



# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Context of work . . . . .	4
1.2	Summary of project progress . . . . .	5
<b>2</b>	<b>Review</b>	<b>5</b>
2.1	History . . . . .	5
2.2	Application of Ray tracing . . . . .	6
2.2.1	Ray Tracing in Architecture . . . . .	6
2.2.2	Ray tracing in engineering . . . . .	7
2.2.3	Ray tracing in animation . . . . .	7
2.3	Scanline rendering . . . . .	7
2.4	Advantages and disadvantages . . . . .	8
2.5	Camera imaging principle . . . . .	8
2.6	Important References . . . . .	9
<b>3</b>	<b>Requirement analysis</b>	<b>10</b>
3.1	Functional requirement . . . . .	10
3.1.1	Position module . . . . .	11
3.1.2	Object module . . . . .	11
3.1.3	Preview module . . . . .	11
3.1.4	Produce module . . . . .	11
3.2	Performance requirements . . . . .	12
<b>4</b>	<b>Implementation</b>	<b>13</b>
4.1	Initial design . . . . .	13
4.2	Main page design . . . . .	14
4.3	Front-page function module . . . . .	14
4.3.1	Position module . . . . .	14
4.3.2	Object module . . . . .	15
4.3.3	Preview module . . . . .	15
4.3.4	Produce module . . . . .	16
4.4	Back-end structure design . . . . .	16
4.5	Objects implantation . . . . .	20
4.6	Materials and texture . . . . .	21
4.7	Texture . . . . .	22
4.8	Source of light . . . . .	22
4.9	Antialiasing . . . . .	23
<b>5</b>	<b>Software testing</b>	<b>23</b>
5.1	White box testing . . . . .	24
5.1.1	Unit test . . . . .	25
5.1.2	Death test . . . . .	27
5.2	Black-box testing . . . . .	27
5.2.1	Request . . . . .	27

5.2.2	Test method . . . . .	28
5.2.3	Example input and result . . . . .	29
<b>6</b>	<b>Responsibility distribution</b>	<b>30</b>

# FINAL REPORT

Group Lucky6

March 28, 2018

## Abstract

This report introduces entire procedures in implementing a lightweight raytracing software with the characteristic of multithreading processing ability. Fundamental geometry principles and essential background knowledge with the respect to object-oriented rendering algorithm are also particularly included, which makes the concept and idea to be more understandable for readers who do not have the relevant background. This report also presents the final raytracing software testing result and discusses the further research direction. Compared with currently available raytracing software online, this software is devised to build an adjustable flexible and lightweight raytracing data structure so the future modification in relevant characteristics will not require an entire re-modification.

## 1 Introduction

### 1.1 Context of work

The entire development cycle lasted for over 2 months, consisting of 5 main stages (the early-stage design, software structure design, the initial edition implementation, Functional iteration update, software testing). Some of these 5 steps are not conducted in sequence but are synchronous. For instance, the software testing is along with functional iteration update each time and software structure is adjusted every time after a new edition. The new function will be checked in order to program a practical template for the specific class structure. As a result, if which any further relevant function modification is required, it can be very convenient. Up to mid-late of March, the latest edition can fulfill the large majority of software requirements that will be described in the later. It provides a relative comfort GUI for users to input parameters and the system will respond an early-stage sketch within the ordinary users tolerant time.

On each stage, the development team has made some mistakes and meet many interesting and typical problems on technical side and teamwork side. Although those difficulties were conquered, they have representative meaning in raytracing software development and are worth considering.

On the stage of early-stage design, the main trouble is how to schedule the general plan, how to allocate jobs for each member and how to determine milestone to build the general structure of the program, which, fortunately, were addressed before software frame construction. The specific methods of dealing with the relationship issues were concluded in the initial report.

On the step of software structure design initial edition implementation, since being lacking solid computer graphics background, back-end group found it very complicated to design and implement a complete frame and scaffold in early-stage. Consequently, they decided to carry out a simple raytracing model and then constantly update and add features. But this resulted in quite sluggish

progress and nearly took 3 weeks to deliver the first edition. Another trouble is to review the 3- D vector calculation formula and implement it through the computer algorithm.

On the stage of functional iteration update, having addressed some problems with respect to the basic concept and necessary math formula transformation. The main difficult shifted to object virtualization (intersection process, the source of light position and background texture). Objects in the pre-set scene are made of different materials (Lambertian. metal, glass) Rays show diverse effects of its encounters with these objects. In addition, depth of focus is a very important problem that worth thinking about. The step of software testing and image quality improvement synchronizes with two steps above. However, along with the thoroughgoing study on ray tracing, quality becomes an urgent problem. The improvement of image quality is bounding to the decreasing speed of rendering and processing. Multithreading in data processing and multiple times rendering in a single pixel are introduced to remedy the limitation in performance and speed.

## 1.2 Summary of project progress

Generally speaking, this ray tracing software achieves the following functions: shape selection, color design, position setting, material design, a source of light setting, texture design, camera setting. Then the system will return a 3D shape sketch without color rendered. The final image will be ant aliased and processed through Perlin noise before delivered to the users. Although the majority of functions are fulfilled in this lightweight raytracing software, there are much more practical functions can be added and there is plenty of room for current techniques to be optimized.

Firstly, motion blur depth of focus properties causes fuzzy image and degree of fuzziness does not increase with distance but is in the same degree that ignoring the distance factor. Then, the up-down rotation of the camera. The meaning of rotation here is no matter on which a camera is set, there is a limitation for a camera to move its view upward or downward. In addition. Besides, as mentioned above, high speed in rendering requires. Multiple rendering of a pixel point. In spite of invoking multithreading, the performance still can be improved by means of using bounding volume hierarchies (Ray tracing in one week). The topic of adding more source of light instead of a single source of light and increasing the color of light instead of only white color available is another implementation direction. Also, by means of literal programming, how to design and construct a complete raytracing algorithm struct and create some template with the purpose of minimizing the size and effect of modification, is still undergoing. This raytracing software aids to understanding the basic concept of computer graphics. But, indeed, it is a just a start or a key to explore this interesting and beautiful universe.

## 2 Review

### 2.1 History

The study on raytracing started from about 1950s but the research on light and view began far earlier. Plato (428-347 B.C) claimed that the eye is a lantern and seeing id a meeting of inner and outer light while Aristotle (384-322 B.C.) presented that the eye is a darkened chamber awaiting the light. He first described the camera ob scura. Only after Alhazen pointed the definition of the optic path, the concept rays have been introduced to the public. Later, Newton and Gauss played important roles in modern views of light and the premises of ray tracing. Newton calmed that corpuscles emitted from shining objects and Gauss demonstrated raytracing through lenses. Rene

Descartes (1967) introduced ray tracing back in 1637 the idea of tracing (or plotting) light rays and their interaction between surfaces. He applied the laws of refraction and reflection to a spherical water droplet to demonstrate the formation of rainbows. As a matter of fact, although successful experiment has been done in the digitizing of photographs, Arthur Appel (1968) emphasized on the importance of illusion of reality and the calculation process. In his paper entitled some techniques for shading machine ” rendering of solids, Arthur collected some recent experimental data and result in the automatic shading of line drawings in order to generate virtual images of objects in the scene. By doing this, the evaluation of the cost of generating such images can be calculated and compared. For the reason that those objects composed of flat surfaces on a digital plotter, the pen-plotters was developed.

Following the Arthur Appels research, first global illumination model by Turner Whitted (2005) and he pointed out: in order to accurately render a two-dimensional image of a three-dimensional scene, global illumination information that affects the intensity of each pixel of the image must be known at the time the intensity is calculated. In a simplified form model is an improved illumination models for shaded display, said by Abrebt. For achieving the quick response and high performance, he suggested to store the rays into a unique dictionary data structure tree which makes it possible for each ray extends from the start point to every surface that is intersected and even the light sources. Most importantly, Turner Whitted was the first to describe how to extend Appels rendering theory for more advance and better rendering algorithm. He extended Appels ray-tracing algorithm model of launching rays to merge reflection and refraction calculation together. In other words, An objects color is influenced by lights and other objects in the scene Simulates specular reflection and refractive transmission

Both Turner Whitted and Arthur Appel pointed out a key word-Global Illumination, a rendering effect, which generally is very expensive to simulate and difficult to simulate efficiently. People can see things because light emitted by light sources, such as the sun, bounces off of the surface of objects. if light rays bounce only once from an object to hit the viewers eyes, It can be described as direct illumination. But if rays emitted by a light source Bounce of several surface of objects and even bounced off a single object for multiple times before views eyes receive this ray. This is what public call indirect illumination for the reason that light rays follow complicated paths before reaching the eye. Some surfaces of an object are not exposed directly to the source of light which means the effect of indirect illumination is visible if an observer look from the top of the object. But they are not completely black. This is because the place that are shaded still receive some light that caused by indirect illumination phenomenon. Direct illumination and indirect illumination make up a significant part of global illumination to produce realistic image. Meanwhile, these two effects are frequently invoked and used when the ray tracing software is Scratch pixel (2017).

