Fresnel Reflectance, Anisotropic Absorption, and Polarization in Rendering, to Show Crystallographic Effects

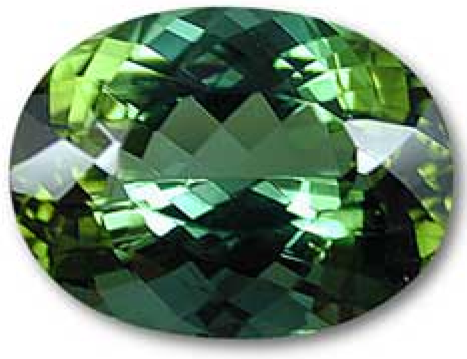
Keith Primdahl

Presented at the CS348B, Image Synthesis Symposium, Stanford University, Spring 2005

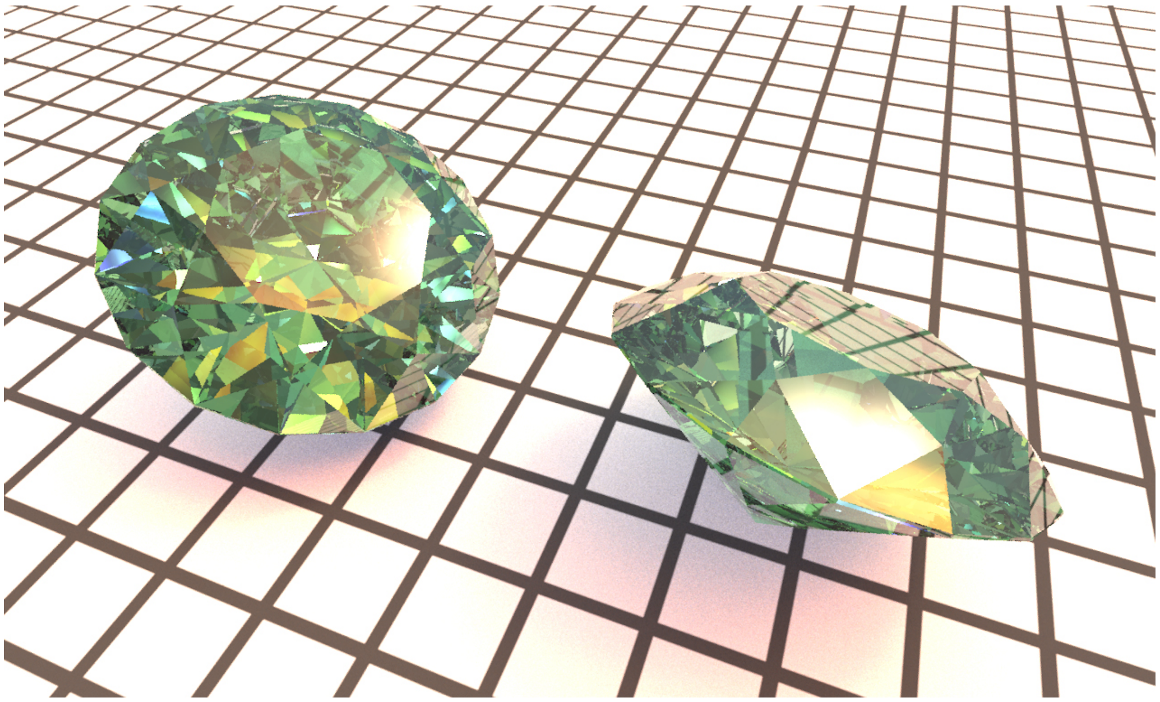
Abstract

A project which incrementally extends pbrt is described. Example scenes showing Fresnel reflection, isotropic absorption, and pleochroism in the gemstone tourmaline are presented. Tourmaline is typical for absorption characteristics in crystals, characteristics which often vary with wavelength and direction. Interesting color and intensity variation results from splitting incident rays, by polarization and birefringence, into ordinary and extra-ordinary rays/directions with absorption a function of wavelength and ray direction with respect to the crystalline axis.

Goal



Result



1 Motivation and Objective

For the past several years, the author has been fortunate to work with some of the world's eminent laser scientists. While perhaps less well-known in computer graphics, crystal optics—and their unique properties with respect to light—are common in the laser community. Figures 1 and 2 show two examples of laser equipment with active crystal elements developed at Lawrence Livermore National Laboratory.[1]

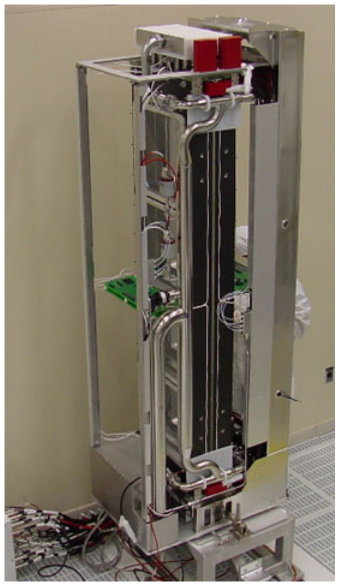
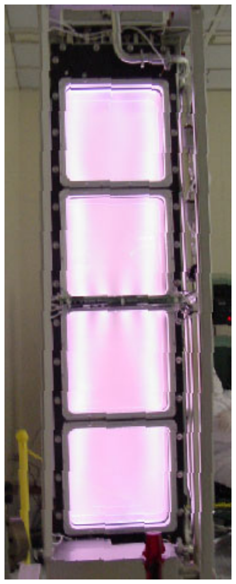



Figure 1, Plasma Electrode Pockels Cell for four 40 cm x 40 cm pulsed infrared lasers of approximately 260 GW per aperture. Using a plasma created on both sides of a Potassium-Di-hydrogen-Phosphate (KDP) crystal optic, we set an electric field across the crystal. The plasma is transparent to infrared wavelengths. Orientation of the laser beam's linear polarization is rotated 90 degrees by the crystal [1].[2]

