# Raytracer

1st Jiakang Weng
TU Dresden
Dresden, Germany
Jiakang.Weng@mailbox.tu-dresden.de

2nd Mingyu Ma
TU Dresden
Dresden, Germany
Mingyu.Ma@mailbox.tu-dresden.de

3rd Yankai Su
TU Dresden
Dresden, Germany
Yankai.Su@mailbox.tu-dresden.de

4th Shaofeng Liang
TU Dresden
Dresden, Germany
Shaofeng.Liang@mailbox.tu-dresden.de

Abstract—Raytracing is a powerful tool for the simulation of propagation paths in large scale applications (e.g. space-borne radar). It can reduce computation times while maintaining a high level of accuracy enabling the estimation of radar scenario and communication performance in complex settings.

Index Terms—electromagnetic waves, propagation, simulation, SBR, raytracing

## I. Introduction

In order to generate visible images in a three-dimensional computer graphics environment, ray tracing is a realistic implementation method. This method works by inversely tracking the optical path that intersects the imaginary camera lens. Since a large number of similar rays traverse the scene, the visible information of the scene seen from the camera angle and the specific lighting conditions of the software can be constructed. Calculate the reflection, refraction, and absorption of light when it intersects with objects or media in the scene.

Ray tracing scenes are often described by programmers using mathematical tools, or by visual artists using intermediate tools, or images or model data captured from different technical methods such as digital cameras. A ray tracing program mathematically determines and replicates the path of light from an image, but in the opposite direction (from the eye back to the origin). Ray tracing is now widely used in computer games and animation, television and DVD production, and movie products. Many manufacturers provide ray tracing programs for personal computers. In ray tracing, each ray path consists of multiple straight lines, almost always containing reflection, refraction, and shadow effects from the origin to the scene. Each ray is represented by a mathematical equation, defining the spatial path of the ray as a function of time, before reaching the screen. The color or color of the target in the passing scene is realized by assigning a color to each light . Every pixel on the screen corresponds to every light that can be traced back to the source. Ray tracing was first invented in the 1960s by scientists in an organization called the Mathematics Applications Group. Some of these scientists became interested in ray tracing as an art, and established an animation photography studio to use ray tracing to produce 3D computer portraits and animations for TV and movies.

## II. Background

### A. The Snell's Law

When light waves propagate from one medium to another with a different refractive index, refraction occurs. The relationship between the incident angle and the refraction angle can be described by Snell's Law. Snell's law is named after the Dutch physicist Willibo Snell, also known as ”refraction law”.

In optics, ray tracing technology uses Snell's law to calculate the angle of incidence and the angle of refraction. In experimental optics and gemology, this law is applied to calculate the refractive index of matter.

Snell's law shows that when light waves propagate from medium 1 to medium 2, if the refractive indexes of the two media are different, the phenomenon of refraction will occur. The incident light and the refracted light are in the same plane, which is called the ”incident plane”. , And the angle with the interface normal fulfills the following relationship:

$$n_1 \cdot \sin\theta_1 = n_2 \cdot \sin\theta_2 \tag{1}$$

Among them, $n_1$and $n_2$ are the refractive indexes of the two media, $\theta_1$ and $\theta_2$ is the angle between the incident light, refracted light and the interface normal, which are called ”incidence angle” and ”refraction angle”, respectively.

This formula is called the ”Snell formula”.

Snell's law can be derived from Fermat's principle or Huygens' principle, translational symmetry, or Maxwell's equations.

In some cases, when light travels from a medium with a higher refractive index to one with a lower refractive index, the angle of the transmitted light could be 90 degree. In optical physics, the energy of transmission will be zero and only reflection will exsit. This phenomenon is named ”Total Reflection”. The critical angle is determined as follows:

$$\theta = \sin^{-1}\left(\frac{n_2}{n_1}\right) \tag{2}$$

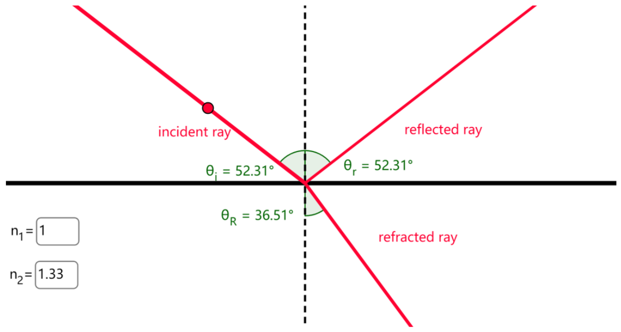


Fig. 1. Snell's Law

This will only happen when light enters a light dense medium (a medium with a higher refractive index) into a light medium (a medium with a lower refractive index) and the angle of incidence is greater than the critical angle. Because there is no refraction (refracted light disappears) but reflection, it is called total reflection. For example, it occurs when light enters air from glass, but not when light enters glass from air. The most common is that the bubbles in the boiling water appear very bright because of total reflection.

## B. Refractive index

Refractive index, the ratio of the speed of light in vacuum to the speed of light in a medium. The higher the refractive index of the material, the stronger the ability to refract the incident light. The higher the refractive index, the thinner the lens, that is, if the thickness of the lens center is the same(the same degree of the same material), the refractive index of the lens with a higher refractive index is thinner than that of the lens with a lower refractive index. The refractive index is closely related to the electromagnetic properties of the medium. According to classical electromagnetic theory, $\varepsilon_r$ and $\mu_r$ are the relative permittivity and relative permeability of the medium, respectively. Refractive index is also related to frequency,which is called dispersion phenomenon. When light is emitted from a relatively dense medium to a relatively thin medium, and the incident angle is greater than the critical angle, total reflection can occur.

The refractive index of the medium *mathitn* is equal to the ratio of "speed of light in vacuum $c$" to "phase speed of light in medium $v$"

$$\mathbf{n} = \frac{\mathbf{c}}{\mathbf{v}} \tag{3}$$

When comparing the two media, the dense refractive index is called the optical dense medium, and the smaller refractive index is called the light sparse medium. The refractive index is closely related to the electromagnetic properties of the medium. According to classical electrodynamics,$\varepsilon_r$ and $\mu_r$ are the relative permittivity and relative permeability of the medium, respectively. The refractive index is also related to the wavelength,

TABLE I
Selected refractive indices at $\lambda$=589nm

| Material | n |
|---|---|
| Vacuum | 1 |
| Gases at 0 °C and 1 atm | |
| Air | 1.000293 |
| Helium | 1.000036 |
| Hydrogen | 1.000132 |
| Carbon dioxide | 1.00045 |
| Liquids at 20 °C | |
| Water | 1.333 |
| Ethanol | 1.36 |
| Olive oil | 1.47 |
| Solids | |
| Ice | 1.31 |
| Fused silica (quartz) | 1.46 [1] |
| Window glass | 1.52 [2] |

called dispersion phenomenon. The refractive index data provided in the manual is for a specific wavelength (usually for sodium yellow light, the wavelength is 5893Å). The gas refractive index is also related to temperature and pressure. The refractive index of air is very close to 1 for light of various frequencies. For example, the refractive index of air at 20°C and 760 mmHg is 1.00027. In engineering optics, the refractive index of air is often regarded as 1, while the refractive index of other media is the relative refractive index to air.