## **2.2 Application of Ray tracing**

Ray tracing has been used in all walks of life. Its appearance has changed the traditional way and has greatly helped the development of a lot of fields. The following are the applications of ray tracing in several fields.

### **2.2.1 Ray Tracing in Architecture**

When architects are submitting designers to customers, the rendered images they submit are very important. In the past, architects could only rely on hand-painting. Obviously, traditional render-

ing methods are difficult to produce realistic lighting effects. Most CAD programs can accurately calculate the object model, but still cannot simulate light. Although some modeling programs allow architects to introduce light into the graph and even specify the position, direction, color, distribution, and highlights and shadows of the light, they still cannot simulate physical scenes where the object interacts with light. In order to accurately simulate the true behavior of light in a particular environment, we must consider all the light in the environment, as well as their reflection, refraction, scattering, and absorption. Therefore, light modeling is very important in this area (Schmitter et al., 2002).

### **2.2.2 Ray tracing in engineering**

Ray tracing can take into account all the light in the environment, which is called global illumination. Global illumination is a model that can accurately simulate the behavior of light in a real physical environment, so it is very useful in the field of engineering, such as solar energy. Engineers can use ray tracing to render real images to determine standards such as lighting levels, brightness gradients, and visual performance, and to better analyze the distribution, direction, and radiation of light to better capture solar energy (Cheng et al., 2014).

### **2.2.3 Ray tracing in animation**

The advent of ray tracing technology has made great progress in animation design. For a long time, traditional animation works were done by hand-painting, and the movement effect was simulated through a series of complicated frameworks. Ray tracing can make animations more vivid, such as reflections and shadows, which is very difficult for traditional hand-painting methods. In addition, it also makes the image more realistic, like real photos. Therefore, the presented animation has better effects and the production process is also easier and more efficient (Glassner, 1988). To sum up, none of these applications can be separated from ray tracing. It is because ray tracing can take into account all the light in the environment, plus its simulation of reflections, refractions, and shadows, to make the effect more realistic.

## **2.3 Scanline rendering**

Scanline rendering is a set of techniques and algorithms that are rendered by a row, rather than by polygon-to-polygon or point-to-point mode. What needs to be rendered polygons according to the y coordinates of the first vertex in order, and then use the scan lines and the list of the front line or polygon intersection calculation image each scan line, then use the intersection of the scan lines with the polygons in the list to calculate each line or each scan line of the image, and the list-discarded polygons are updated as the active scan line gradually counts down the image.

One advantage of this approach is that there is no need to transfer all the vertices of the main memory to the working memory, only intersect with the current scan line boundary constraints of vertices need to read the working memory, and each point data read it only once. The speed of the main memory is usually much lower than that of the central processing unit or the cache, which can greatly increase the speed of operation by avoiding multiple access to the vertex data in the main memory.

## 2.4 Advantages and disadvantages

Raytracing has many advantages and weakness. This is an idea of plotting light rays and tracings interaction between camera and surfaces. Back to raytracing history, pioneers used basic ray casting algorithm to extend light. When the ray arrived the surface, it would separate to three new kinds of rays, reflection, refraction and shadow. Because of the mirror-reflection, the reflected ray continued to intersect. Also, the surface depends on the results, the refraction ray may enter or exit the material so that produce the output about the point and light. Therefore, raytracing owns some advantages with the help of tracing renderers: raytracing can do the perfect simulation of lighting realistically, which is better than scanline rendering and ray casting. Raytracing algorithm can simulate natural result which can lead to accumulate output of study. And reflection or shadows are difficult to simulate other methods to accomplish. Finally, it is simple and convenient to implement and produce the impressive visual results. Ray tracing properties of light are influenced by pixel color intensity. Thousands of calculations need to be generated to determine every pixel color.

Conversely, raytracing also exists many weaknesses about methods and results. Firstly, the performance is at a low level, some details cannot be showed clearly and concretely. Scanline algorithms and other algorithms use data coherence to share computations between pixels, while raytracing normally begins with each eye ray separately. This separation offers many other strengthens, such that it could hit more rays to perform separation and improve image quality at last. The advantages of accurate inter-reflection cannot make up the ordinary output. In this aspect of separation, other methods upon raytracing for certain parts of algorithms, can give the better results.

Here are some influences about producing the weakness and continuing advantages. If objects are intersected the light, some form of light or rays may go through if the object is transparent to some degree. More complicated render algorithms include this notion detect not only if light returns, but the color intensity of the light.

## 2.5 Camera imaging principle

The pinhole camera, also known as a camera box, is a simple optical imaging device that encloses a box. On one side, it is a small hole that produce an image of the outer space. On the other side, the box has the straight line of light.

In the natural environment and in the daily life, objects using this principle are easy to find, and people all over the world have been observing them since ancient times. The possible earliest description of this observation may have originated in the 5th century BC and it was written by the Chinese philosopher Motti. In the Western Hemisphere, Aristotle asked without asking any satisfactory answer why the sunlight passing through the quadrilateral did not produce an angled image, but formed a circular image in the 4 BC. The day passed through the sieve, it shows that image of the food, the leaves of the tree, or the gap between crossed fingers forms a crescent on the ground. In 10 AD, Arab physicists and mathematicians studied the inverted image formed by a small hole and pointed out the straight line of light. Until the Italian artist and inventor Leonardo da Vinci developed the first detailed description of the pinhole camera, he used it to study the perspective.

Initially, the camera with black box was a room that projected the image onto the wall through an opening in the opposite wall. It was used to observe the solar eclipse and to check the projections. Later it became a portable instrument with a focusing lens. Such instruments are often used as



drawing aids and at the dawn of the history of photography they formed the basis of camera construction. Pinhole cameras eventually also applied to modern science. In the middle of the 20th century, scientists discovered that it can be used to take X-ray radiation and gamma rays absorbed by ordinary lenses. As a result, the pinhole camera then found its own way in spacecraft and space.

In the late 19th century, photographic technology became more mature in photography, because the photographic technique noticed this perfect and clear image when making gentle contours. The pinhole camera was later abandoned until several artists began using it in their experiments in the late 1960s, which awaken people's attention to this simple camera device. It continues to this day. The image in the pinhole camera which is created was based on the straight line of light. Each point on the surface of the illuminated object reflects light in all directions. The hole allows a certain number of these rays to continue their travel until they meet the projection plane where they produce an inverted image of the object. Therefore, this point is not reproduced as a point, but as a small disc, resulting in a slightly defocused image. This description shows that the smaller the hole, the clearer the image. However, light is essentially a fluctuating phenomenon, so once the size of the opening is commensurate with the size of the light wavelength, diffraction occurs. In other words, if the hole is too small, the image will be out of focus.

The image created by the pinhole camera has some features that we could not find in classic lens photography. Since the process requires a central projection, the image in the pinhole camera presents an ideal viewing angle. Another special feature is the infinite depth of field. In a single photograph, objects can be captured with the same sharpness regardless of whether the object is very close or far away. The pinhole camera angle is very wide. However, the light takes longer to reach the edge of the negative than the center, so the picture is less exposed along the edge and therefore darkens.

Some of the disadvantages of the pinhole camera are the amount of light allowed by the small aperture, which complicates and sometimes completely prevents photography of the moving subject. Exposure times are usually measured in seconds or minutes, but in poor light conditions this can be hours or even days.

## 2.6 Important References

- «Ray Tracing in One Weekend» This book is easy to understand and very helpful for beginners. By reading this book, we have understood some basic knowledge about rays, simple camera, background, and adding objects.
- «An Introduction to Ray Tracing» The first chapter of the book is an overview, which describes some theoretical knowledge, and gives a deeper understanding of ray tracing. The second and third chapters have main algorithms for implementing various surfaces, and then gradually implement these surfaces based on the previously learned code architecture and corresponding detailed reference papers. The fourth and subsequent chapters begin to introduce knowledge of reflection models, lighting models, and textures. The book introduces the mainstream algorithms and core algorithms that were implemented on various surfaces at the time. The book also provides enough references, and then based on these papers, all the surfaces mentioned in the book can be drawn. Through the study of this book, tracing out various surfaces becomes easier (Elsevier, 1989).
- «Ray Tracing from the Ground Up» The purpose of reading this book is to learn more about lighting models, reflection models, textures and materials. After learning, we have a

better understanding of the lighting model. When the geometry of the object is determined, lighting determines the display of the entire scene. Therefore, the generation of realistic graphics depends on how to establish a suitable lighting model. In addition, we have also studied the Phong lighting model, which contains three kinds of lights. Ambient light is a type of light where multiple reflections of light generated by adjacent objects eventually reach equilibrium. It can be approximated that ambient light in the same environment has a uniform light intensity distribution. When light strikes the surface of a relatively rough object, the lightness and darkness of a certain point on the surface of the object do not change with the position of the observer. This kind of scattering is equivalent to all directions. This phenomenon is called diffuse reflection of light. Specular reflection light refers to the light produced by light striking a relatively smooth surface. It is characterized by a high light area on a smooth surface (Suffern, 2016). Additionally, by learning the knowledge in this book, we have a deeper understanding of materials and textures. Textures are more biased toward "graphs," and materials are more toward "attributes." For example, to process the same cube model, texture information can be thought of as a map, such as a texture map of wood or a texture map of marble; adding material information can be thought of as adding properties to the cube. Attributes mainly refer to the reflection coefficient, refractive index, such as the attributes of wood or the attributes of marble. From another point of view, the textured model is static and superficial and will not change due to changes in the external environment. However, the material-added model is dynamic and essential, and it can be more real when the external environment changes.