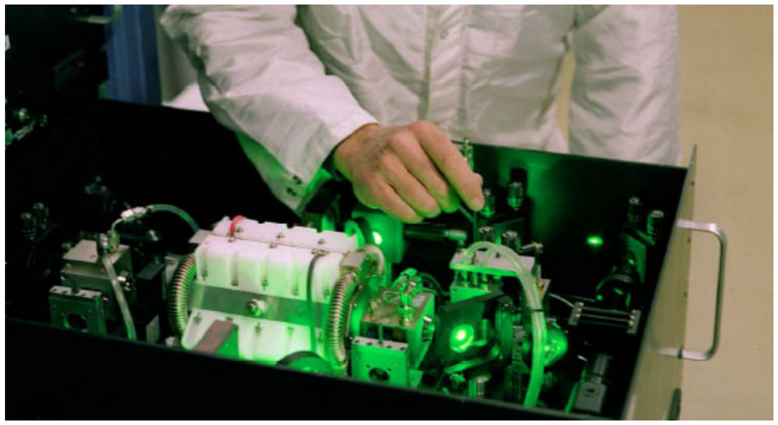


Figure 2, Neodymium, doped in Yttrium-Aluminum-Garnet (YAG) crystal, is the lasing medium where the infrared output is converted to green (450 W continuous) by a Lithium-Triborate (LBO) crystal [4].

[1] Papers by McClain, et.al. [14][15] are particularly interesting in that they make a distinction between non-optically active and optically active crystal materials.

[2] For an overview of the National Ignition Facility (NIF) project at Lawrence Livermore National Laboratory: http://www.llnl.gov/nif/index.html

While crystals are a common design element in many laser systems, and Beyerle and McDermid [3] cite laser harmonic generators (frequency converters, as in Figure 2) and polarizing prisms as anisotropic crystal applications worthy of rendering development. However, neither polarization nor birefringence is addressed in pbrt—undoubtedly, because these phenomena are generally not considered "visible." Yet, Minnaret describes Haidinger's Brush, an example of visible polarization in the twilight sky [16]; albeit, a phenomenon probably not suitable for rendering. Können provides further examples of naturally occurring polarization; in particular, he describes colorful effects which can be obtained by careful observation, with filters, of double refraction in birefringent crystals (p. 156-7) [11].

A more "mainstream" manifestation of crystalline optical effects can be found in crystal gemstones. Many crystalline materials are birefringent; i.e., they will split an optical ray by dual refraction. For our purposes, crystals can be distinguished as *uniaxial* or *biaxial*, to denote the presence of one crystal axis or two. Incident light which is parallel to a crystal axis will behave the same as for non-crystalline dielectrics; e.g., following Snell's law, etc. Incident light approaching from all other directions will, at the incident plane, be split by polarization into two directions. The first of these directions will be in accordance with Snell's law; hence, is known as *ordinary*. The second of these directions is determined by a variable index of refraction. Because this behavior is not ordinary, it is known as *extra-ordinary* [17]. Published values are available for the index of refraction for these rays when the incident light is perpendicular to the crystal axis. From this value when perpendicular, the extra-ordinary index of refraction varies with the angle until matching the ordinary index at the crystal axis. Because the ordinary and extra-ordinary rays have different polarizations, they are absorbed differently by the crystal. Depending upon the angle with which a ray enters a crystal, the extra-ordinary ray will have variable direction; consequently, its absorption will vary—the resulting variation in crystal gemstone color is known as *pleochroism* (many colors) [8] [9]. A crystal displaying this characteristic is known as *dichroic*. For biaxial crystals, one must keep track of three indices of refraction, and ray angle with respect to two crystalline axes.

An extension of pbrt to more correctly render crystal properties does not appear to have been previously addressed; moreover, is relevant to the author's work (actually, the associated study of crystal effects alone is worthwhile). To correctly model the dual-refractive indices of birefringent materials, both of which vary with wavelength, further requires implementation of dispersion, at least for the general case.[3] For the tourmaline example which follows, strict modeling of wavelength-dependent refractions may not be necessary, as all but a narrow spectrum of wavelengths face considerable absorption. The ultimate objective is the rendering of images similar to the photographs of Figure 3.

[3] The author had separately reasoned out much of the traditional forward ray-tracing approach presented by Sun et.al. [21]. Bennett and Amezcua successfully used a wavelength dependence defined in 5-nm increments, for a total of 95 defined wavelengths; albeit, with the goal of modeling thin-film interference [2]. While their implementation was limited to color representation within their BRDF, the objective here is similarly to achieve a continuous spectrum for wavelength-dependent effects. However, Wilkie, et.al. show examples using only 8 regularly-spaced samples, along with a jittering method by which the otherwise-unacceptable color aliasing can be made tolerable [24]. Sun also points out that dispersive color aliasing is possible, and further asserts that the number for samples should not be less than for sampling spectral functions [21]; which still leaves significant room for good judgment. For true dispersion, rays generated at each incremental wavelength will propagate throughout the dispersive geometry; however, the full spectrum must ultimately be converted back to RGB; so, probably not so many incremental wavelengths are necessary (Spectrum class provides a method to convert Spectrum of any number of samples into XYZ, and then from XYZ to RGB; these methods are called by the ImageFilm class prior to WriteImage; ref CS348b text p.378 [19]).