## C. Fresnel equations

Snell's law (also known as Snell–Descartes law and the law of refraction) is a formula used to describe the relationship between the angles of incidence and refraction, when referring to light or other waves passing through a boundary between two different isotropic media, such as water, glass, or air.

In order to further describe the reflection and transmission of light (or electromagnetic radiation in general) when incident on an interface between different optical media, Fresnel coefficients (or Fresnel equations) must be considered. When light incident a boundary between a medium with refractive index $n_1$ and a second medium with refractive index $n_2$, both reflection and refraction of the light may occur.

Before the mathematical analysis of Fresnel equations, there are six important rules about reflection and refraction should be mentioned.

1) The frequency of the reflected wave and the refracted wave is equal to the frequency of the incident wave;
2) The incident light, the reflected light and refracted line are in the same plane
3) The angle of incidence is equal to the angle of reflection.
4) The ratio of the sine of the angle of incidence to the angle of refraction is inversely proportional to the relative refractive index. (Snell's law)
5) The amplitudes of the incident wave, reflected wave and refracted wave satisfy the Fresnel equations.

And the amplitudes of plane electromagnetic wave is related to polarization (s polarization and p polarization).

6) If the incident wave is s- or p-polarized, the reflected and refracted waves are also s- or p-polarized.

s-polarization: E is perpendicular to the plane of incidence, and the electric field has no component perpendicular to the interface. p-polarization: E is parallel to the plane of incidence and there is an electric field component perpendicular to the interface.

Consider that the interface between the two media is an infinite plane, and a plane electromagnetic wave incidents on the interface.

$$\mathrm{E}(\mathrm{x}, t) = \mathrm{E}_0 e^{j(\mathrm{k}\cdot\mathrm{x}-\omega t)} \tag{4}$$

Two new waves at the boundary emerge, those in medium 1 are called reflected waves, and those in medium 2 are called refracted wave.

$$\mathrm{E}'(\mathrm{x}, t) = \mathrm{E}_0' e^{j\left(\mathrm{k}'\cdot\mathrm{x}-\omega' t\right)} \tag{5}$$

$$\mathrm{E}''(\mathrm{x}, t) = \mathrm{E}_0'' e^{j\left(\mathrm{k}''\cdot\mathrm{x}-\omega'' t\right)} \tag{6}$$

The simplest assumption: both reflected and refracted waves are plane waves.

Note: The total field strength in the medium 1 is the superposition of the incident wave and the reflected wave, while in the medium 2 there are only refracted waves. At

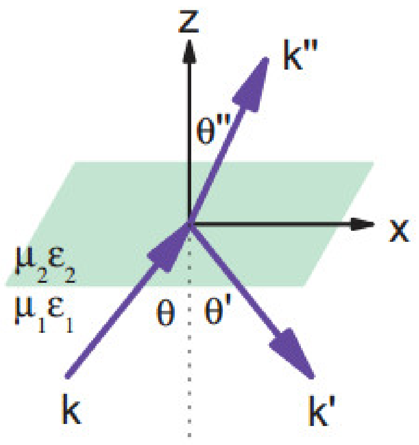


Fig. 2. Fresnel equations

any time t and any position (x; y; 0) on the interface z = 0, the boundary conditions are satisfied:

$$\mathrm{n}\times(\mathrm{E}+\mathrm{E}') = \mathrm{n}\times\mathrm{E}'' \tag{7}$$

$$\mathrm{n}\times\left(\mathrm{E}_0 e^{j(\mathrm{k}\cdot\mathrm{x}-\omega t)} + \mathrm{E}_0' e^{j\left(\mathrm{k}'\cdot\mathrm{x}-\omega' t\right)}\right) = \mathrm{n}\times \mathrm{E}_0'' e^{j\left(\mathrm{k}''\cdot\mathrm{x}-\omega'' t\right)} \tag{8}$$

The frequency of the reflected and refracted waves is equal to the frequency of the incident wave:

$$\omega = \omega' = \omega'' \tag{9}$$

$$k_x = k_x' = k_x'' \tag{10}$$

$$k_y = k_y' = k_y'' \tag{11}$$

Without loss of generality make $k_y = 0$ The incident light, the reflected light and refracted line are in the same plane $y = 0$.

$$k\sin\theta = k'\sin\theta' = k''\sin\theta'' \tag{12}$$

$$k = k' = \frac{\omega}{v_1} \tag{13}$$

$$k_y = k_y' = k_y'' \tag{14}$$

Draw a conclusion

- The angle of incidence is equal to the angle of reflection

$$\theta = \theta' \tag{15}$$

- The relationship between the angle of incidence and the angle of refraction

$$\frac{\sin\theta}{\sin\theta''} = \frac{v_1}{v_2} = \sqrt{\frac{\mu_2\varepsilon_2}{\mu_1\varepsilon_1}} = \frac{n_2}{n_1} \tag{16}$$

Generally (except for ferromagnetic media) $\mu \approx \mu_0$,therefor $\frac{n_2}{n_1} \approx \frac{\varepsilon_2}{\varepsilon_1}$

For s-polarization

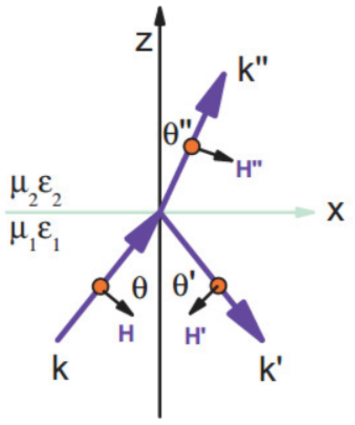


Fig. 3. s-polarization

$$E = E' = E'' \tag{17}$$

$$H\cos\theta - H'\cos\theta' = H''\cos\theta'' \tag{18}$$

Use $H = \sqrt{\frac{\varepsilon}{\mu}}E$, make $\mu = \mu_0$, therefore,

$$\sqrt{\varepsilon_1}(E - E')cos\theta = \sqrt{\varepsilon_2}E''cos\theta'' \tag{19}$$

$$\frac{\sin\theta}{\sin\theta''} = \sqrt{\frac{\varepsilon_2}{\varepsilon_1}} \tag{20}$$

$$\frac{E'}{E} = \frac{\sqrt{\varepsilon_1}cos\theta - \sqrt{\varepsilon_2}cos\theta''}{\sqrt{\varepsilon_1}cos\theta + \sqrt{\varepsilon_2}cos\theta''} = -\frac{\sin(\theta-\theta'')}{\sin(\theta+\theta'')} \tag{21}$$

$$\frac{E''}{E} = \frac{2\sqrt{\varepsilon_1}cos\theta}{\sqrt{\varepsilon_1}cos\theta + \sqrt{\varepsilon_2}cos\theta''} = \frac{2\cos\theta\sin\theta''}{\sin(\theta+\theta'')} \tag{22}$$