### 3 Requirement analysis

#### 3.1 Functional requirement

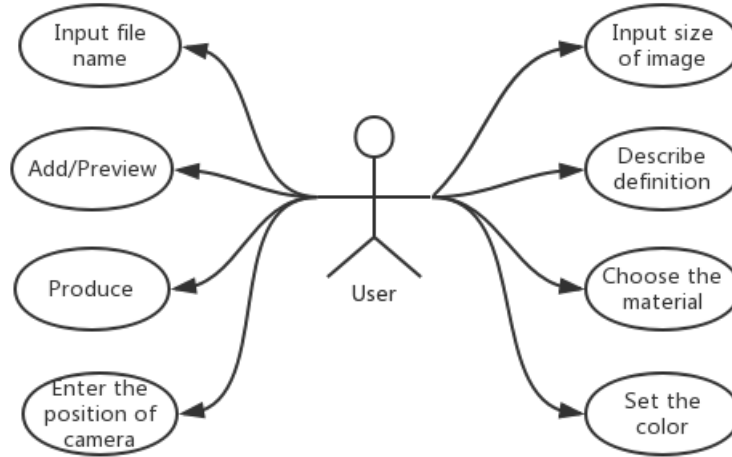


Figure 1: Front use-case diagram

### 3.1.1 Position module

This module includes the image's properties, which contain name, definition, length and width (respectively x and y). This x and y only represent the size of the image. And if need, the definition represents the details of the image. The data separated by position and sight angle. According to the rectangular coordinate system. The three blanks should be entered by the position of looking from. The field of view is needed to input the angle of sight, whose value range is from 0 to 360. In particular, it changes as the situation differs.

### 3.1.2 Object module

In this module, shape and material are the two directions to present. The first is shape, which includes the Sphere, rectangle, and light. They are the main objects we mainly studied. On the other hand, aspects of material and texture include Marble, Metal, Glass, Pure Color and Checker Texture. These materials and textures will be used when the shape of Sphere is selected. All these classifications are showed in seven tab-controls. If the shape of a sphere is chosen, only one material should be corresponding to Sphere. And the other is pure color by default.

- Sphere  
It contains the position of the sphere. X, Y, Z represent the physical position, and radius represents the size. Correspondingly, the materials and textures should be chosen.
- Rectangle  
It is also named Cube. X Y, Z represent the cube's length, width, and height. Correspondingly, the materials and textures should be chosen.
- Light  
The light has four properties. X Y, Z show the color of origin, which is like the light source. The object doesnt have to be lightened.

### 3.1.3 Preview module

During the users are entering the values in position module, the below screen will be creating a new scene. After the users enter the values in object module and click on the Add button, the preview image will be shown on the screen. Users can also interact with it through the mouse. If the users are not satisfied with the preview, they can delete the object and read it.

### 3.1.4 Produce module

If the users are satisfied with the preview, the can click the button, then the data will be sent to the backend.

Here are the main running process of users. Users can input the size and position of image and camera in the first module and set file name themselves. And then if choosing the material rather than light, users need to determine which material should be selected, which color should be set. Besides, the camera's looking destination is located, which is really helpful for users. They only need to set their sight of position.

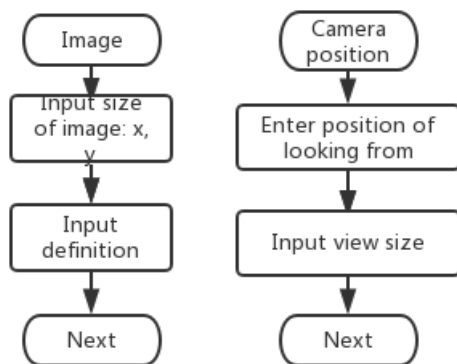


Figure 2: Flow chart(1)

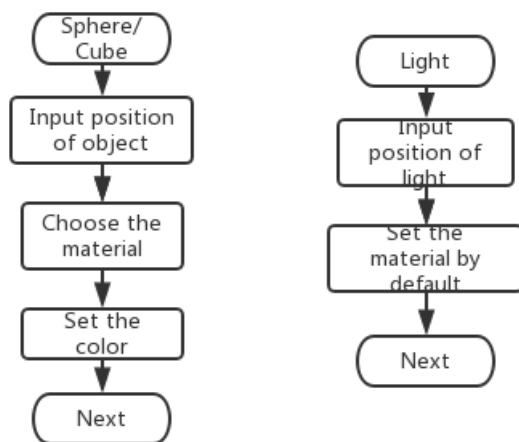


Figure 3: Flow chart(2)

### 3.2 Performance requirements

- Usability

In today's Internet applications, ease of use is becoming more and more important. An

application with good usability can bring better user experience to users. This application should meet the following requirements to improve ease of use that one button to complete without excessive operation and operation method is consistent with mainstream applications in the market.

- Good interactivity

When developing and creating a website page, this site should have a simple web page layout, user-friendly function design, and intuitive and convenient setting of the function navigation bar. Each function design in race tracing should be very consistent with the facts. Based on user-friendly and user-friendly, this website can be used very conveniently.

- Extensibility

With the update of technology, market demand and user demand are constantly changing, so an application must have extensibility, to facilitate the subsequent iteration and update, the application is iterative development, the function is also plug-in, so it has Excellent extensibility.

- Maintainability

Each piece of software has a variety of problems. Most of these problems are due to bugs after development. Improper use causes damage or compatibility problems. Therefore, the application is maintainable so that it can be timely if problems occur.

- Stability

This application has good stability to avoid crashes caused by excessive instructions or data. Therefore, a proper architecture should be used for development, and stress tests should be used during testing to reduce the breakdown rate as much as possible.

## 4 Implementation

### 4.1 Initial design

So how to build this a lightweight raytracing software with multithreading process feature? A lightweight raytracing software here means the system can get a cool image with no API and logic in code is not to complex to understand for the reader. There are also some templates or base classes that are created for benefit further modification and change. Multithreading can accelerate the color rendering performance and boost data processing efficiency of the software system.

The chart below shows the entire development cycle for this lightweight raytracing software. Do you remember there are some problems that plague the development team on each step written in the previous introduction section? Effective and useful solutions are summarized as well that will be listed in later part corresponding to each problem.

On the initial step, the development team leader made a serve mistake in group division as he arranged the team members who are not very skillful coder and none of them is very familiar with GUI programming and web application implementation. This Is because being in view of back-end algorithm implementation is the key part to the overall software, group leader left 3 more skillful members in back-end development group. However, progress has been slow in front-end development. The reason seems to be obvious, teacher is very important for students to learn

Table 1: Time Schedule

Number	Goal	Implementation time
1	Collect basic knowledge and information about ray tracing	20, January - 8, February
2	Implement the basic class vector object and ray	9, February - 16, February
3	Implement different material	17, February - 24, February
4	Implement different texture	25, February - 4, March
5	Implement multithreading	5, March - 12, March
6	Implement Python Server	13, March - 20, March
7	Finish the back-end report and related files	20, March - 27, March

something new which is equally applied to every area. Being lacking an experienced team member to lead front-end development group to deepen their understanding of web site designing and developing, they made slow progress. A quick member re-allocation was made and Qiaowei Ni then took responsibility for Front-end programming training. Frequent and efficient team communication twice in a week can not only reconcile opinion differences in developing. Besides weekly progress report plays an important role in monitoring project progress and motivate some team member who wants to be lazy or drop out.