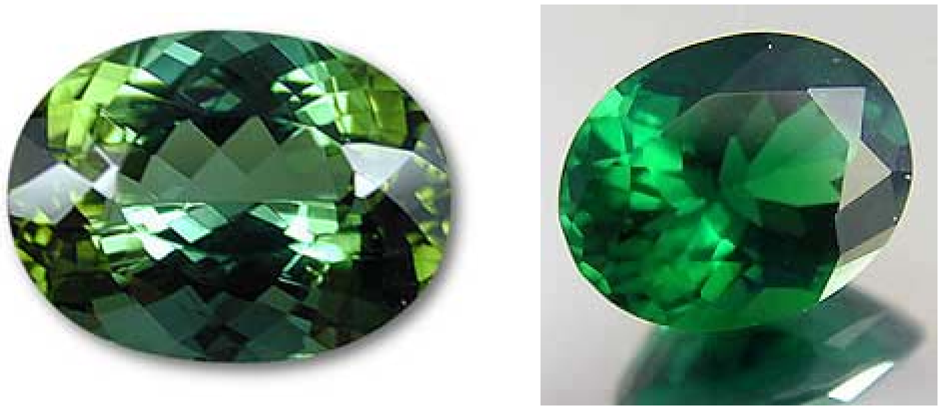


Figure 3, Left: Green tourmaline, courtesy of www.palagems.com. Right: Chrome tourmaline, courtesy of www.gemfix.com. In both cases, the white-highlight facets are reflections of illumination sources, color elsewhere results from [anisotropic] bulk absorption, while the variance in color intensity is due to polarization and differential absorption. Notice, especially in the left image, the color tendency towards blue or yellow in specific directions.

2 Approach

2.1 Fresnel Reflectance

Fresnel reflective is already available in pbrt; moreover, is incorporated into the glass material plug-in**.** So, the task here was merely to incorporate into a scene, and demonstrate the effect.

2.2 Absorption

Sun et.al. [20] model absorption in diamonds after the Bouger-Lambertian law, as does our CS348b text [19]. Guy and Soler provide a derivation of RGB absorption coefficients for tourmaline and a few other crystals [8]. Specifically, they study measured absorption spectra, then determine RGB absorption coefficients by a linear approximation, which they further show to be valid over small distances.

The task here was to create and implement a simple approach for attenuation; specifically, less complex than a pbrt VolumeIntegrator, as emission, etc. is not of interest. Nevertheless, creation of a VolumeRegion and use of a VolumeIntegrator was considered, but deemed not feasible due to the difficulty of defining the volume (as noted later, capability to model scatter may have been more valuable then expected). The selected approach was to apply Bouger-Lambertian separately to red, green, and blue for the distance of the ray between incident and excitant intersections, using coefficients provided by Guy & Soler; that is, to incorporate absorption as an additional dimension over which to integrate the Light Transport Equation. A new Material plug-in, specifically a Crystal class, was developed, with its data structure expanded for extra-ordinary index of refraction, two sets of RGB absorption coefficients, and a new method for determination of RGB absorption values based on ray direction and absorption length within the crystal. The implementation requires that crystals be modeled with their optical axis parallel to their local X axis.

2.3 Polarization and Birefringence

Glassner [7], Tannenbaum, et.al. [22], Wolf and Kurlander [25] and Wilke et.al. [23] all deal with polarization by the use of coherency matrices (CMs), generally with coherency modification matrices (CMMs) used to describe intersection with polarizing surfaces. Wilke et.al. [23] further describes modeling based on Stokes parameters, which offer the advantages of using only real-valued terms to describe all polarization states, and use of Müller matrices for ray weights (also real valued). Furthermore, the first of the four vectors in the Stokes notation is the unpolarized intensity of the ray. Delvin provides a comprehensive survey of spectral rendering [6].

The selected implementation uses a coherency matrix to track the polarization state of a ray, and modification matrices used to update the polarization state at each intersection. The resulting approach is close to a software-only version of the hardware-oriented implementation Guy and Soler [8]. In particular, Guy & Soler show that only slight error is introduced by treating all intersections as between isotropic materials (the incident air, of course, is isotropic), and that the optical axis corresponds to the surface normal—for this case, calculation for the Fresnel coefficients is greatly simplified. So, while the data structure is provided for calculation of the extra-ordinary index of refraction, and the angle between the crystal axis and ray incidence is calculated for anisotropic absorption, all refractions are actually treated as ordinary.

Polarization, one can think of the common textbook example where a single ray of light encountering a dielectric surface at Brewster's angle is split into two rays, a reflected ray of *perpendicular,* or *s* polarization, and a transmitted ray of *parallel,* or *p* polarization, where perpendicular and parallel refer to the direction of electric field oscillation with respect to the plane of incidence-reflection. The general case is, unfortunately, a bit more complicated. The polarization state can be expressed as a 2 x 2 matrix of complex elements in the ray's local coordinate system. For intersections among dielectric materials, the imaginary terms vanish, but cross-product terms remain for states of partial polarization. In this polarization-state matrix, the current radiance is represented by the trace; hence, maintenance of the polarization-state tracks the cumulative energy loss as well.

$$J == \begin{bmatrix} J_{xx} & J_{xy} \\ J_{yx} & J_{yy} \end{bmatrix}$$

If one were able of start with perfectly incoherent light (rarely the actual case, as light passing through the earth's atmosphere is partially polarized to varying degrees), the polarization could be expressed as:

$$J_{incoherent} = \frac{1}{2} L_0 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

With L_0 representing the initial radiance for the ray.

Upon the ray's first intersection, it will generally split into transmitted and reflected rays, each with a partial state of polarization. At each successive intersection, these partially-polarized rays further split with further change to the their polarization states. Modification matrices for each intersection are necessary:

$$M^t \approx R_{\theta i} \begin{bmatrix} F_p^t & 0 \\ 0 & F_s^t \end{bmatrix} R_{\theta t} \qquad M^r \approx R_{\theta i} \begin{bmatrix} F_p^r & 0 \\ 0 & F_s^r \end{bmatrix} R_{\theta r}$$

Where r and t denote reflection and transmission, respectively while ≈ denotes the simplification. Three rotation matrices allow for calculation of Fresnel coefficients always in a coordinate system defined by the surface and its normal; hence, $R_{\theta i}$, $R_{\theta t}$, and $R_{\theta r}$ are for rotation of the surface normal to the direction of incidence, transmission, and reflection respectively:

$$R_\theta = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

Finally, the polarization-state is updated for each step (intersection) of the ray:

$$J_k = M_k J_{k-1} M_k^T$$

Where M is the modification matrix associated with the reflective or transmissive ray selected for subsequent propagation, presently at its k^{th} intersection.