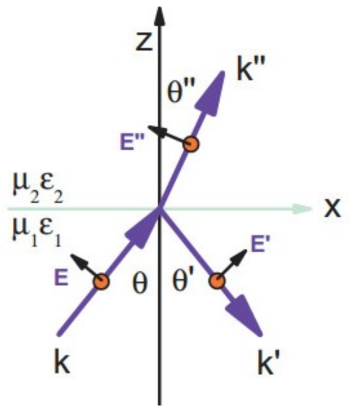


Fig. 4. p-polarization

For p-polarization

$$H = H' = H'' \tag{23}$$

$$E\cos\theta - E'\cos\theta' = E''\cos\theta'' \tag{24}$$

Use $H = \sqrt{\frac{\varepsilon}{\mu}}E$, make $\mu = \mu_0$, therefore,

$$\sqrt{\varepsilon_1}(E - E') = \sqrt{\varepsilon_2}E'' \tag{25}$$

$$\frac{\sin\theta}{\sin\theta''} = \sqrt{\frac{\varepsilon_2}{\varepsilon_1}} \tag{26}$$

$$\frac{E'}{E} = \frac{\tan(\theta - \theta'')}{\tan(\theta + \theta'')} \tag{27}$$

$$\frac{E''}{E} = \frac{2\cos\theta\sin\theta''}{\sin(\theta + \theta'')\cos(\theta - \theta'')} \tag{28}$$

There are 3 special cases. Brewster' Law, Half-Wave Loss, Total Internal Reflection.

Brewster' Law: When $\theta + \theta'' = \frac{\pi}{2}$ ,$\frac{E'}{E} = 0$ in equation(**). Therefore, p-polarized component has no reflected waves, and the reflected light becomes completely s-polarized light; The incident angle at this time is called Brewster angle.

Half-Wave Loss: During s-polarized reflection, when$\varepsilon_2 > \varepsilon_1$, $\theta > \theta''$. Therefore, in the Fresnel equation (17) for s-polarization, $\frac{E'}{E}$ is negative, that is, the phase of electric field for reflected wave is reverse to the incident wave. This phenomenon is called half-wave loss during the reflection process.

Total Internal Reflection: If $\varepsilon_1 > \varepsilon_2$, when the incident angle of electromagnetic waves $\theta = arcsin(\frac{n_2}{n_1})$, $\theta'' = \frac{\pi}{2}$ That is, $sin\theta \geq \frac{n_2}{n_1}$. This phenomenon is called total Internal Reflection. This angle is called critical angle.

## D. Triangle mesh

Triangle mesh is a kind of polygon mesh. Polygon mesh is also called "Mesh", it's a data structure used in computer graphics to model various irregular objects. In the real world,the surface of objects is composed of curved surfaces, but in the computer world, however, only discrete structures can be used to simulate continuous things in reality. So the curved surface in the real world is actually composed of a large number of small polygon patches in the computer. For example, the model in the Fig.5.

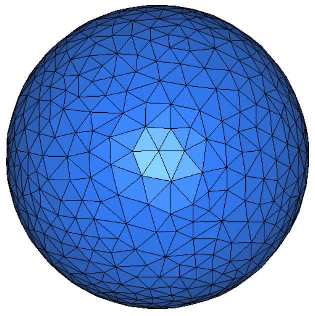


Fig. 5. Example of Triangle Mesh

In fact, a large number of small triangles are used inside the computer to form such a shape. Such a collection of facets is called Mesh. Mesh can be composed of triangles or other plane shapes such as quadrilaterals, pentagons, etc.; Because all the plane polygons can also be subdivided into triangles, usually a triangle mesh composed of triangles could be used to represent the surface of an object.

For the Triangle Mesh, most of kernels use indices and vertex to set the data of Triangle, not all the Triangle needs 3 vertices, because it can be connected with other triangle, so we need to set the vertex data. And a triangle whose vertices are laid out counter-clockwise has its geometry normal pointing upwards outside the triangle plane, we use indices to control counter-clockwise or clockwise.

## E. Bounding Volume Hierarchy

A bounding volume hierarchy (BVH) is a tree structure on a set of geometric objects. All geometric objects are wrapped in bounding volumes that form the leaf nodes of the tree. These nodes are then grouped as small sets and enclosed within larger bounding volumes. These bounding volumes, in turn, are also grouped and enclosed within other larger bounding volumes in a recursive fashion, eventually resulting in a tree structure with a single bounding volume at the top of the tree. Bounding volume hierarchies are used to support several operations on sets of geometric objects efficiently, such as in collision detection and ray tracing [3].

For example, when Snell's law in Python is implemented, multiple geometry or planes should be calculated, for complex 3D scene,a algorithm called AABB should be used. Let the ray operate with each graph to calculate the

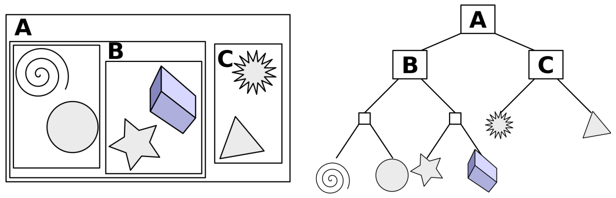


Fig. 6. Example of BVH Tree

intersection point, and then determine its true intersection point, so algorithmic complexity is n. And a tree structure could be used to manage graphics , which can reduce the complexity to $log(n)$.

Implementing BVH requires two stages of work: Build and Traversal. The construction work considers how to construct a binary tree that can effectively describe the current scene information. The key to this is how to divide assumed all objects scattered randomly in the scene Partition, that is, decide which objects should be divided into the left subtree and which objects should be divided into the right subtree.

The better way is Surface Area Heuristic(SAH). The SAH algorithm calculates the mean value of the average complexity based on the size of the surface area and the direction of the rays, thereby dividing the BVH tree into the structure with the lowest mean value.

First select an axis with the largest divergence in the graphic distribution, and then spilt it equally along the axis in space, Fig.7. shows two different scheme spilting, and traversal all of scheme to find the best one.

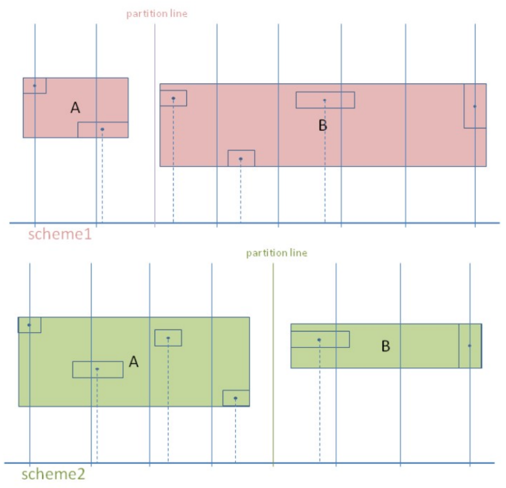


Fig. 7. Example of SAH Spilt

SAH does not completely ”non-overlap” or make the distribution after partitioning ”uniform”, like Fig.8. but each time it makes a partition, it chooses the best plan under the current situation, so it is called Heuristic algorithm.