## 4.2 Main page design

At first, a client was decided to be designed. However, a client needs to develop different versions for different operating systems. In addition, the client is not a lightweight choice, because it always takes up more storage space and memory. To avoid above-mentioned problems, we decided to use a web page as GUI. There are several advantages of using a web page as a GUI. One the one hand, the web pages are lightweight enough to meet requirements. On the other hand, the operating system does not need to be considered, and it only needs to consider the version of the browser. To make the process convenient and efficient, Bootstrap was used to develop the GUI. Bootstrap is the world's most popular library of GUI components for developing responsive web-based and mobile device Web projects. It is a set of open source toolsets for HTML, CSS, and JS development. A web page can be developed by using its Sass variables and lots of mixings, responsive grid systems, extensible prefabricated components, powerful jQuery-based plug-in systems.

## 4.3 Front-page function module

Function module can be divided into five parts, which are position module, object module, add module, preview module and produce a module. Each module has specific functions, the following is the detailed description.

### 4.3.1 Position module

In this module, users can enter the information of the image, which includes name, size, and definition. They can also enter the position of the camera, which means where the camera looks from. In addition, the value of a field of view also needs to be entered. All coordinates are determined by the three values of x, y, and z, so there are nine input boxes in total, and each input box contains a placeholder, which shows grey words, to prompt users what need to be entered.

### 4.3.2 Object module

The main function of this module is to select the object type and enter its properties. The initial version is to display all the objects and their properties on one page, and the user enters the value in the corresponding input box according to their requirements. However, there have been several problems. One is that all the input is displayed on the same page, making the entire page appear bloated; The other problem is that different objects have the same or different properties, if they are all displayed on one page, the users will be misled, which is easy to make mistakes for users; The third problem is that if they are displayed on the same page, the difficulties of acquisition and addition of data will be increased. In order to avoid the above problems, li tags and tags are used to distinguish three different objects. By using this method, it is possible to click on different object names to show their unique properties, thus avoiding confusion with the properties of other objects. Different objects have different properties. The following explains one by one according to three different objects.

For the sphere, the properties that need to be entered include the coordinates of the center of the sphere (x, y, and z), and the radius. Four input boxes are provided to the user to enter this value, and each input box contains a placeholder to indicate what users need to enter. There are five types of materials or textures belong to the sphere, which is marble, metal, glass, pure color and checker texture. Among them, marble and glass have no sub-attributes, but metal, pure color, and checker texture require a sub-attribute, which is color. In order to distinguish five different materials, li tags and tags are used again. Three input boxes are used to input three values of RGB.

For the rectangle, the properties that need to be entered include the coordinates of two diagonal vertices because a pair of diagonal vertices can define a rectangle. Six input boxes are provided to the user to enter this values and each input box contains a placeholder to indicate what users need to enter. The rectangle has five types of materials or textures, which are marble, metal, glass, pure color and checker texture. Among them, marble and glass have no sub-attributes, but metal, pure color, and checker color require a sub-attribute, which is color. In order to distinguish five different materials, li tags and tags are used again. Three input boxes are used to input three values of RGB.

The properties of the light are radius and center position (x, y, and z). Four input boxes are provided to the user to enter this value, and each input box contains a placeholder to indicate what users need to enter.

### 4.3.3 Preview module

Preview module aims to allow the users to quickly sketch out a scene and verify that their scene is as expected so that the users can generate the correct results. Users need to enter values at first, then add the objects, and finally click the Add button, after that, scenes and objects will appear. The user can also adjust the viewing angle and distance by using the mouse.

As mentioned above, different pages are displayed for different objects so each page has a dedicated Add button. After the user enters the value, the added number of each object will be displayed in the lower area by clicking the button. At the same time, the data in the upper area will be cleared to allow the user to create the next object.

The Add buttons are used in both preview function and production function. The Add button is used for previewing functions and producing functions. The Add button retrieves the parameters

of each object by calling the instruction (`getElementById`) in JavaScript. For the preview function, this button also calls the function (`initSphere` and `initRectangle`) to generate a preview image and passes the obtained parameters to the function. In order to implement this function, a scene must be constructed first. An open source 3D graphics JS engine, `three.js`, is really helpful. The render, initialization performance plug-in, controller, window change trigger function is needed to be defined in JavaScript. All of these are used to make a display of objects and user interaction successful.

The position of the camera is set at (0, 0, -15). The closest distance from the camera to the origin is 5, and the maximum distance from the camera to the origin is 1500. To make the scene more vivid, a default object is set in the scene. It is a cube with a length, width, and height of 500. The color of this object is grey and its position is (0, -500, 0). It can be used as a ground in the scene, which can make the direction of the object more clearly. Since there are some differences between the value of the constructor and the user input, some calculations are needed to get the result. For the sphere, the radius and the ball center coordinates can get directly. However, for a rectangle, the only available value is the coordinates of the two diagonal vertices. In order to know their length, width, and height, the absolute value of the difference of coordinates needs to be calculated. In order to get the position coordinates, the sum of the coordinates needs to be divided by two.

#### 4.3.4 Produce module

If the users are satisfied with the scene, they can generate the scene by clicking on the Produce button. In this process, how to get the value entered by the users is very important. The value entered by the users can be obtained by a JavaScript method, `getElementById`, since each input box contains a unique ID. After these values are obtained, string splicing can be performed, and then spliced strings can be sent to the backend.

### 4.4 Back-end structure design

During the implementation period, the first technique difficulties is when the software architecture was built. At first, this project aims to design and construct a complete functional framework for each ray emitted for the source of light. For achieving this goal, Various factors should be taken into consideration but due to the complex relationship between each ray with the object. The workload for building such complete structure requires solid background in computer programming skills and deep understanding in computer graphics. Furthermore, if the plan is to design a template function for each ray, the overall amount of calculation is relatively large and most importantly light is reflected in every possible direction which means there is countless likelihood for a light. It can be a waste of resources. Moreover, difficulty in building a satisfactory initial framework obstacle the project progress and After a brief and urgent discussion, target was changed back to implement a simple raytracing structure that has serval advantages which will be talked about later;

Also, object-oriented programming (OOP) is a programming technique based on the concept of "objects" and class, which may contain data and methods. In OOP, computer programs are designed by making them out of objects that interact with one another.(Kindler, E.; Krivy, I. (2011) 313343) Depending on this situation, when objected oriented techniques is used, abstract base classes are defined for general entities (e.g., a object abstract base class defines the interface that all geometric shapes must implement and actually source of light is an object which has



different properties or characteristics from ordinary geometric objects). In MATT PHARRs(2016 ) complete ray tracing models, he listed about 10 base class: Shape, Aggregate, Camera, Sample, Filter, Material, Texture, Medium, Light, Integrator. In fact, in terms of ray tracing algorithm designing, the number of base classes division is based on the degree of detail of functional modules. Their relationship in quantity is negatively correlated in some extent. For example, according to Wikipedia definition, the entire raytracing consists of two main parts (various effect of intersection between lights and objects and information collection on image plane). But too simple division results in not very easy work in defining some special classes (like vector, texture). Peter Shirley also tried to make his ray tracing model as simple and clear as possible in his book entitled Ray tracing in One weekend. Simple code and brief class division is particularly helpful for the implementation of the basic framework and readers understanding. However, Increasing the number of modules appropriately enables Implementation of each of each abstract class more easily when the system functionality is extended or new feature is added.

Therefore, this ray tracing software compromise the number of classes and the complexity of the algorithm and, strictly speaking, there are 6 base classes that are defined, which can be found in the chart.