The Fresnel coefficients in the above can be found by [17]:

$$F_r^s = \frac{\cos\theta - (n_2/n_1)\sqrt{1-[(n_2/n_1)\sin\theta]^2}}{\cos\theta + (n_2/n_1)\sqrt{1-[(n_2/n_1)\sin\theta]^2}}$$

$$F_r^p = \frac{-(n_2/n_1)\cos\theta + \sqrt{1-[(n_2/n_1)\sin\theta]^2}}{(n_2/n_1)\cos\theta + \sqrt{1-[(n_2/n_1)\sin\theta]^2}}$$

$$F_t^s = \frac{2\cos\theta}{\cos\theta + (n_2/n_1)\sqrt{1-[(n_2/n_1)\sin\theta]^2}}$$

$$F_t^p = \frac{2\cos\theta}{(n_2/n_1)\cos\theta + \sqrt{1-[(n_2/n_1)\sin\theta]^2}}$$

Here, θ denotes the angle between the surface normal and the direction of incidence, with η_1 and η_2 the indices for refraction on either side of the intersected surface where the light is traveling from a medium "1", and intersecting a medium "2."

2.4 Rendering Strategy

While it might seem that photon mapping is ideal for rendering of crystals, the faceted shapes of crystal gemstones are meant to minimize "leakage" of light. Since the scene modeled here has diffuse surfaces only in the direction where caustics are prevented by the gemstones' design, there would be little benefit to using photon mapping.

A polarization-tracking integrator was developed, with additions for handling anisotropic absorption. The plug-in was modeled from the Direct Lighting integrator provided in pbrt. "Direct Lighting" is somewhat of a misnomer, as this integrator actually does track specular reflections to the recursive depth requested (hence, "direct lighting" only for diffuse surfaces). While it would be natural to add data member to the ray class for tracking polarization state, doing so would mean all pbrt rendering would be impacted by a larger Ray data structure. So, the polarization state for each ray is tracked within the expanded surface-integrator's data structure (similar to data member *rayDepth*). Provisions were made to similarly track whether the ray has encountered a birefringent material, and if so, whether the ray is designated *ordinary* or *extra-ordinary*.

Since the basic Direct Lighting integrator terminates each ray at its first intersection with a diffuse surface, it neatly avoids significant branching of rays. Incorporation of polarization and birefringence, with a ray split at each specular intersection, could quickly become unmanageable. The author's original intention for each such split was to randomly select which single ray to propagate, then adjust its radiance by the probability of its being selected, thereby ensuring an unbiased result (i.e., an inverted approach to the Russian Roulette technique of adjusting for the probability of being terminated). Unfortunately, the associated implementation would require a complete rewrite of the surface integrator code being modified—better to leave this for future efforts.

2.5 Limitations

Models of oval-shaped faceted gemstones were obtained in both .dxf and .obj format. Unfortunately, a conversion path from .dxf to .pbrt was never found. The .obj model was mostly quad elements; moreover, quad elements without a consistent vertex-callout pattern; accordingly, this model would load using Mark Colbert's Wavefront Shape converter (ref. the downloads page on the web site for the CS348b text [19]); however, "holes" remained even after several attempts to renumber the quads into triangle pairs. So, rendering was limited to the diamond model presented, which provides at least two limitations:

1) To maximize internal reflection back towards the viewer's most likely direction, crystal gemstones' facet design is based on each selected crystal material's index of refraction. An early study of this effect, with results provided in figure 4, confirmed that it would be wiser to use diamond's index of refraction (2.4), rather than tourmaline's (1.6).

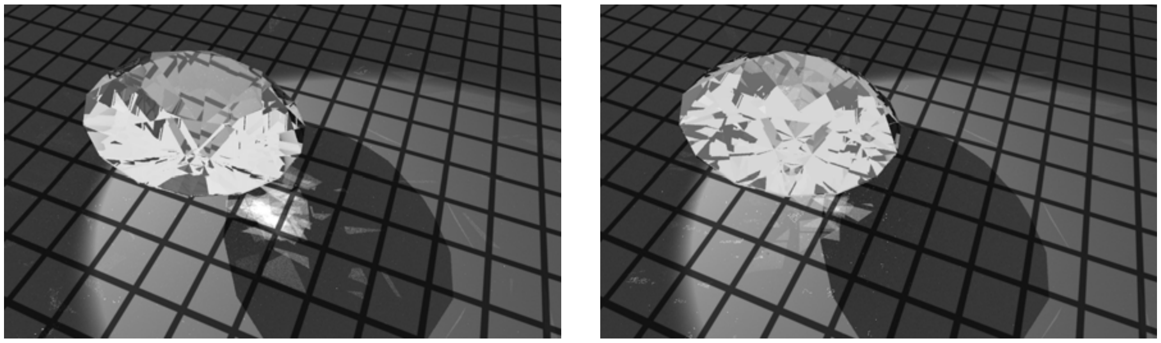
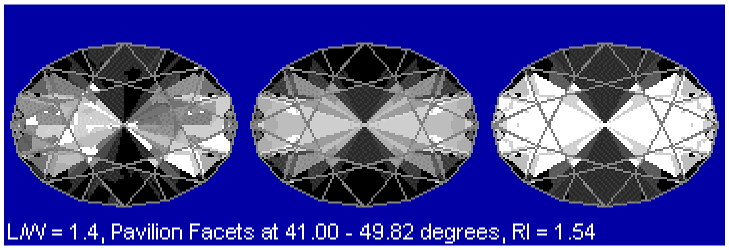


Figure 4, Diamond brilliant-cut style rendered with pbrt's Photon Integrator. Left: with miss-matched index of refraction. Right: with matched index of refraction. Both images created early in scene development, prior to incorporation of Fresnel reflection and finalization of lighting.