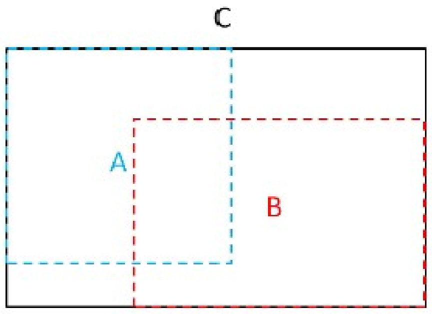


Fig. 8. Example of Spilt

## F. Test Driven Development

Test-driven development, referred to as TDD, is a new development method different from the traditional software development process. It requires writing test code before writing the code of a certain function, and then only writing the function code that passes the test, and promoting the entire development through the test. This helps to write concise, usable and high-quality code and speed up the development process. Mr. Kent Beck first recommended the ”test-driven” best practice in his Extreme Programming (XP) methodology [4]. He also wrote a book called ”Test-Driven Development”, detailing how to implement it. After several years of rapid development, test-driven development has grown into an independent software development technology, and its reputation has even overshadowed extreme programming.

The basic idea of test-driven development is to write test code before developing functional code, and then only write functional code that passes the test, thereby driving the entire development process with testing. This helps to write simple, usable, and high-quality code, has high flexibility and robustness, can quickly respond to changes, and accelerate the development process.

The basic process of test-driven development is as follows:

1) Quickly add a test.
2) Run all the tests (sometimes only need to run one or a part) and find that the newly added test fails.
3) Some methods which may not be perfect could be added in the program, so the test code could run as soon as possible.
4) Run all the tests and all pass.
5) Refactor code to eliminate duplicate design and optimize design structure.

The advantages of TDD:

1) Can effectively avoid the waste caused by excessive design. However, it was also emphasized that a com-

plete design and implementation before development can effectively avoid the waste caused by refactoring.

2) Allows developers to have a more comprehensive perspective in development

The disadvantages of TDD:

1) The developer may only complete the code that satisfies the test, while ignoring the implementation of actual requirements. Some programmers believe that pair programming can effectively avoid this problem.
2) Will slow down the development of actual code, especially for the prototype development that requires development speed. Here we need to consider that the development speed needs to include two aspects of function and quality. The pure code speed may not completely represent the development speed.
3) Some methods which may not be perfect could be added in the program, so the test code could run as soon as possible.
4) Make development focus on use cases and test cases, rather than the design itself. There is currently a lot controversy about this view.

### G. Tools

a) Python: Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant white space. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. [5]

Compared to C++, Python has a simpler syntax, and Python also has many extension packages. such as Numpy, it can be used to calculate vectors directly, Sympy can also be used to solve equations with symbols. With it, complex equations could be solved easily, and overloading operator could be very well avoided.

In addition, Python also supports interactive shells, such as ipython, which is very friendly to beginners in programming, which also speed up language learning and greatly improve the debugging capabilities.

b) C++: Although Python is very convenient for mathematical operations and friendly enough for beginners, but Python has cross-platform packaging problems and the performance is not good enough. Therefore, projects with huge calculations such as ray tracing generally use better performance and better use of programming languages with different data structures, such as C++.

c) Embree: Intel® Embree is a collection of high-performance ray tracing kernels, developed at Intel. The target users of Intel® Embree are graphics application engineers who want to improve the performance of their photo-realistic rendering application by leveraging Embree's performance-optimized ray tracing kernels. The single-ray traversal kernels of Intel® Embree provide high performance for incoherent workloads and are very easy to integrate into existing rendering applications. Using the stream kernels, even higher performance for incoherent rays is possible, but integration might require significant code changes to the application to use the stream paradigm. In general for coherent workloads, the stream mode with coherent flag set gives the best performance. Intel® Embree also supports dynamic scenes by implementing high-performance two-level spatial index structure construction algorithms. [6]

Embree is compiling with C++, and Embree API could be used to realise Raytracing.

## III. Simulation Design

### A. Physical and Mathematical Modelling

1) Calculate reflected light: According to the previously mentioned law of reflection, the angles have the following relationship:

$$\Theta_{incidentlight} = \Theta_{reflectedlight}$$

In the three-dimensional coordinate system, the refraction vector need to be calculated. In the Fig.9 a plane AB, incident light vector $\vec{i}$, normal vector $\vec{n}$, and reflected light vector $\vec{r}$ are showed.

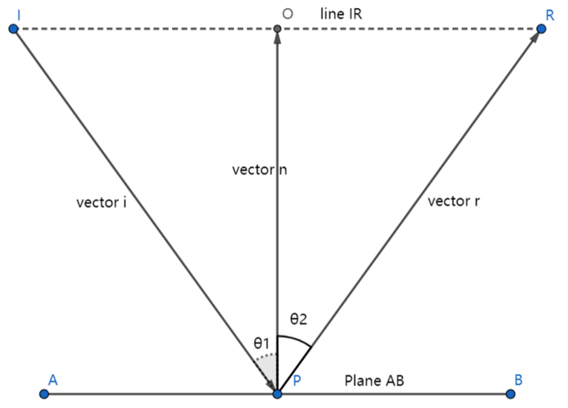


Fig. 9. vector i is the incident light, vector r is the reflected light, AB is a plane, and there is $\Theta_1 = \Theta_2$

Because in the figure $\Theta_1 = \Theta_2$, then $\vec{i} + \vec{PO} = \vec{r} - \vec{PO}$, so we get $\vec{r} = \vec{i} + 2 \cdot \vec{PO}$, and here notice that $\vec{PO} \neq \vec{n}$. And then calculate $\vec{PO}$, $\vec{PO} = -\left(\vec{i} \cdot \vec{n}\right) \cdot \vec{n}$ And then take $\vec{PO}$ to the front equation, the vector of the reflected light will be:

$$\vec{r} = \vec{i} - 2 * \left(\vec{i} \cdot \vec{n}\right) \cdot \vec{n}; \qquad \vec{i} \cdot \vec{n} \neq 0$$

When the incident is vertical to the plane($\vec{i} \cdot \vec{n} = 0$), this case needs to be dealt with separately. And just set the output as $\vec{r} = -\vec{i}$.