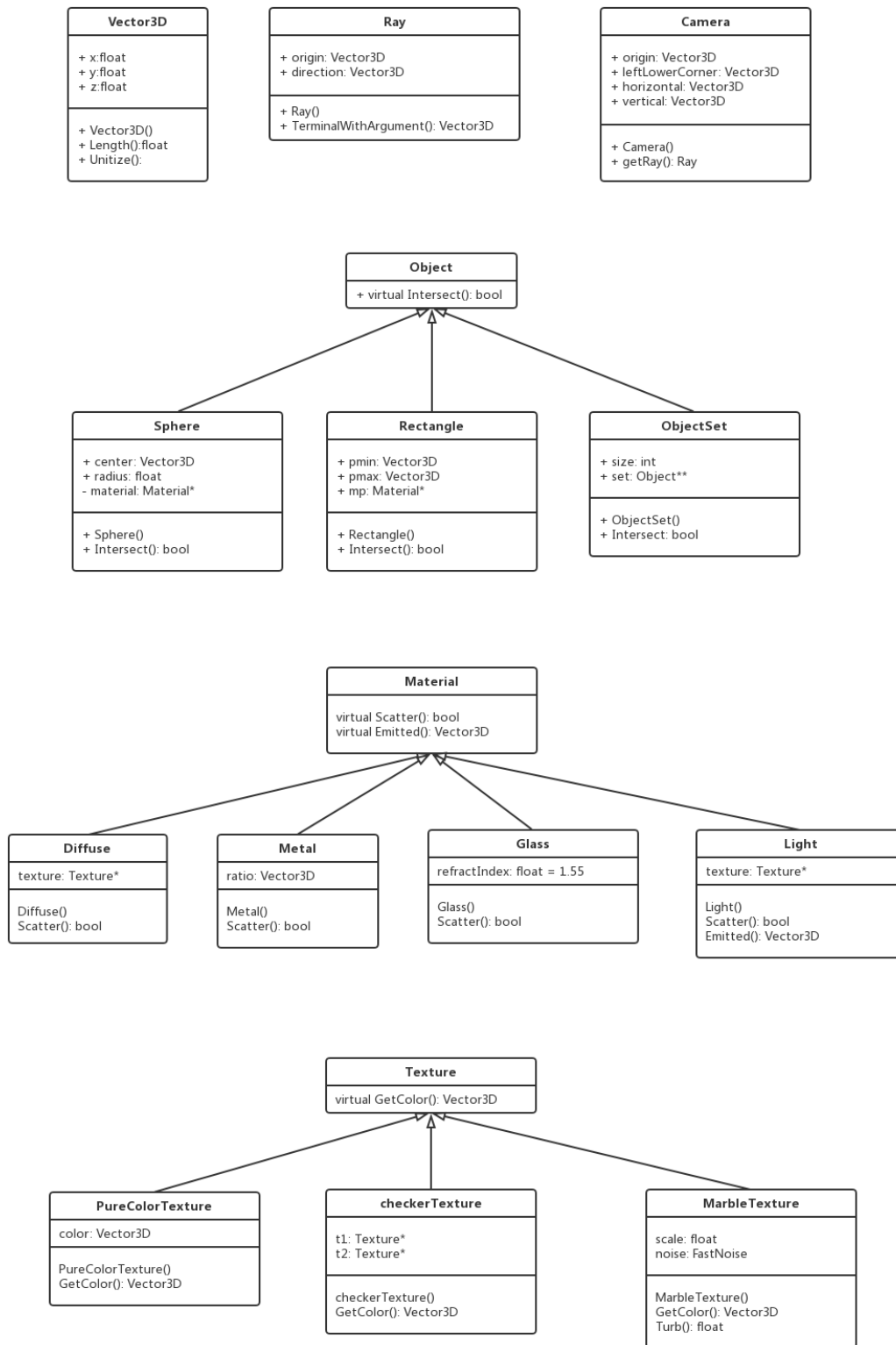


Figure 4: Class diagram

A camera class can be assumed as virtual eyes(receptor) but actually, it is not a point receptor and it will be discussed later when the depth of focus is introduced. It is the starting from of each ray which means the image produced is based on where the camera is located. Backward tracing method is applied to this point instead of forwarding tracing. The main difference between forward tracing in nature and the backward tracing is the direction of tracking path of light. To be more specific, system track rays from the virtual eyes to the objects rather than tracing light source from objects to eyes, which is the reversed direction of what happens in the natural world. There are three obvious disadvantages of Forward ray tracing. In nature, rays are normally reflected in every possible direction and thus very few amounts of rays will hit the eyes eventually. Because before the image is produced, the number of rays that hit the camera lens is unpredictable. The whole rays emitted from a source of light need to be calculated as well as their intersections with the geometric shape. Forward ray tracing is possible in terms of theory as it simulates the way light travels in nature on a computer but is not efficient or practical. In a seminal paper entitled An improve illumination Model for Shaded Display and published in 1980, Turner Whitted wrote: In an obvious approach to ray tracing, light rays emanating from a source are traced through their paths until they reach viewer. Since only a few will reach the viewer, this approach is wasteful.

In view of those drawbacks of forward tracing above, backward tracing provides a convenient solution. If a ray intersects with an object then the number of rays will cast shadow ray from the hit point to the scenes light, which informs the system that the original hit point is in a shadow.

Accordingly, the camera class is the foundation of ray tracing structure, as it determines the direction of the rays. Furthermore, the mode of projection and angle of view are both relevant to cameras position. Moreover, the position and angle of the camera in this program is adjustable which requires one more parameter called Standard\_Upward\_View. This new parameter set a limitation to the axis from starting point to view direction. It means When the direction in which the camera emits light is no longer the same as the parallel to projection perspective, the camera itself will form a new a plane that is not parallel to the imaginary canvas. However, the angle at which the camera rotates up and down can be limited by the. Standard\_Upward\_View. This. Therefore, upward limit direction can only be vertical with standard three-dimensional vector set in the vertical virtual scene.

Secondly, a clear and specific implementation of the 3D vector has a significant influence on connecting each base class with the image plane together. This is because every object in the virtual scene has its own position that corresponds to the standard geometric coordinate. Besides, every ray emitted from a camera can be described as a ray function  $p(t) = A + t*B$ . Here  $p$  is a 3D position along a line in the virtual scene.  $A$  stand for the starting point of this ray which can be approximately considered as a position of camera and  $B$  is the ray direction.

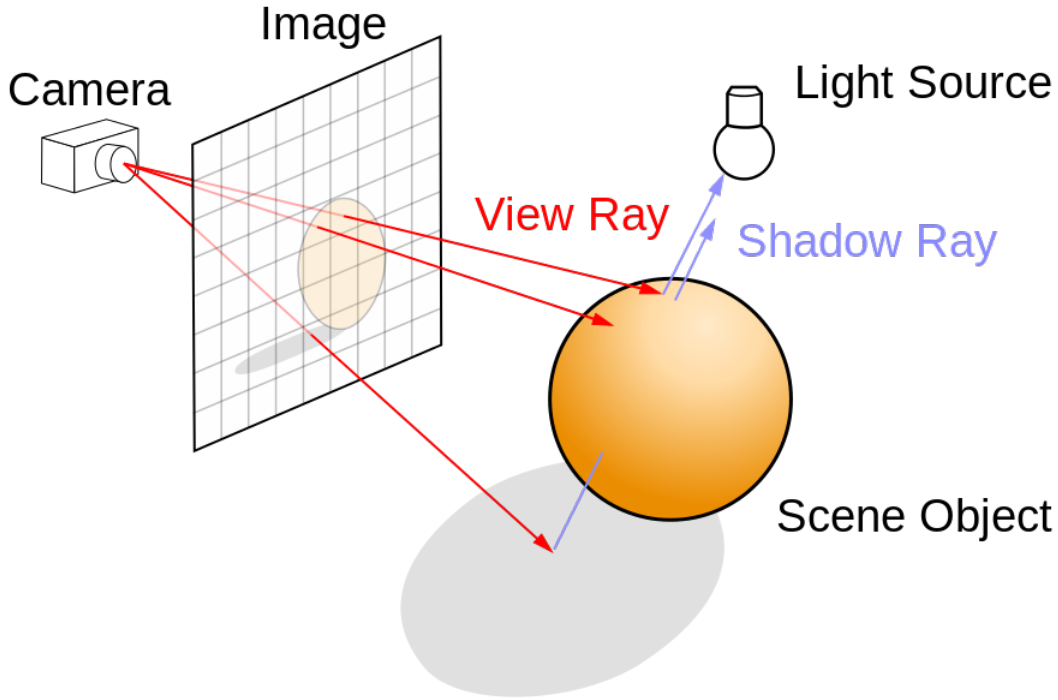


Figure 5: Raytracing base model from Wikipedia

The parameter  $t$  is the most vital factor in determining the direction of light from the same starting point. In many systems these vectors are 3D but these vectors can be implemented with for parameters (3D plus a homogeneous coordinate for geometry). Tom Dalling(2014) claimed that This four-dimensional space is called "projective space," and coordinates in projective space are called "homogeneous coordinates.". When a projector is projecting a 2D image on the screen. It is not difficult to define two-dimensional coordinate of the projected image the extra dimension is the distance from the camera to the imaginary scene which is isolated from the coordinates in three-dimensional space in the virtual scene. So, the system can collect the color information and render color directly.

These vectors can also represent color as X, Y and Z can represent red, blue and green light respectively and they can form RGB color model. It seems that the additional homogeneous coordinate will not be invoked in RGB but actually the homogeneous parameter can be responsible for transparency adjustment of the image. Three-dimensional Coordinate (X, Y, Z, W) map to RGB color model plus transparency adjustment (red, green, blue, the degree of transparency).

Operand overloading is a crucial step to 3D geometric vectors class construction. In this part, construction emphasizes on realizing the implementation of vectors cross and dot product by means of overloading  $*$ ,  $·$ . After overloading those base vector operations, the next stage is to build a connection between rays from a camera and virtual objects.

