Groovy Graphics Assignments: Ray-traced Transmission

Andrew T. Duchowski
Clemson University
Visual Computing
duchowski@siggraph.org

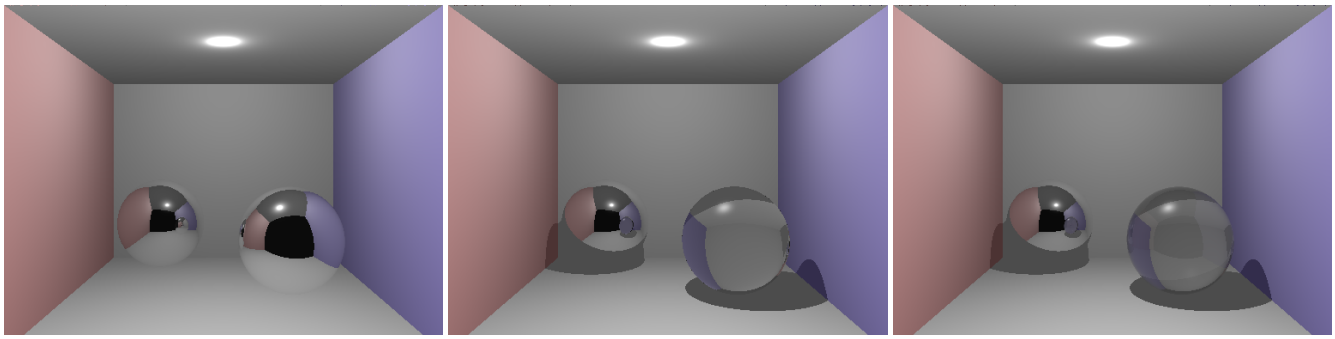

Figure 1: Example ray-traced images without and with transmission ($n = 1.805$), and with Schlick's approximation, respectively.

ABSTRACT

This groovy graphics assignment introduces transmissive rays to a basic reflective ray tracer. The assignment is groovy because it introduces transparent objects, application of Snell's Law, and allows for testing of advanced variations (e.g., Schlick's approximation).

CCS CONCEPTS

• **Computing methodologies** → Rendering; Image manipulation; Graphics systems and interfaces;

KEYWORDS

Ray tracer, refraction, transmission

ACM Reference Format:
Andrew T. Duchowski. 2018. Groovy Graphics Assignments: Ray-traced Transmission. In *Proceedings of SIGGRAPH '18 Educator's Forum.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3215641.3215651

1 OVERVIEW

In this assignment, the student is asked to introduce a transmission ray to their basic reflective ray tracer (see Figure 1).

Transmission rays implement refraction through transparent objects as governed by Snell's Law, $n \sin\theta = n_t \sin\phi$, where n and n_t are the indeces of refraction (w.r.t. vacuum) of the surrounding medium and transparent object, respectively. An index of refraction of 1.000293 models air while 1.805 models flint glass (71% lead).

SIGGRAPH '18 Educator's Forum, August 12-16, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5884-2/18/08.
https://doi.org/10.1145/3215641.3215651

As derived by Shirley [2002], the transmission ray's direction is

$$\mathbf{t} = \frac{n}{n_t}(\mathbf{u} - (\mathbf{u}\cdot\mathbf{n})\mathbf{n}) - \mathbf{n}\sqrt{1 - \left(\frac{n}{n_t}\right)^2 (1 - (\mathbf{u}\cdot\mathbf{n})^2)} \tag{1}$$

with incoming ray direction $\mathbf{u}$, surface normal $\mathbf{n}$, and indeces of refraction n and n_t. If the term in the radical is negative, there is no refracted ray and the ray should be reflected within the object (total internal reflection), i.e.,

$$\mathbf{t} = \mathbf{u} - 2(\mathbf{u}\cdot\mathbf{n})\mathbf{n} \tag{2}$$

Equation (1) can be implemented as a vector class `refract` member function that returns the refracted vector unless the expression inside the radical is negative, in which case a reflected vector is returned by calling the vector's `reflect` member function, which implements (2). With `vec_t` denoting the vector class in question, prototypes for the vector class member functions are suggested, where the second argument to `refract` is the index of refraction of the transparent object being intersected by the transmission ray (outside index of refraction modeling air is assumed).

```
vec_t vec_t::refract(const vec_t&, float);
vec_t vec_t::reflect(const vec_t&);
```

To render the transmitted color, students are instructed to blend the resultant surface color with the transmitted color via linear interpolation. Schlick's 1994 approximation is mentioned in passing and left to students as an optional implementation (see below).