2) Calculate the intersection point: The incident light can be described by a point and a direction vector in space. The point assume as $pl = [pl_0, pl_1, pl_2]$, and the direction vector as $\vec{d} = [v_0, v_1, v_2]$. In this way, the function of the light can be described by the following equation:

$$f(x) = \begin{cases} x = & pl_0 + v_0 * t \\ y = & pl_1 + v_1 * t \\ z = & pl_2 + v_2 * t \end{cases}$$

The plane can be described by a point $po = [pp_0, pp_1, pp_2]$ and a normal vector $\vec{n} = [n_0, n_1, n_2]$ in space. All the vector in plane is vertical to the normal vector: $\vec{n} \cdot \vec{x_i} = 0$ Assuming that vector $\vec{x_i} = [x - pp_0, y - pp_1, z - pp_2]$ is a vector in the plane. $\vec{x_i}$ is perpendicular to the normal vector.

$$n_0 * (x - pp_0) + n_1 * (y - pp_1) + n_2 * (z - pp_2) = 0$$

Then take the point $po$ in the equation, and get the numerical solution of t.

$$t = \frac{n_0 * (pp_0 - pl_0) + n_1 * (pp_1 - pl1) + n_3 * (pp_2 - pl_2)}{n_0 * v_0 + n_1 * v_1 + n_2 * v_2}$$

The position of the intersection can be calculated by the t parameter equation.

3) Calculate the refracted light: According to the previously mentioned Snell's Law, the angles have the relationship:

$$n_1 \cdot \sin\Theta_1 = n_2 \cdot \sin\Theta_2$$

The following figure describes this angle relationship. Here assumes the incident light from air to the water. The refraction index from air to water is 1 to 1.33.

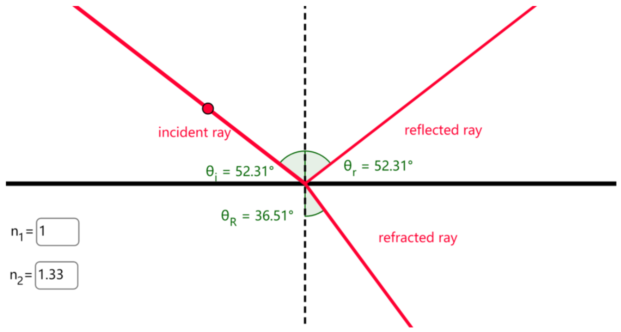


Fig. 10. In this figure, the angle between the light and the plane normal is 52.31°, and the ratio of the index n1 to n2 is 1:1.33. Then the angle between the reflected light and the plane normal is 52.31°, the angle between the refracted light and the plane normal is 36.51°

the function for calculate the refracted light:

$$\vec{t} = \vec{n} \times \vec{a} \times \vec{n} \cdot n_1/n_2 + \sqrt{|\vec{a}|^2 - |\vec{n} \times \vec{a} \times \vec{n}|^2} \cdot \vec{n}$$

and the vector n should meet the following conditions:$\vec{a} \cdot \vec{n} > 0$

And when the incident light from the medium with a larger index of refraction to the medium with a smaller index of refraction, when then incident angle and the index meet the following conditions, the total reflection will occur. The figure shows the light from water into the air, and the refracted light does not exist.

TABLE II
Non hypothesis

| Cases | Output value |
|---|---|
| Incident light vertical to the plane | The reflected light is opposite to the incident light, and the refracted light is in the same direction as the incident light. |
| Incident light parallel to the plane | no intersection, no light. |
| Incident light vector is null | |
| The normal vector of the plane is null | |
| Total reflection | No refracted light. |

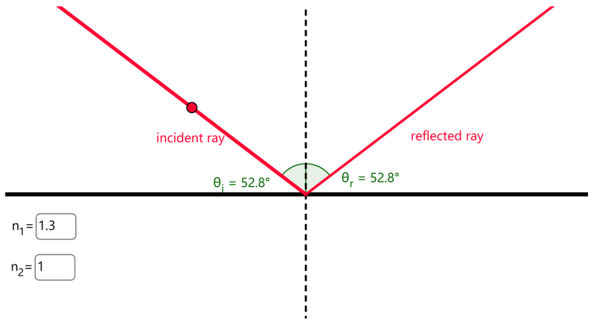


Fig. 11. In this figure, the angle between the light and the plane normal is 52.8°, and the ratio of the index n1 to n2 is 1.3:1. Then the angle between the reflected light and the plane normal is 52.8°, the refracted light not exist.

## B. Test cases

In software engineering, a test case is a specification of the inputs, execution conditions, testing procedure, and expected results that define a single test to be executed to achieve a particular software testing objective, such as to exercise a particular program path or to verify compliance with a specific requirement [7]. It is necessary to prepare corresponding hypothesis and Non hypothesis.

1) Hypothesis: In hypothesis cases there will be a normal situation for the program. The example in Fig.10 is a hypothesis case. If this Hypothesis is correct, the result of the program should be consistent with the result of the mathematical calculation, that is, the angle between the calculated reflected light vector and the refracted light vector satisfies the above relationship.

2) Non Hypothesis: Non hypothesis cases (or special cases) will cause calculation errors due to computer or mathematical operations. The table II shows the Non hypothesis for the program.

In order to run the test in python programming, Pytest is needed. Pytest is a practical full-featured Python testing tool that help write better programs [8]. Use Pytest to test the cases in the corresponding program until all the above black box tests passed.

## C. The use of Embree

1) Compiling Embree: In this project Embree was compiled on Ubuntu, and the release version of Embree on the Windows platform was installed. The embree-based program was developed on Windows platform. The official recommendation is that using Cmake to compile to generate project files on different platforms. In this Windows Platform, Visual Studio was being used as IDE to compile the project.

2) Embree API: The simplest case: one Device object, one Scene object and one Geometry object.

- Device object: it allows different components of the application to use the Embree API without interfering with each other.
- Scene object: it's a container for a set of geometries, and contains a spatial acceleration structure which can be used to perform different types of rayqueries.
- Geometry object: it depends on geometry type, like triangle mesh.

3) A Simple Demo: According to the official tutorial and the official API a hit of ray and scene could be achieved. The scene contains 8 triangle mesh, an has 2 rays.

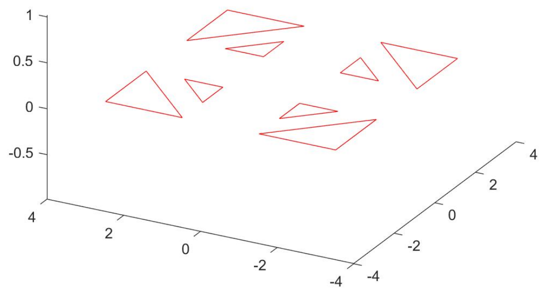


Fig. 12. Triangle location

BVH Tree is not being used in this demostration, so the complexity of Intersect's algorithm is $o(n)$. At the same time, the running time of the program has been calculated, although there are millisecond tolerance, the accuracy is sufficient.

## IV. Result

### A. The Result of Physical and Mathematical Modelling

The details of the model have been introduced in detail above, and the code is written according to the model, and the simulation results are obtained by simulating different situations (initial parameters). In order to make the illustration clear, each scenario only contains one plane and one incident light, which also means, the simulation is limited in one-surface-cases.