#### 4.5 Objects implantation

It is easy to create serval objects in the virtual scene but the problems are how these objects can be seen and what to do if there is an overlapping between two objects. Besides, what is different with sphere implementation and rectangle implementation? The first step is to create an object

list(object\_List) to store each object. A sphere can be constructed when its center position is determined and its radius of the sphere is set while It requires 6 different surfaces to make up a rectangle. So how to let the system know this is a rectangle, not 6 isolated surfaces. In fact, the solution to let the system know the rectangle or let eye recognize the overlapping object is quite similar. The ray tracing algorithm takes an image made of pixels and for each pixel, it emits a ray into the image plane. As a result, following the path of light when a ray intersects more than one object, the object whose intersection point is the closest to the camera, will be selected and thus the factor  $t$  can be calculated based on hit point coordinate and this rays representation( $p(t)$ ). Furthermore, the  $t$  obtained above is the minimum  $t$  factor and the maximum number can be computed through the fundamental formula Intersection equations of spheres and lines in three-dimensional space.

$$\begin{aligned}\|\mathbf{o} + t\mathbf{d} - \mathbf{c}\|^2 &= r^2 \\ \|\mathbf{d}\|^2 t^2 + 2\mathbf{d} \cdot (\mathbf{o} - \mathbf{c})t + \|\mathbf{o} - \mathbf{c}\|^2 - r^2 &= 0\end{aligned}$$

Figure 6: Equation of Ray and Surface Intersection

Similarly, the system only needs to get the relative position of the rectangle diagonal and then can produce a specified rectangle in the scene. As the coordinate of the rectangle is available, the value range of  $t$  can be calculated separately in X, Y, Z three coordinate by connecting the starting point of rays and the vertex of that rectangle. Therefore, All the rays that come into contact with a specified object are grouped by the  $t$ s scope and stored in a dynamic Intersection array. At this point, playing an auxiliary role, special struct called record is developed for recording the basic information of each intersected ray. It is a hit if  $x_0 < x < x_1$  and  $y_0 < y < y_1$ .

#### 4.6 Materials and texture

In previous section description, while In the process of creating objects and making them visible, reflection function is implemented potentially in advance. Because the reflection is a more easily simulated effect. The key is to use the normal to calculate the direction of a reflected ray.

In fact, the design of different materials for objects is to achieve the effect of their surface contact with light. For instance, When the light encounters glass material or mirror surfaces, it will be refracted. By contrast, when it hit a Lambertian surface, a diffuse effect can be observed. The effect of light and metal surface contact also different.

This software has implement reflection, refraction diffusion Three light effects common in the natural world but it does not mean the surface of each material has only one light effect. When the system treats mirror surfaces, this kind of material is reflective and refractive at the same time. The result depends on the direction of the ray and both the normal of the object and the index of refraction. To calculate the color at that hit point, the algorithm is divided into 3 steps: reflection color computation, refraction color computation, and Fresnel equation application. On the first stage, On the second stage, the refraction index is required for result calculation. At last,

Fresnel equation is applied to compute how much percentage of light will be reflected and how much percentage of light will be refracted. Peter Shirley (2016 pp44-45) recommended using fuzziness and randomization to simulate the Fresnel equation is also very convenient.

In addition, light diffusing on metal surfaces function can be illustrated by a picture:

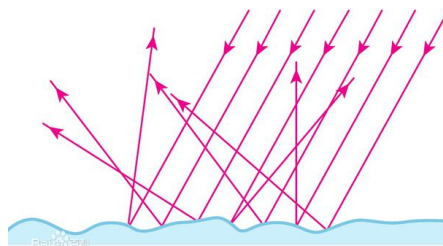


Figure 7: Diffuse effect from Baidu

A ray hitting diffuse objects not only take on the color of its surroundings but also modulate their own intrinsic color. In the meantime, abortion may happen rather than refraction. For these two factors, a smart method is to pick a random point from the surface and emit a ray from the hit point to randomized point and the algorithm only focus on the random point only on the surface. More importantly, how to store the intersection information when there are many objects in a scene? The answer is to continue store each intersection and the updated information in the previous array until this ray does not hit an object anymore.

## 4.7 Texture

Interestingly, texture characteristics can be interpreted as a material that does not have special effects. When you browse the output image, you may notice that all objects are above a large and wild floor. This special floor is also a very large sphere and that is the reason why the texture is also relevant to the material class. `solid_texture` is a function that makes the colors on a surface procedural. The implementation method of `solid_texture` is to add a judgment before rendering the floor which can generate some solid texture like a checkerboard. To sum up, each material can mix different effects than just one and the light may touch other objects after touching the first one.

In order to get cool looking solid textures, the rendering algorithm uses an open source noise generation library implemented by Jordan Peck because Joardans open source noise library has a large collection of a different noise algorithm which has been optimized for speed without sacrificing.

## 4.8 Source of light

In fact, the setting of the light source in a virtual scene can also be seen as similar to the realization of the implantation of a particular object class like the texture. The unique feature of a source of light is a ray belong to a source of light will not be reflected or refracted when it encounters with an object but stay at the intersected point and show its original color. To be more specific, the reason why the effect of a light source in the scene presented in the image generated by the software and the effect of this light source projected on the object can be seen is light emitted from that light source will acquire a new feature. This feature, technically, will stop refraction and reflection

happen when its owner hit an object and therefore the color of the light is based on the light source that is pre-set. Theoretically, the principle of light source implementation is from source to object but in practice, the direction is opposite from object to the light source. A ray whose final computed direction that point to a source of light will be recorded and the reconstructed to make it a light source ray and its color will be changed to the color of that source.

## 4.9 Antialiasing

Obviously, when there are several objects and source of light created in the scene, each given pixel may contain several samples. Also, normally, a picture taken by a real camera may not have jaggies along edges as the pixels on edge are blend of the foreground and background color. This effect can be implemented by designing a random function to average to colors of the samples in a pixel. So, the color transition on edge between foreground and background color will be softer.

## 5 Software testing

In our project, we choose a very useful test tool called Google C++ test (gtest). As a very widely used test tool for C++, we believe it would be better to choose Gtest as our tool for these reasons:

- Tests should be independent and repeatable. It's a pain to debug a test that succeeds or fails as a result of other tests. In this case, the architecture of Gtest can work well for Gtest isolate the tests by running them separately, which means once a test failed, you can quickly be debugging and checking.
- Tests should be well organized and reflect the structure of the tested code. In Gtest, the testers can group some related tests into a test case group, which can be easily managed and modify. By doing this, the test cases in one group can share data and subroutines, which can be an extremely convenient structure to start a new code base based on an existing project.
- Tests should be portable and reusable. The open-source community has a lot of code that is platform-neutral, its tests should also be platform-neutral. Google C++ Testing Framework works on different OSes, with different compilers (gcc, MSVC, and others), with or without exceptions, so Google C++ Testing Framework tests can easily work with a variety of configurations. (Note that the current release only contains build scripts for Linux - we are actively working on scripts for other platforms (Gennadiyev, 2017)).
- In most test tools, when we meet a failure, the test would stop and provide as much information about the bug as possible. But Gtest would not stop at the first failure, it only stops the current test and continues with the other tests. With this function, we can set up tests that report non-fatal failures after which the current test continues, which let the testers detect not only one bug in on compile circle.
- Gtest have a large and rich test library for C++ test and after many years supplements and modify, this test tool can handle most C++ test mission. In our project, especially back-end, we use a lot of C++ code to establish out the algorithm which makes Gtest a very suitable test tool for our project.

- Gtest can also meet the speed requirement in the test period. Gtest does not contain those complex algorithm code, but allow reuse shared resources across tests to reduce test time.