2) As a crystal gemstone design is elongated from the diamond "brilliant-cut" style to an oval, gemstone designers encounter an effect know as "bow-tie," where the optimum facet angle cannot simultaneously be cut in the wide and narrow radii of the crystal. Generally the facets of the larger radius are sub-optimum; hence, leak light. This results in darkening towards the wide radii. While techniques are available to minimize this effect [11], it cannot be entirely eliminated, and is in fact visible in figure 3.


L/W = 1.4, Pavilion Facets at 41.00 - 49.82 degrees, RI = 1.54
Figure 5, Darkening due to light leakage through back-side facets. For oval gemstones, it is not possible to match the index of refraction at all the back-side, or *pavilion*, facets. Image courtesy of http://www.rockhounds.com

3 Results

In addition to showing the multi-reflective properties of gemstones, examples showcasing Fresnel reflection and anisotropic absorption were successfully rendered, with the anisotropic absorption approach based on a model of RGB absorption for birefringent splitting by polarization.

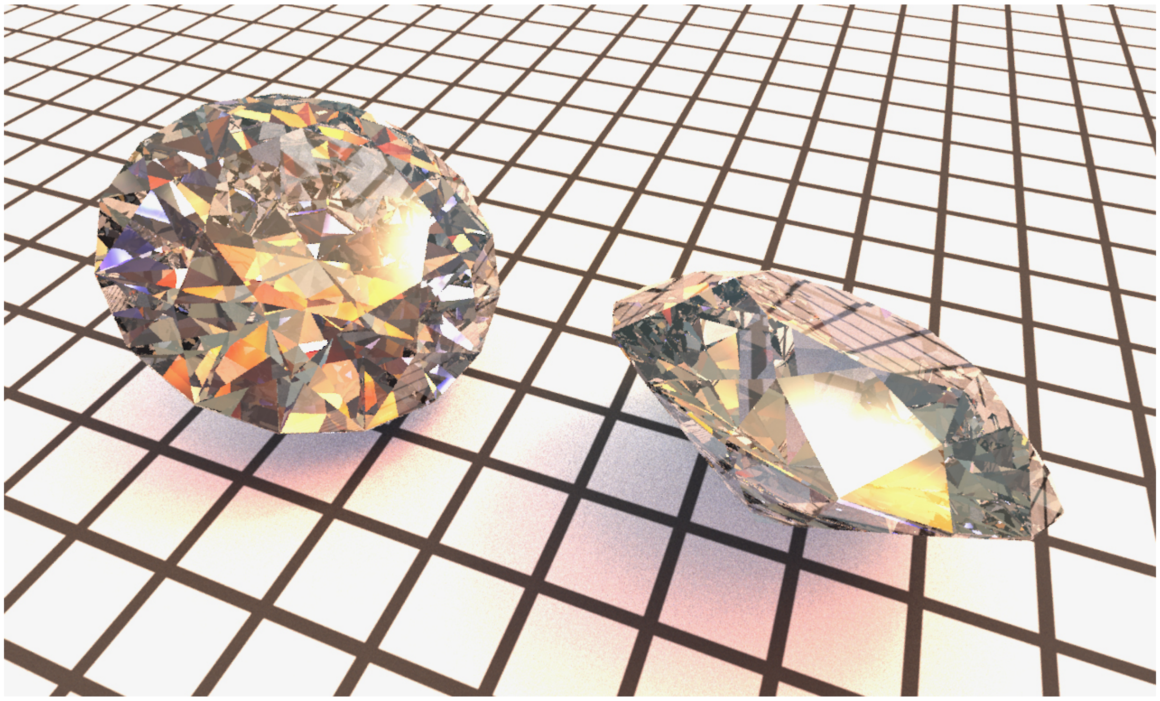


Figure 6, Fresnel reflection without absorption. Note the reflected grid on the top facet of the right gemstone, while the left gemstone show interior reflections through its top facet.