1) The result of random cases: In order to make demostration easy, the length of incident light, reflected light, and refracted light is set to unit 1, and only the plane near the intersection point is drawn. Randomly select an incident light and plane as a known condition, make sure the reflected light and refracted light exist, the fig.13 is shown below.

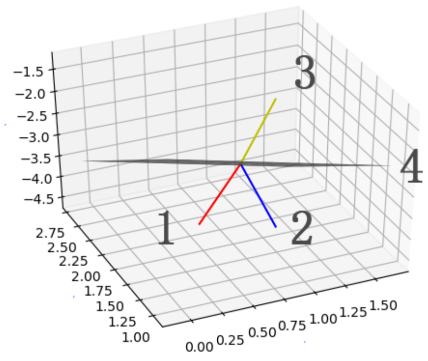


Fig. 13. The incident ray vector is [2,1,4], the point in incident light is [1,2,-3], the plane passes through the point [0,0,0] with the normal vector [2,4,3], The index ratio between incident medium and refract medium is 1:1.3. The point and the direction of the reflected light is [0.9, 1.95, -3.2], [-0.16554461 -0.98574288 -0.03009902], the direction of the refracted light is [0.43694564 0.33083027 0.83643879]

In the fig.13 , the red line 1 represents the incident light, the blue light 2 represents the reflected light, the yellow light 3 represents the refracted light, and the gray area 4 represents the object plane. When changing the index ratio between incident medium and refract medium, there will be some situation that the refracted light does not exist like the fig.14.

When the incident light is parallel to the plane, the incident vector is perpendicular to the plane normal vector, and there is no intersection between the light and the plane, so there is no reflected light or refracted light. The fig.15 shows this situation.

When the incident light is nearly vertical to the plane, the incident vector is nearly parallel to the plane normal vector, then the reflected light nearly coincides with the incident light, the refracted light has nearly the same direction as the incident light, which is shown in fig16

2) The result of test cases: For the test cases in Pytest, the results of light simulation need to be verified by the pytest program. In other words, compare the results of light simulation with the results of using mathematical methods, and if the results are the same within the error tolerance range, the test for this case can be passed. If the results are not the same(or exceed the tolerance range),

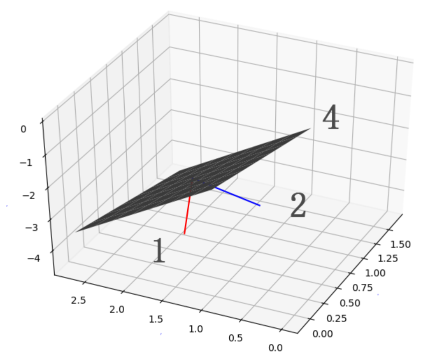


Fig. 14. The incident ray vector is [2,1,4], the point in incident light is [1,2,-3], the plane passes through the point [0,0,0] with the normal vector [2,4,3], The index ratio between incident medium and refract medium is 2:1. The point and the direction of the reflected light is [0.9, 1.95, -3.2], [-0.16554461 -0.98574288 -0.03009902], because of the Total reflection there is no transmitted (refracted) light.

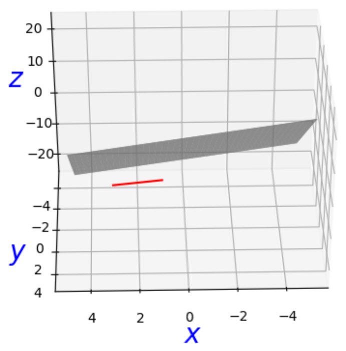


Fig. 15. The incident ray vector is [2,1,2], the point in incident light is [1,2,-3], the plane passes through the point [0,0,0] with the normal vector [-1,4,-1].

the algorithm needs to be modified and retest until the result fufill the condition.

The table III shows the four different cases for the pytest. The case 1 will have no refracted light because of total reflection. Case 2 shows the incident light is vertical to the plane. In case 3 set the incident light as null. In case 4 the incident light is parallel to the plane.

The table IV shows the expected results of the various situations in table III. Here, when the ray direction vector is [0,0,0], it means that the ray does not exist.

## B. Embree Demo

1) Our Demo: Only the result in Command is printed, there is no image output.

Single ray structure are define in this structure, it's also in Embree API.

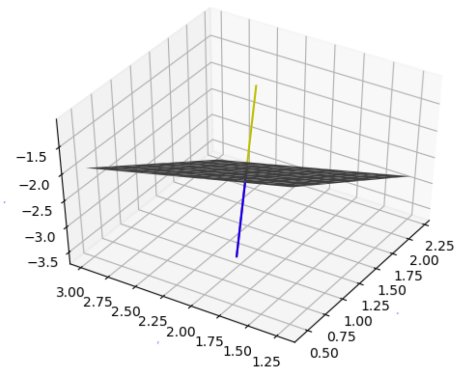


Fig. 16. The incident ray vector is [2,1,1.999](normal as [0.6668 0.3334 0.6665]), the point in incident light is [1,2,-3], the plane passes through the point [0,0,0] with the normal vector [2,1,2]. The index ratio between incident medium and refract medium is 1:1. The point and the direction of the reflected light is [1.4445, 2.2223, -2.5557], [-0.6665 -0.3333 -0.6669], the direction of the refracted light is [0.6668 0.3334 0.6665]

TABLE III
Default values for the four cases

| Cases | Incident light [Point], [Direction] | Plane [Point], [Normal vector] | Index |
|---|---|---|---|
| Case 1 | [-1,1,0], [1,-1,0] | [0,0,0], [0,1,0] | $\sqrt{2}:1$ |
| Case 2 | [0,1,0], [0,-1,0] | [0,0,0], [0,1,0] | 1:1 |
| Case 3 | [0,0,0], [0,0,0] | [0,0,0], [0,1,0] | 1:1 |
| Case 4 | [0,1,0], [1,1,0] | [0,0,0], [-1,1,0] | 1:1 |

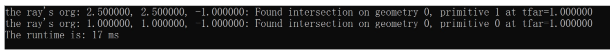
```
#include <embree3/rtcore_ray.h>
struct RTC_ALIGN(16) RTCRay
{
    float org_x;
    float org_y;
    float org_z;
    float tnear;
    float dir_x;
    float dir_y;
    float dir_z;
    float time;
    float tfar;
    unsigned int mask;
    unsigned int id;
    unsigned int flags;
};
```

TABLE IV
The expected value of the results of the four cases

| Cases | Reflected light [Direction] | Point | Refracted light [Direction] |
|---|---|---|---|
| Case 1 | [1,1,0] | [0,0,0] | [0,0,0](not exits) |
| Case 2 | [0,1,0] | [0,0,0] | [0,-1,0] |
| Case 3 | - | False | - |
| Case 4 | - | False | - |

the ray's org is the coordinate of ray origin. and geometry 0 is the geometry ID of our geometry object, primitive is ID of the triangle, base on the indices buffer, tfar is end of ray segment.