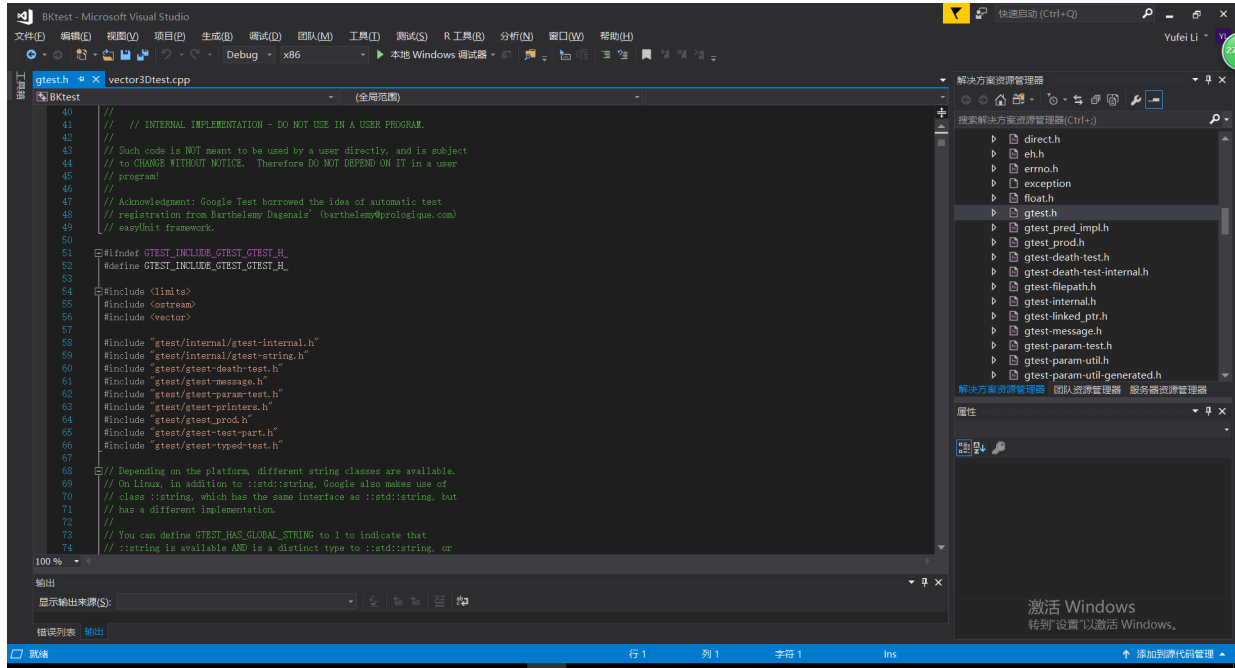


Figure 8: Google C++ Test Code Library

## 5.1 White box testing

In our project, the white-box test mostly focuses on the back-end code. The white box testing exercises every visible path of the source code to minimize errors and create an error-free environment. The whole point of white-box testing is the ability to know which line of the code is being executed and being able to identify what the correct output should be (Williams, 2006). And we also use test management software to manage our test cases and test progress. We use test log which is a widely used test tool to manage our white box testing. This software allows us to build a test project and manage a database of all the test case and information about the related sources.



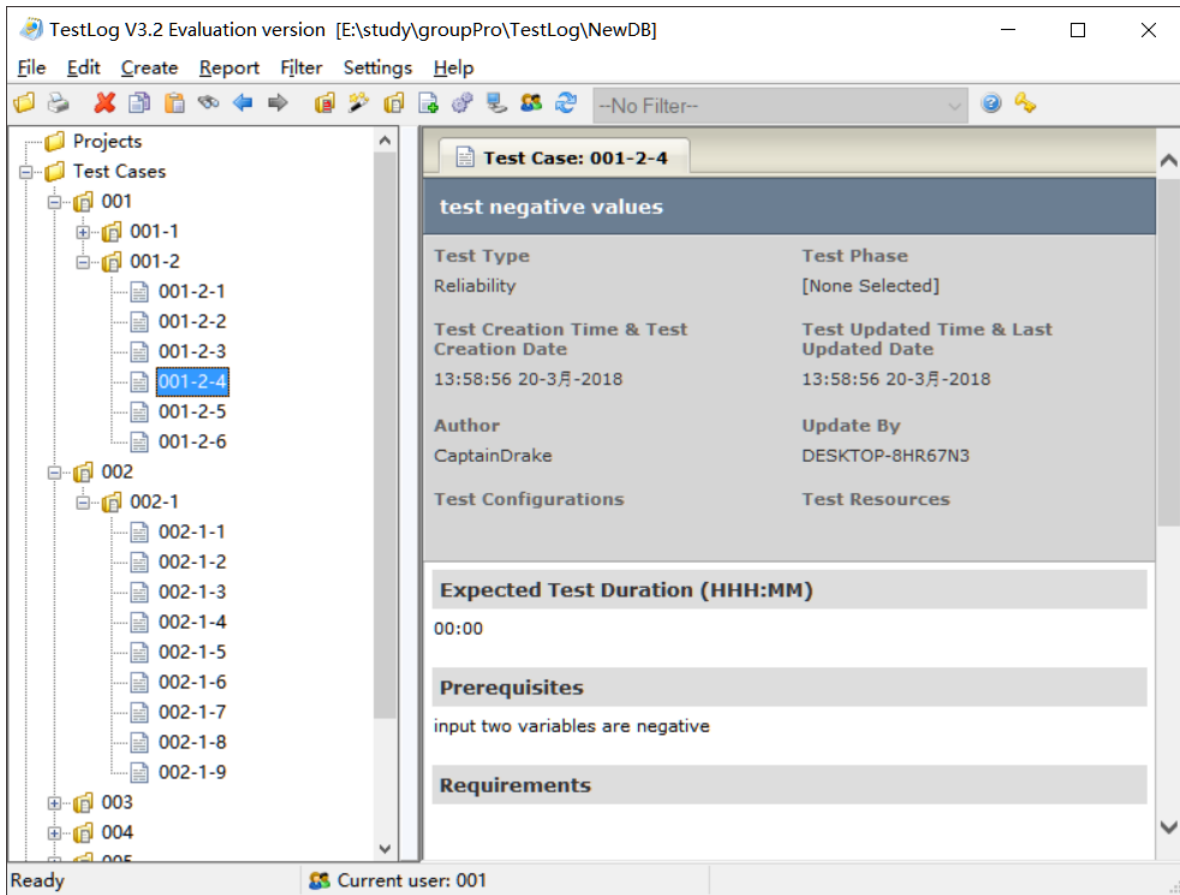


Figure 9: Test case in management software(TestLog)

### 5.1.1 Unit test

In the test period, we analyze the logic and judgment structure, then we build a lot of test cases to cover all the conditions. For example, in class `vector3D`, we have a method named `Length` which is used to calculate the length of the vector and determine whether it is valuable.

Type	Test Suite	ID	Title	Expected	Type	Phase	Created	Updated	Author	Update	Configuration	Resources	Prerequisites	Description	Results	Notes	Field	Link
suite		1	Vector3D											the basic				
case	001\001-1	001-1-1	test normal	0:00	Reliability	None	Sel	2018/3/19	14:23:25	2018/3/19	15:20:31	CaptainDr	DESKTOP-SH	A basic function (4, return the				
case	001\001-1	001-1-2	test big	0:00	Reliability	None	Sel	2018/3/19	15:18:54	2018/3/19	15:42:34	CaptainDr	DESKTOP-SH	input big input (30, expect val				
case	001\001-1	001-1-3	test zero	0:00	Reliability	None	Sel	2018/3/19	15:26:44	2018/3/19	15:42:44	CaptainDr	DESKTOP-SH	input all expect val				
case	001\001-1	001-1-4	test negat	0:00	Reliability	None	Sel	2018/3/19	15:27:52	2018/3/19	15:42:51	CaptainDr	DESKTOP-SH	test all input len expect val				
case	001\001-1	001-1-5	test 2 neg	0:00	Reliability	None	Sel	2018/3/19	15:29:15	2018/3/19	15:42:59	CaptainDr	DESKTOP-SH	input 2 neg input len expect val				
case	001\001-1	001-1-6	test one	0:00	Reliability	None	Sel	2018/3/19	15:45:01	2018/3/19	15:45:01	CaptainDr	DESKTOP-SH	input the input len expect val				
suite		1	Unitize-te											test the l				
case	001\001-2	001-2-1	test normal	0:00	Reliability	None	Sel	2018/3/20	13:53:29	2018/3/20	13:53:29	CaptainDr	DESKTOP-SH	test normal input val the final				
case	001\001-2	001-2-2	test Big	0:00	Reliability	None	Sel	2018/3/20	13:55:15	2018/3/20	13:55:15	CaptainDr	DESKTOP-SH	test some input val the final				
case	001\001-2	001-2-3	test negat	0:00	Reliability	None	Sel	2018/3/20	13:56:41	2018/3/20	13:56:41	CaptainDr	DESKTOP-SH	input all input val the final				
case	001\001-2	001-2-4	test negat	0:00	Reliability	None	Sel	2018/3/20	13:58:56	2018/3/20	13:58:56	CaptainDr	DESKTOP-SH	input two input val the final				
case	001\001-2	001-2-5	test negat	0:00	None	Sel	None	Sel	2018/3/20	14:00:27	2018/3/20	14:00:27	DESKTOP-SH	input one input val the final				
case	001\001-2	001-2-6	test empty	0:00	None	Sel	None	Sel	2018/3/20	14:01:21	2018/3/20	14:01:21	DESKTOP-SH	test value input x = the final				
suite		2	RayTest											Test class				
case	002\002-1	002-1-1	variable	0:00	Reliability	None	Sel	2018/3/20	14:55:49	2018/3/20	14:55:49	CaptainDr	DESKTOP-SH	test the input K = generate				
case	002\002-1	002-1-2	test K ov	0:00	Reliability	None	Sel	2018/3/20	15:01:17	2018/3/20	15:01:17	CaptainDr	DESKTOP-SH	test second input K = generate				
case	002\002-1	002-1-3	variable	0:00	Reliability	None	Sel	2018/3/20	15:04:14	2018/3/20	15:04:14	CaptainDr	DESKTOP-SH	test third input K = generate				
case	002\002-1	002-1-4	variable	0:00	None	Sel	None	Sel	2018/3/20	15:06:04	2018/3/20	15:06:03	DESKTOP-SH	test first input k = generate				
case	002\002-1	002-1-5	variable	0:00	Reliability	None	Sel	2018/3/20	15:07:04	2018/3/20	15:07:04	CaptainDr	DESKTOP-SH	test second input K = generate				
case	002\002-1	002-1-6	variable	0:00	Reliability	None	Sel	2018/3/20	15:08:14	2018/3/20	15:08:14	CaptainDr	DESKTOP-SH	test second input K = generate				
case	002\002-1	002-1-7	variable	0:00	Reliability	None	Sel	2018/3/20	15:10:16	2018/3/20	15:10:16	CaptainDr	DESKTOP-SH	test first input K = generate				
case	002\002-1	002-1-8	variable	0:00	Reliability	None	Sel	2018/3/20	15:12:21	2018/3/20	15:12:21	CaptainDr	DESKTOP-SH	test second input K = generate				
case	002\002-1	002-1-9	variable	0:00	Reliability	None	Sel	2018/3/20	15:13:17	2018/3/20	15:13:17	CaptainDr	DESKTOP-SH	test third input K = generate				
suite		3	CameraTest											Test class				
case	3	003-1	variable	0:00	Reliability	None	Sel	2018/3/20	15:41:37	2018/3/20	15:41:37	CaptainDr	DESKTOP-SH	test all create a create ob				
case	3	003-2	variable	0:00	Reliability	None	Sel	2018/3/20	15:44:29	2018/3/20	15:44:29	CaptainDr	DESKTOP-SH	test all create a create ob				
case	3	003-3	variable	0:00	Reliability	None	Sel	2018/3/20	15:45:42	2018/3/20	15:45:42	CaptainDr	DESKTOP-SH	test all create ne create suc				