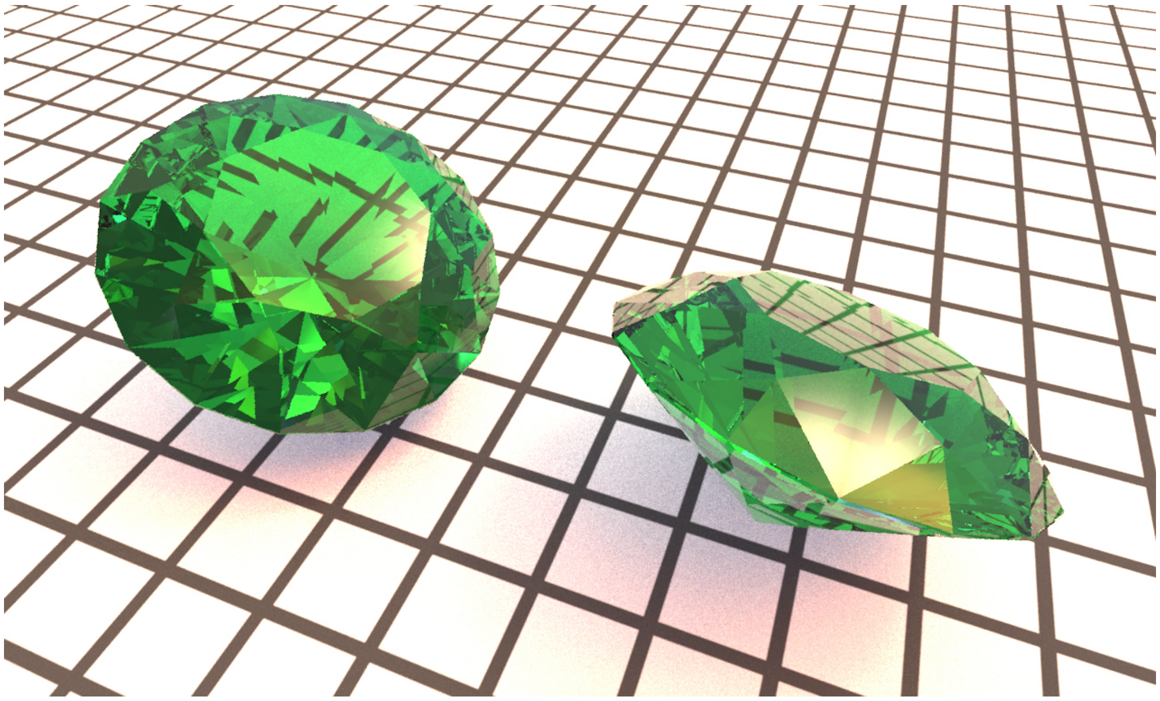


Figure 7, Constant-value absorption at each surface. Surprisingly, this does show depth by deepening of the color; i.e., less radiance to transmit at each bounce.

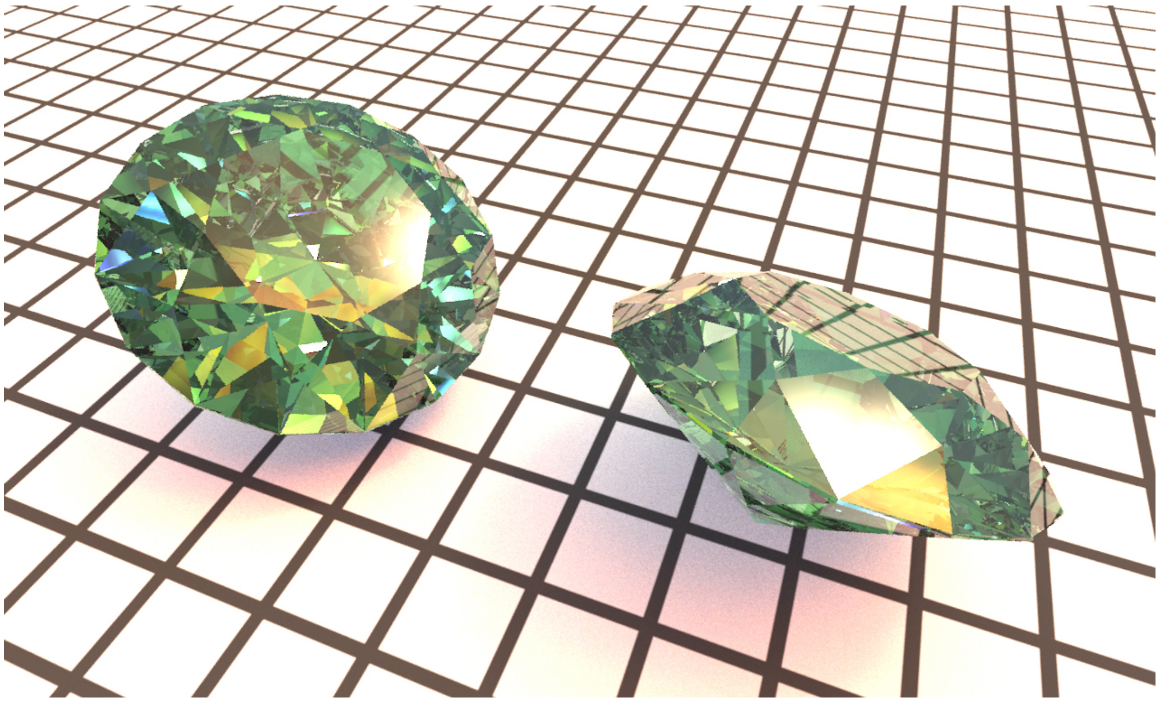


Figure 8, Anisotropic absorption. For both gemstones, the crystal axis is left-right along grid lines. The tendency toward blue along the crystal axis, towards yellow perpendicular to the crystal axis is observable in the left gemstone.

Given the evolved approach to implementing polarization, further scene development would be necessary to show visible manifestations of polarization. For example, use of polarization filters as described by Können [12] and Minnaert [16], or reflections in a rain-slick street for a subtle effect.

4 Future Work

Before proceeding further, models of oval crystals, correctly faceted for tourmaline—and possibly other crystal gemstones—should be obtained.

- Addition of scatter to the anisotropic absorption, to blend the edges of facet reflections; i.e., blend the color shifts towards blue and yellow in tourmaline, as distinct edges to these colorizations are not realistic.
- Implementation of polarization-state tracking, as described earlier, possibly with the described approach to handling ray splits without biasing the Monte Carlo results.
- Once polarization is in place, generation of perfectly polarized light, and/or filtering out rays of select polarization orientation would be straight-forward. Accordingly, interesting visualizations of polarization effects could readily be modeled. Können provides a number of examples [12].
- Dispersion may be useful for rendering of crystallographic effects.
- As mentioned earlier, crystals often have an active role in lasers. While small-aperture Pockels cells, Faraday rotators, and waveplates, to name a few examples are readily available, it is not clear whether the associated polarization effects are easily modeled in commercial ray-tracing packages for optics design.

5 Acknowledgments

The author gratefully acknowledges the assistance provided by Professor Pat Hanrahan and Ren Ng, both of Stanford University. Also, his appreciation for the interesting and comprehensive presentation of physically-based rendering this past quarter.

6 References

[1] P. Arnold, C. Ollis, A. Hinz, C. Robb, K. Primdahl, J. Watson, M. O'Brien, W. Funkhouser, P. Biltoft, R. Shelton, W. Tapley, and W. DeHope: "Deployment, Commissioning, and Operation of Plasma Electrode Pockels Cells in the National Ignition Facility." *Proceedings of the SPIE Photonics West*, Society of Photonics and Instrumentation Engineers, San Jose, CA, 2004.