1.1 Optional

With the functions `reflect` and `refract` implemented as specified, try implementing Shirley's solution to Schlick's 1994 approximation to the Fresnel equations for light transport (both reflection and refraction of light may occur at the surface interface where refractive indices differ). In this case two rays are spawned when

Table 1: Groovy graphics assignment metadata.

Summary	Introduction of ray-traced transmission.
Learning Outcomes	Students should be able to implement a correctly working transmissive ray tracer and be able to explain refraction.
Classification(s)	(3) Fundamentals (8) Rendering
Audience	Computer Science students in 1^{st} or 2^{nd} semester.
Dependencies	Students must have written a basic ray tracer prior to implementation of this assignment.
Prerequisites	The assignment builds on earlier assignments that required development of a basic ray tracer.
Strengths	Students like seeing how refraction works.
Weaknesses	Some care is needed to properly write the ray `reflect` and `refract` functions.
Variants	Implementing Schlick's approximation to the Fresnel's equations as per Shirley 2002, §9.7.
Assessment	The image produced should include a transparent object.

intersecting a transparent surface: a reflection and a transmission ray. The result is linearly interpolated with interpolant R:

$$R = R_0 + (1 - R_0)(1 - \cos\theta)^5,$$

where

$$R_0 = \left(\frac{n_t - 1}{n_t + 1}\right)^2$$

is the reflectance at normal incidence.

1.2 Caveats

Students are reminded that when a ray enters a transparent sphere, its `hits` routine should now calculate both roots of the quadratic equation and return the smaller of the two, unless the smaller of the two is less than some $\epsilon = 0.00001$ in which case the larger should be returned. That is, with ray anchor position $\mathbf{r}$ and direction $\mathbf{u}$ and sphere center $\mathbf{c}$ and radius r, compute $d = b^2 - 4ac$, where $a = \mathbf{u} \cdot \mathbf{u}$, $b = 2(\mathbf{r} - \mathbf{c}) \cdot \mathbf{u}$, $c = |\mathbf{r} - \mathbf{c}|^2 - r^2$, to obtain $t_0 = (-b - \sqrt{d})/(2a)$ and $t_1 = (-b + \sqrt{d})/(2a)$ and return t_0 if $t_0 < t_1$ unless it is behind the ray position (i.e., $t_0 < 0$). If t_1 is returned, the ray exits the sphere.

Calculation of t_1 can lead to infinite loops where rays can get stuck in a region close to the sphere surface. To prevent this, students are urged to keep track of the number of ray reflections or bounces. If the number of bounces exceeds say 5, then exit the `hits` routine (prune this ray as it has already bounced too many times).

2 METADATA

Metadata is given in Table 1. The assignment is meant for beginning Computer Science students who have written a ray tracer, e.g., as in an introductory course to computer science, see Duchowski [2014] for information concerning course and assignment progression.

3 MATERIALS

Materials required for this assignment is the basic layout of refraction along with particulars for implementation in code beyond the basic theory of Snell's Law. An input file specifying the virtual environment (e.g., Cornell box) is also provided, and it is assumed students have already written the file parser.

4 NOTES/LESSONS LEARNED

This assignment is usually solved by most if not all programmers of a ray tracer and so is not particularly unusual. The assignment, as given, is groovy from an educator's point of view because it assembles information from multiple sources into a convenient vector class representation (rays) that either refract or reflect, as expressed by the `vec_t` implementation.

Although statistics are not available regarding number of students solving this problem or with problems they may have met, one student's comment stands out in memory: being a math major, Carrie Eisengrein noted that although she knew what vectors (rays) were, she appreciated being able to use them in code and to visualize properties such as refraction as expressed by her code.

Another point of interest is that this assignment follows on a basic re-working of the ray tracer from C to C++ (see Duchowski [2014] for detailed information regarding the relevant course sequence) and, importantly, introduction to `OpenMP`. Both the recasting into object-oriented code and its parallelization help make this assignment fairly palatable.

5 SUPPLEMENTAL MATERIALS

The instructor's C++ code is provided along with this assignment description, tested with the `g++-mp-6` compiler on a Mac (running `macOS` version 10.13.4).

REFERENCES

Andrew T. Duchowski. 2014. Photons: Evolution of a Course in Data Structures. *Computer Graphics Forum* 33, 1 (2014), 294–304. https://doi.org/10.1111/cgf.12279

Christophe Schlick. 1994. An Inexpensive BRDF Model for Physically-based Rendering. *Computer Graphics Forum* 13, 3 (1994), 233–246.

Peter Shirley. 2002. *Fundamentals of Computer Graphics.* A. K. Peters, Ltd., Natick, MA.