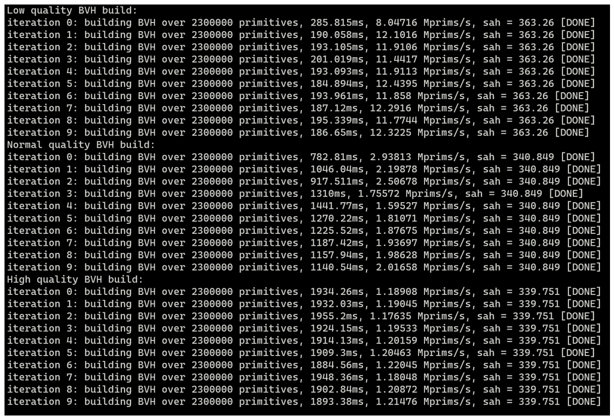
Fig.17 shows two rays hit 2 Triangles, and the program run time is 17ms.



```
the ray's org: 2.500000, 2.500000, -1.000000: Found intersection on geometry 0, primitive 1 at tfar=1.000000
the ray's org: 1.000000, 1.000000, -1.000000: Found intersection on geometry 0, primitive 0 at tfar=1.000000
The runtime is: 17 ms
```

Fig. 17. Demo Result

a) BVH Result: Embree's Tutorial gives a Demo to show the performance of BVH with SAH.



```
Low quality BVH build:
iteration 0: building BVH over 2300000 primitives, 285.815ms, 8.04716 Mprims/s, sah = 363.26 [DONE]
iteration 1: building BVH over 2300000 primitives, 190.058ms, 12.1016 Mprims/s, sah = 363.26 [DONE]
iteration 2: building BVH over 2300000 primitives, 193.105ms, 11.9106 Mprims/s, sah = 363.26 [DONE]
iteration 3: building BVH over 2300000 primitives, 201.019ms, 11.4417 Mprims/s, sah = 363.26 [DONE]
iteration 4: building BVH over 2300000 primitives, 193.093ms, 11.9113 Mprims/s, sah = 363.26 [DONE]
iteration 5: building BVH over 2300000 primitives, 184.894ms, 12.4395 Mprims/s, sah = 363.26 [DONE]
iteration 6: building BVH over 2300000 primitives, 193.961ms, 11.858 Mprims/s, sah = 363.26 [DONE]
iteration 7: building BVH over 2300000 primitives, 187.12ms, 12.2916 Mprims/s, sah = 363.26 [DONE]
iteration 8: building BVH over 2300000 primitives, 195.339ms, 11.7744 Mprims/s, sah = 363.26 [DONE]
iteration 9: building BVH over 2300000 primitives, 186.65ms, 12.3225 Mprims/s, sah = 363.26 [DONE]
Normal quality BVH build:
iteration 0: building BVH over 2300000 primitives, 782.81ms, 2.93813 Mprims/s, sah = 340.849 [DONE]
iteration 1: building BVH over 2300000 primitives, 1046.04ms, 2.19878 Mprims/s, sah = 340.849 [DONE]
iteration 2: building BVH over 2300000 primitives, 917.511ms, 2.50678 Mprims/s, sah = 340.849 [DONE]
iteration 3: building BVH over 2300000 primitives, 1310ms, 1.75572 Mprims/s, sah = 340.849 [DONE]
iteration 4: building BVH over 2300000 primitives, 1441.77ms, 1.59527 Mprims/s, sah = 340.849 [DONE]
iteration 5: building BVH over 2300000 primitives, 1270.22ms, 1.81071 Mprims/s, sah = 340.849 [DONE]
iteration 6: building BVH over 2300000 primitives, 1225.52ms, 1.87675 Mprims/s, sah = 340.849 [DONE]
iteration 7: building BVH over 2300000 primitives, 1187.42ms, 1.93697 Mprims/s, sah = 340.849 [DONE]
iteration 8: building BVH over 2300000 primitives, 1157.94ms, 1.98628 Mprims/s, sah = 340.849 [DONE]
iteration 9: building BVH over 2300000 primitives, 1140.54ms, 2.01658 Mprims/s, sah = 340.849 [DONE]
High quality BVH build:
iteration 0: building BVH over 2300000 primitives, 1934.26ms, 1.18908 Mprims/s, sah = 339.751 [DONE]
iteration 1: building BVH over 2300000 primitives, 1932.03ms, 1.19045 Mprims/s, sah = 339.751 [DONE]
iteration 2: building BVH over 2300000 primitives, 1955.2ms, 1.17635 Mprims/s, sah = 339.751 [DONE]
iteration 3: building BVH over 2300000 primitives, 1924.15ms, 1.19533 Mprims/s, sah = 339.751 [DONE]
iteration 4: building BVH over 2300000 primitives, 1914.13ms, 1.20159 Mprims/s, sah = 339.751 [DONE]
iteration 5: building BVH over 2300000 primitives, 1909.3ms, 1.20463 Mprims/s, sah = 339.751 [DONE]
iteration 6: building BVH over 2300000 primitives, 1884.56ms, 1.22045 Mprims/s, sah = 339.751 [DONE]
iteration 7: building BVH over 2300000 primitives, 1948.36ms, 1.18048 Mprims/s, sah = 339.751 [DONE]
iteration 8: building BVH over 2300000 primitives, 1902.84ms, 1.20872 Mprims/s, sah = 339.751 [DONE]
iteration 9: building BVH over 2300000 primitives, 1893.38ms, 1.21476 Mprims/s, sah = 339.751 [DONE]
```

Fig. 18. BVH Result

Fig.18 shows that it has over 2300000 primiteves, and the run time is base on the quality is between 200ms and 2000ms. This demo only has 8 simple triangle primitives, and the run time is 17ms. so the conclusion is that BVH is very fast.

## V. Discussion

In the field of computational electromagnetics, ray tracing is widely used to predict the propagation characteristics of radio waves in mobile communication environments, and can identify all possible ray paths between transmission and reception in multipath channels. The following mainly discusses the principles and specific implementation methods in conjunction with relevant literature.

There are two types of methods for ray tracing, one is image method, the other is brute force ray tracing method. The image method is based on the law of reflection. First, it is assumed that equivalent sources (images) will be created when the ray of the transmitting antenna encounters objects, and the signal at the receiving point is the sum of all sources. The mirror image method is appropriate when the geometric structure is relatively simple and the number of reflections considered is not large. S.H.CHEN et al. used shooting and bouncing ray (SBR) method to model the radio propagation channels in tunnels. [9] In the image method, only the contribution of direct and reflected rays is considered, and projection and diffraction are ignored, which inevitably reduces the accuracy of prediction, but the advantage of this method is simple calculation.

The ray tracing method regards the electromagnetic waves emitted by the transmitting antenna as many rays, and its transmitting antenna is considered as a point source. In order to determine all rays from the launch point to the receiving point, all angles from the launch point must be considered, which is the elevation and azimuth angles in the spherical coordinate system. Models, due to the limitation of operation speed and computer memory, are divided into two types: 2D (low complexity) and 3D (high complexity).