[2] S. Bennett and A. Amezcua: "Simulating interference effects in LRT: iridescence in biological structures." *CS348 Final Projects*, Stanford University, 2001. http://www-graphics.stanford.edu/courses/cs348b-competition/cs348b-01/iridescent_butterflies/morpho.html

[3] G. Beyerle and I. McDermid: "Ray tracing formulas for refraction and internal reflection in uniaxial crystals." *Applied Optics*, 1998.

[4] J. Chang, E. Dragon, C. Ebbers, I. Bass, C. Cochran: *An Efficient Diode-Pumped Nd:YAG Laser with 451 W of CW IR and 182 W of Pulsed Green Output.* Technical Report, UCRL-JC-129421, Lawrence Livermore National Laboratory. 1998.

[5] R. Chipman and A. Russell: "Mechanics of polarization ray tracing." *Journal of Optical Engineering,* 1995.

[6] K Devlin, A Chalmers, A Wilkie, and W Purgathofer: "Tone reproduction and physically based spectral rendering." *Eurographics 2002: State of the Art Reports*, 2002.

[7] A. Glassner, *Principles of Digital Image Synthesis*, Morgan Kaufmann Publishers Inc., San Francisco, CA, 1994.

[8] S. Guy and C. Soler: "Graphics gems revisited: fast and physically-based rendering of gemstones." *ACM Transactions on Graphics (TOG),* ACM Press, New York, NY, 2004.

[9] C. Hurlbut and R. Kammerling: *Gemology*, 2nd edition, Wiley Interscience, Hoboken, NJ 1991.

[10] H.W. Jensen *Realistic Image Synthesis Using Photon Mapping* A.K. Peters, 2001.

[11] R. Keller: *Getting rid of Those Bow Tie Blues – Part I*, Http://www.rockhounds.com

[12] G. Können: *Polarized Light in Nature. Cambridge University Press*, 1985.

[13] S. Kunz: *A Brief Review of Gemstone Optical Properties From a Lapidary's Perspective*. July 2000. http://www.cigem.ca/pdf/sandrine.pdf

[14] S. McClain, L. Hillman, and R. Chipman: "Polarization ray tracing in anisotropic optically active media, I. Algorithms." *Journal of the Optical Society of America*. Optical Society of America, 1993.

[15] S. McClain, L. Hillman, and R. Chipman: "Polarization ray tracing in anisotropic optically active media, II. Theory and physics." *Journal of the Optical Society of America*. Optical Society of America, 1993.

[16] M. Minnaert: *Light and Color in the Outdoors*. Springer-Verlag, New York, NY, 1993.

[17] K.D. Möller: *Optics*. University Science Books, Mill Valley, CA 1988.

[18] S. Peercy and S. Mark: "Linear color representations for full speed spectral rendering." *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive techniques,* September 1993.

[19] M. Pharr and G. Humphreys: *Physically Based Rendering: From Theory to Implementation,* Morgan-Kaufmann Publishers, 2004. http://www.pbrt.org/

[20] Y. Sun, F. Fracchia, and M. Drew: "Rendering diamonds." *Proceedings of the 11th Western Computer Graphics Symposium*, WCGS, 2000.

[21] Y. Sun, F. Fracchia, and M. Drew: "Rendering light dispersion with a composite spectral model." *International Conference on Color in Graphics and Image Processing*, 2000.

[22] D. Tannenbaum, C. Tannenbaum, and M. Wonzy: "Polarization and birefringency considerations in rendering." *Computer Graphics, SIGGRAPH 1994 Proceedings*, ACM Press, New York, NY 1994.

[23] A. Wilkie, R. Tobler, and W. Purgathofer: "Combined Rendering of polarization and fluorescence effects." *Proceedings of the Eurographics Workshop on Rendering*, London, England, June 2001.

[24] A. Wilkie, R. Tobler, and W. Purgathofer: "Raytracing of dispersion effects in transparent materials." WSCG 2000 Conference Proceedings, February 2000.

[25] L. Wolff and D. Kurlander: "Ray tracing with polarization parameters." *IEEE Computer Graphics and Applications,* 1990.

Appendix, pbrt Implementation Details

Development and rendering were performed on Windows PCs, in Microsoft Visual.Net.

The diamond model was commercially-purchased .obj model, brought into pbrt using Mark Colbert's Wavefront Shape converter, which is a pbrt plug-in (ref. downloads on web site for the CS348B text [19]). The model initially contained normals as well, which triggered smoothing of the facets by pbrt! If not for the experience of assignment #1, this might have been a real difficulty.

The importance-sampled infinite light developed for assignment three was used, with the grace and sky-pollution maps. In addition, a small amount of additional white light was added with an infinite light source.

Unfortunately, crystal effects could not be implemented merely by introduction of new pbrt plug-ins. Accordingly, careful modification of pbrt core code was required. Most of the work is in reflection.h/.cpp and directpolarization.cpp. The bullet lists provided here were developed as my "project plan." Hence these tasks were modified as difficulties were encountered; nevertheless, they are left in "plan" form (i.e., not in past tense, as is now generally the case; moreover, probably not comprehensive).

