49}$$

$$c_1 = f(\begin{pmatrix} x_{c_1} \\ 0 \end{pmatrix}) \quad c_2 = f(\begin{pmatrix} x_{c_2} \\ 0 \end{pmatrix}) \tag{50}$$

I derived equation 50 by analysis of all cases (bi-convex, bi-concave, and meniscus). Finally, the endpoints of the line slicing the arcs can be found(also the endpoints of linear lines connecting the arcs). These are found by first finding only the un-rotated x coordinate (x_1 or x_2) as the y coordinate is $\pm h$, the points are f applied to every combination of x and y. Through pythagerous (and using root signs such that the x is correct),

$$(x_n - x_{c_n})^2 + (\frac{1}{2}h)^2 = r_n^2 \tag{51}$$

$$x_1 = -\text{sgn}(R_1)\sqrt{r_1^2 - (\frac{1}{2}h)^2} + x_{c_1} \tag{52}$$

$$x_2 = \text{sgn}(R_2)\sqrt{r_2^2 - (\frac{1}{2}h)^2} + x_{c_2} \tag{53}$$

3.4 Speed of Math Functions

To effectively optimise, knowledge of how expensive each math function is comparatively is important. [5] has data on this for various CPU's. I used the data for 11th gen Intel, as it is recent. Latency is the cycles required to

execute, and throughput is the number of instructions that can be executed each cycle through pipelining. Below is a table of relevant float instructions:

Instruction	Latency	1/Throughput	Description
FADD	3	1	$+, -$
FMUL	4	1	$\times$
FDIV	14-16	4-5	$\div$
FCOM	3-4	1	$<, >, =, \leq, \geq$
FXAM	6	2	get if Nan,$\pm\infty, 0$
FPREM	27-30	26	mod,truncated quotient
FPREM1	29-60	26	mod,rounded quotient
FRNDINT	21	11	round
FSCALE	11	11	$\times 2^n \quad n \in \mathbb{Z}$
FXTRACT	11	10	split mantissa,exponent
FSQRT	14-21	4-7	$\sqrt{}$
FSIN	60-120	60-120	sin
FCOS	60-130	60-130	cos
FSINCOS	60-140	60-140	$\sin(x)$,$\cos(x)$
FPTAN	60-140	60-140	tan
FPATAN	100-160	100-160	$\tan^{-1}(x/y)$
FYL2X	50-105	50-105	$y \log_2(x)$
FYL2XP1	60-70	60-70	$y \log_2(x+1)$

References

[1] (revised 8 April 2017) Bruce Justin Lindbloom, Gamma correction function, *XYZ to RGB*
`http://www.brucelindbloom.com/index.html?Eqn_XYZ_to_RGB.html`

[2] (revised 7 April 2017) Bruce Justin Lindbloom, Transform from CIE XYZ to RGB, *RGB/XYZ Matrices*
`http://www.brucelindbloom.com/index.html?Eqn_RGB_XYZ_Matrix.html`

[3] (revised 4 August 2025) Approximations of CIE XYZ from wavelength, *CIE 1931 color space*
`https://en.wikipedia.org/wiki/CIE_1931_color_space#Analytical_approximation`

[4] (revised 29 July 2025) Fresnel equations, *Fresnel equations*
`https://en.wikipedia.org/wiki/Fresnel_equations`

[5] (revised 25 July 2025) Agner Fog, Speed of various math functions, page 364, *4. Instruction tables*
`https://www.agner.org/optimise/instruction_tables.pdf`

[6] (visited 10 August 2025) R Nave, Intensity of diffraction orders, *Grating Intensity*
`http://hyperphysics.phy-astr.gsu.edu/hbase/phyopt/gratint.html#c2`