The 2D model is a simplification of the actual situation. It is considered that the transmitting and receiving points have the same height, and the radio waves propagate in the same plane; or it can be considered that the 3D model is projected onto a plane. Therefore, the emission point is regarded as a unit circle, and the accuracy and calculation speed can be adjusted by increasing or decreasing the number of rays, that is, changing the angle of the sector.

The 3D model is relatively close to the actual situation. Now the transmitting antenna is non-directional, which means that the transmitting point is regarded as a unit sphere. The rays under the 3D model is much more complicated than that of the 2D model. The solution of this problem in the 3D model is to divide the unit sphere equally with the ray tube at the launch point (transmitting antenna). Therefore, each ray tube is required to have the same solid angle, and the wave front of each ray tube at the same distance from the emission point must have the same size and shape. S. H. Chen et al. used a triangular ray tube. [10] S. Y. Seidel et al. used the icosahedron to approximate the unit sphere. [11]

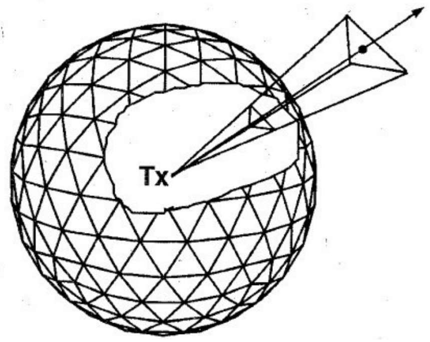


Fig. 19. Triangular ray tubes shot from transmitting antenna [10]

The calculation speed can be increased and the complexity can be reduced by setting the attenuation threshold and selecting the appropriate model.

In the ray tracing process, because the number of objects in the indoor environment is less than that in the outdoor, but the density is large, an algorithm for directly finding

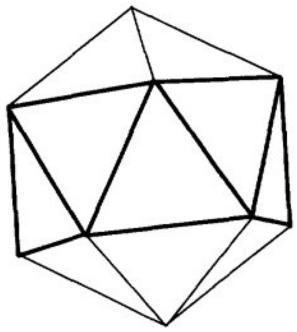


Fig. 20. A regular icosahedron that is inscribed inside a unit sphere. Rays are launched through icosahedron vertices and at intersection points of tessellated triangular faces. [11]

the meeting point of each object is generally used. The rays emitted by the transmitting antenna form reflected, transmitted or diffracted rays when they encounter an object, so the whole tracking process is a recursive process. To calculate the field strength of these new rays, the process of tracking these rays is equivalent to the tracking of the source rays.

In the actual tracking process, it is impossible and unnecessary to trace endlessly. Therefore, a limit is usually set according to the required accuracy. When the field strength of the ray attenuates to this threshold, the tracking of the ray should be abandoned, the depth of the recursion depends on the attenuation threshold of the abandoned rays. If the rays are received or run out of the room, the tracking should stop.

At the same time, as the number of objects encountered increases, the number of rays will increase exponentially. For general buildings, the number of indoor objects may be large, so the number of rays is very large. There are two commonly used tracking models. One is planar tracking, that is, two-dimensional tracking, which does not consider the ceiling and floor reflections, and transmits the transmitting antenna and receiving antenna and all objects vertically on a plane; the other is three-dimensional tracking. The height of the transmitting and receiving antennas, as well as the ceiling and floor reflections are considered. The former has a fast calculation speed due to the small number of rays being tracked, but the accuracy is relatively low; the latter requires a lot of calculation time, but the prediction accuracy is high. In practical problems, an appropriate ray tracing model can be selected according to actual needs.

## VI. Conclusion

In this project, the group studied the physical process of light transmission between two medium as well as other basic knowledge of raytracing. Python has been used to simulate the light transmission when it hits a smooth surface(between two media). Several test cases have been designed to realize the hypothesis and non-hypothesis. After the simulation, one group member(Weng) tried a raytracing demo using embree. However, the test cases designed in this project is not comprehensive and not very rigorous. The following issues has not being considered in a single-ray-one-surface-transmission simulation: 1. Numerical operation could affect result accuracy. 2. The order of operation may also influence the accuracy. 3.Some test cases are depended on the limits of numerical approximation of the machine(incident angle very close to 90 degree or 0 degree).

## VII. Contribution of the members

At the beginning of this Hauptseminar, the 4-person-group is divided into 2 working groups(Su Yankai and Ma mingyu in one, Liang Shaofeng as well as Weng Jiakang in another). The 2 groups are assigned with a same task: simulating the light propagation in different cases. In Su-Ma group, Ma wrote the light propagation algorithms and realized the test cases, Su helped visualized the result and designed the test cases. In Weng-Liang group, Weng wrote the light propagation algorithms and tried to realized the light propagation between 2 planes. Liang help designed the test cases and helped test the whole algorithm. Weng also attempted to run a simple simulation making use of embree, and successfully got a result. The 2 groups documented their working progress as they proceed their work. In final documentation phase, each member of the 4-person-group was assigned with a separate part, Su helped other group members polish their parts after they finished. After the documentation, Su and Weng made the slides and prepared for the presentation, Ma and Liang made the poster.

## References

[1] Malitson, "Refractive Index Database, [On the electrodynamics of moving bodies]," refractiveindex.info. Retrieved, June 20 2018.

[2] A. . Faick, C.A.; Finn, "The index of refraction of some soda-lime-silica glasses as a function of the composition," National Institute of Standards and Technology, vol. 322, no. 10, pp. 891–921, December 30, 2016.

[3] Wiki. Bounding volume hierarchy. [Online]. Available: https://en.wikipedia.org/wiki/

[4] K. Beck, Test-Driven Development by Example. Addison Wesley.

[5] D. Kuhlman, A Python Book: Beginning Python, Advanced Python, and Python Exercises, 2012.

[6] Intel. Embree overview. [Online]. Available: https://github.com/embree/embree

[7] "Iso/iec/ieee international standard - systems and software engineering – vocabulary," ISO/IEC/IEEE 24765:2010(E), pp. 1–418, 2010.

[8] H. Krekel et al. pytest. [Online]. Available: https://docs.pytest.org/en/stable/

[9] S. H. Chen and S. K. Jeng, "Sbr image approach for radio wave propagation in tunnels with and without traffic," IEEE Transactions on Vehicular Technology, vol. 45, no. 3, pp. 570–578, Aug 1996.

[10] ——, "An sbr/image approach for radio wave propagation in indoor environments with metallic furniture," Transactions on Vehicular Technology, vol. 45, no. 1, pp. 98–106, Jan 1997.

[11] S. Seidel and T. Rappaport, “Site-specific propagation prediction for wireless in-building personal communication system design,” IEEE Transactions on Vehicular Technology, vol. 43, no. 4, pp. 879–891, Nov 1994.