Input Files
- gem.pbrt
- Jewels_diamond.obj

Files created
- crystal.cpp, crystal.vcproj
- directpolarization.cpp, directpolarization.vcproj

Files modified
- pbrt.h
- color.h
- primitive.h
- material.h
- reflection.h/.cpp

pbrt.h
- Add forward declarations for class Crystal and class DirectPolarization

color.h
- Remove the friend qualifier on `Spectrum::Exp()`.

primitive.h
- Add `bool isGeometric();` because dynamic_cast would not work in directpolarization (maybe because it's a `const Primitive *primitive` in Intersection (and deleting the const was clearly having a cascading effect of required core changes).
- Add methods `getMaterial()` and `getShape()` to access private data members.

material.h

- Add method `virtual bool isBirefringent() const { return false; }`
- Add method `virtual Spectrum Absorption(const Ray &ray) { return; }`
- Add overloaded method `virtual Spectrum Absorption(const Ray &ray, const Vector opticalAxis) { return; }`

reflection.h/.cpp

- Add float *eo_eta_i* and *eo_eta_t* to FresnelDielectric class.
- Create new FresnelDielectric constructor, with float *eo_eta_i* and float *eo_eta_t*, edit old FresnelDielectric constructor for default *eo_etai* and *eo_etat* values.
- Add overloaded version of `SpecularReflection::Sample_f()`, adding `const RayDiffferential &ray, const Intersection &, float *M`, and new code to calculate the Fresnel coefficients and load the modifier matrix; i.e., M^r. Eliminate call to `FresnelDielectric::Evaluate()`. Include test for total internal reflection. Do not use FrDiel().
- Add overloaded version of `SpecularTransmission::Sample_f()`, adding `const RayDiffferential &ray, const Intersection &, float *M`, and new code to calculate the Fresnel coefficients and load the modifier matrix; i.e., M^t. Eliminate call to FresnelDielectric::Evaluate(). Include test for total internal reflection. Do not use FrDiel().
- Leave `FresnelDielectric::Evaluate()` unchanged (but also unused by the overloaded `SpecularReflection::Sample_f()` and `SpecularTransmission::Sample_f()`.
- Add overloaded versions of both `BSDF::Sample_f()`, adding another arg `const RayDiffferential &ray, const Intersection &isect, float *M`, and just executing the same code, but adding `if(bxdf->BSDFType == SPECULAR)`, if true, then call the overloaded `bxdf::Sample_f()`.
- Add overloaded method `virtual Spectrum BxDF::Sample_F()` with additional args `const RayDiffferential &ray, const Intersection &isect, float *M`. In BxDF, this will merely call the basic `BxDF::Sample_f()`.
- Add utility functions to calculate and return the four Fresnel coefficients.
- No change to FresnelConductor class.
- No API stuff here.

Create crystal.cpp (a new material plug-in, code copied/edited from glass.cpp)

- Add float *eo_index.*
- Add *eo_index* to constructor with default value in `CreateMaterial()`.
- Add method `bool isBirefringent() const { return true; }`
- Add absorption coefficients; i.e., the three Spectrum *ko* and *ke* values
- Add *kappa_o* and *kappa_eo* to constructor with default value in `CreateMaterial()`.
- Add method `Spectrum Absorption(const Ray &ray)`; find the start-to-intersection distance of the ray, then use `Spectrum::Exp()`, passing the product of the distance and the absorption coefficients (*kappa_o*). The method will return a Spectrum with 0-to-1 absorption values. Results from this method for future use, or possibly for comparison images.
- Add overloaded method `Spectrum Absorption(const Ray &ray, const Transform object2world)`; find the start-to-intersection distance of the ray, then use

`Spectrum::Exp()`, passing the product of the distance and the weighted absorption coefficients. The method return a Spectrum with 0-to-1 absorption values.

- Add comment: // Crystal-axis orientation assumed along object's x-axis. (Note: the "Z" axis might be a better convention, but the models I have are with Z along the axis of symmetry; hence, not too good for showing the desired effects).
- In `GetBSDF()`, add loading of second index of refraction to creation of BSDF; i.e., add as additional arg to the arg-call to FresnelDielectric constructor.

Create directpolarization, as a plug-in integrator, copying/editing code from directlighting.cpp

- Okay to use all the same class, data, and method names, as this will never be invoked at the same time as the basic directlighting.
- No new API stuff here.
- Add a data member bool *isPolarized* to the DirectLighting class, for use in `Li()`. This will set for propagation of each ray, set to true upon first intersection with a polarizing surface (implementation detail: for now treat only birefringent materials as polarizing, but set-up for more general application).
- Create data member float *J*[4] to track the polarization state of each ray through its successive bounces. Add initialization to identity in the constructor.
- Create utility function, for calculation of a rotation matrices, $R_{\theta i}$, $R_{\theta r}$, and $R_{\theta t}$, which rotate the incident, reflective and transmissive rays into a coordinate system defined by the surface intersected and that surface's normal.
- Create utility function for multiplying and transforming the polarization-state modification matrices.

DirectLighting::Li():

- Add test for if the incident ray is from within the crystal; i.e., check ray direction and normal orientation; this is just `Dot(Ray.d, w0) < 0` (unless inputting triangles produced elsewhere messes up the pbrt convention).
- `Li()` has an Intersection object for each intersection. Intersection objects have a Primitive data member. Add dynamic type cast to test if Primitive is a GeometricPrimitive. Note: compiler would not allow the dynamic_cast: `GeometricPrimitive *gPrimitive = dynamic_cast<GeometricPrimitive *>(isect.primitive);` so, added `bool isGeometric()` to Primitive and GeometricPrimitive classes.
- GeometricPrimitive objects have *material* as a data member. Add, conditionals as necessary to the GeometricPrimitive's `isBirefringent()`; to determine whether to call the absorption method on the material (not strictly necessary, as an absorption method is also added to the base Material class, which simply returns).
- Modify calls to `BSDF::Sample_f()`; i.e., for specular reflection and specular transmission, to include additional arg `const RayDiffferential &ray, const Intersection &, float *M`, for call to new overloaded version. Add conditionals for which method to call.
- Add code to rotate the modification matrix into world coordinates; separate cases for reflected and transmitted rays; i.e., get R, then find RMR.
- Add test `if (rayDepth == 0)`; if ture, set isPolarization to false, reinitialize J.
- Upon return of call to `BSDF::Sample_f()`, find the $J = MJM^T$ matrix product and modify the Spectrum *L* value accordingly.