Figure 10: Test Cases list

Just like above, the Unit test contains all the class of the back-end unit. We test all the normal value of the algorithm, and the results showed there is no obvious bug in the back-end code both in logic and conditions. Here is an example result:

```

C:\WINDOWS\system32\cmd.exe
Running 1 test from 1 test case.
Global test environment set-up.
1 test from LengthTest
LengthTest.vector3Dtest
LengthTest.vector3Dtest (0 ms)
1 test from LengthTest (1 ms total)

Global test environment tear-down
1 test from 1 test case ran. (2 ms total)
PASSED
1 test.
请按任意键继续. . .

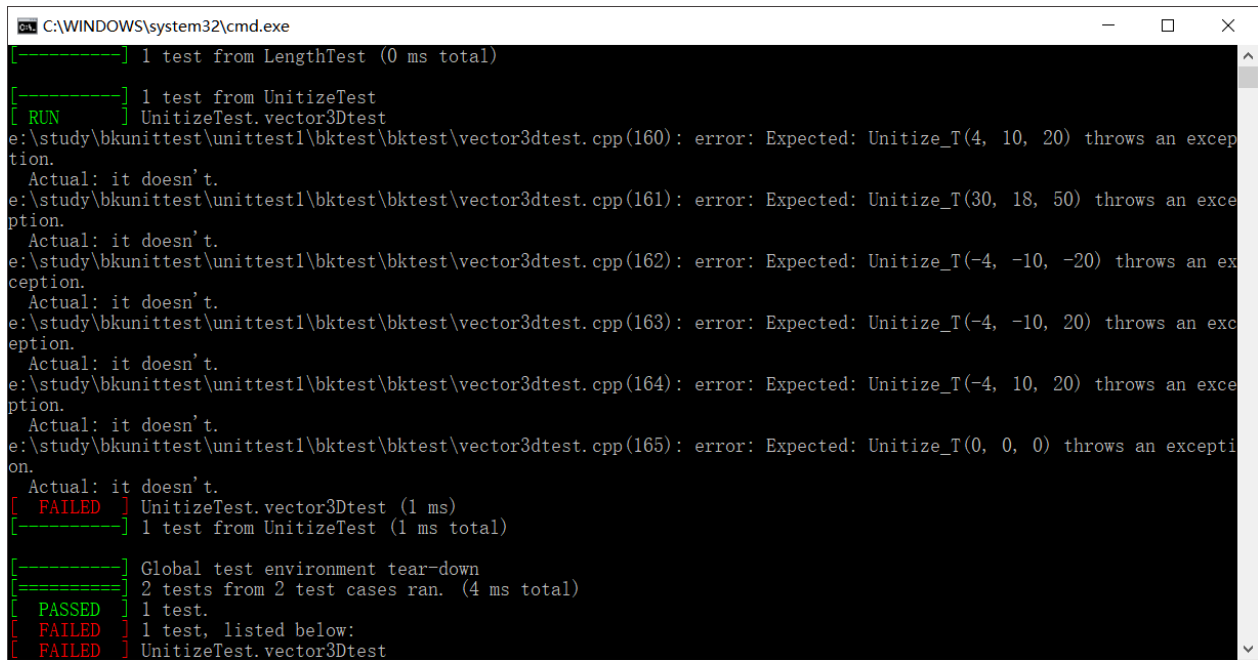
```

Figure 11: Unit Test example

### 5.1.2 Death test

In many applications, there are assertions that can cause application failure if a condition is not met. These sanity checks, which ensure that the program is in a known good state, are there to fail at the earliest possible time after some program state is corrupted. If the assertion checks the wrong condition, then the program may proceed in an erroneous state, which could lead to memory corruption, security holes, or worse. Hence it is vitally important to test that such assertion statements work as expected (Safyan, 2011).

In our project, we test some over-range buffers and some illegal value and operating. Such as we input a very long number to an int value or calculate and store the float value into an int value. So we got some result like this:



```
C:\WINDOWS\system32\cmd.exe
[-----] 1 test from LengthTest (0 ms total)

[-----] 1 test from UnitizeTest
[ RUN      ] UnitizeTest.vector3Dtest
e:\study\bkunittest\unittest1\bktest\bktest\vector3dtest.cpp(160): error: Expected: Unitize_T(4, 10, 20) throws an exception.
Actual: it doesn't.
e:\study\bkunittest\unittest1\bktest\bktest\vector3dtest.cpp(161): error: Expected: Unitize_T(30, 18, 50) throws an exception.
Actual: it doesn't.
e:\study\bkunittest\unittest1\bktest\bktest\vector3dtest.cpp(162): error: Expected: Unitize_T(-4, -10, -20) throws an exception.
Actual: it doesn't.
e:\study\bkunittest\unittest1\bktest\bktest\vector3dtest.cpp(163): error: Expected: Unitize_T(-4, -10, 20) throws an exception.
Actual: it doesn't.
e:\study\bkunittest\unittest1\bktest\bktest\vector3dtest.cpp(164): error: Expected: Unitize_T(-4, 10, 20) throws an exception.
Actual: it doesn't.
e:\study\bkunittest\unittest1\bktest\bktest\vector3dtest.cpp(165): error: Expected: Unitize_T(0, 0, 0) throws an exception.
Actual: it doesn't.
[ FAILED ] UnitizeTest.vector3Dtest (1 ms)
[-----] 1 test from UnitizeTest (1 ms total)

[-----] Global test environment tear-down
[=====] 2 tests from 2 test cases ran. (4 ms total)
[ PASSED ] 1 test.
[ FAILED ] 1 test, listed below:
[ FAILED ] UnitizeTest.vector3Dtest
```

Figure 12: Death Test example

In our project, vectors cannot set to zero and camera cannot set a null value.

## 5.2 Black-box testing

Black-box testing is a testing strategy that ignores the internal mechanism of a system or component and focuses solely on outputs generated in response to selected inputs and execution conditions (Gao et., 2003).

### 5.2.1 Request

Back-end:

- Needs to take an input file describing a scene.

- Renders the completed scene as a PNG, at a user-specified size (e.g. 500x500 pixels), to a user-specified file name.
- At a minimum: the renderer needs to be able to work with 3 dimensional coordinates; basic shapes (spheres, cubes); colours; translucency.
- Huge scope for extras: multiple light sources, textures, different shapes etc.
- The ray tracing algorithm must meet the valuable speed request.

Front-end:

- The GUI editor needs to allow users to quickly sketch out a scene, and save it to a file that the back-end can then take as input.
- The GUI must not do ray tracing by default.
- The GUI needs to provide a way of creating and editing every shape and setting the back-end can deal with.

### 5.2.2 Test method

We use Use case testing method which is a kind of system testing method. By doing this, we build a test case of input values and expect results and test all of them.

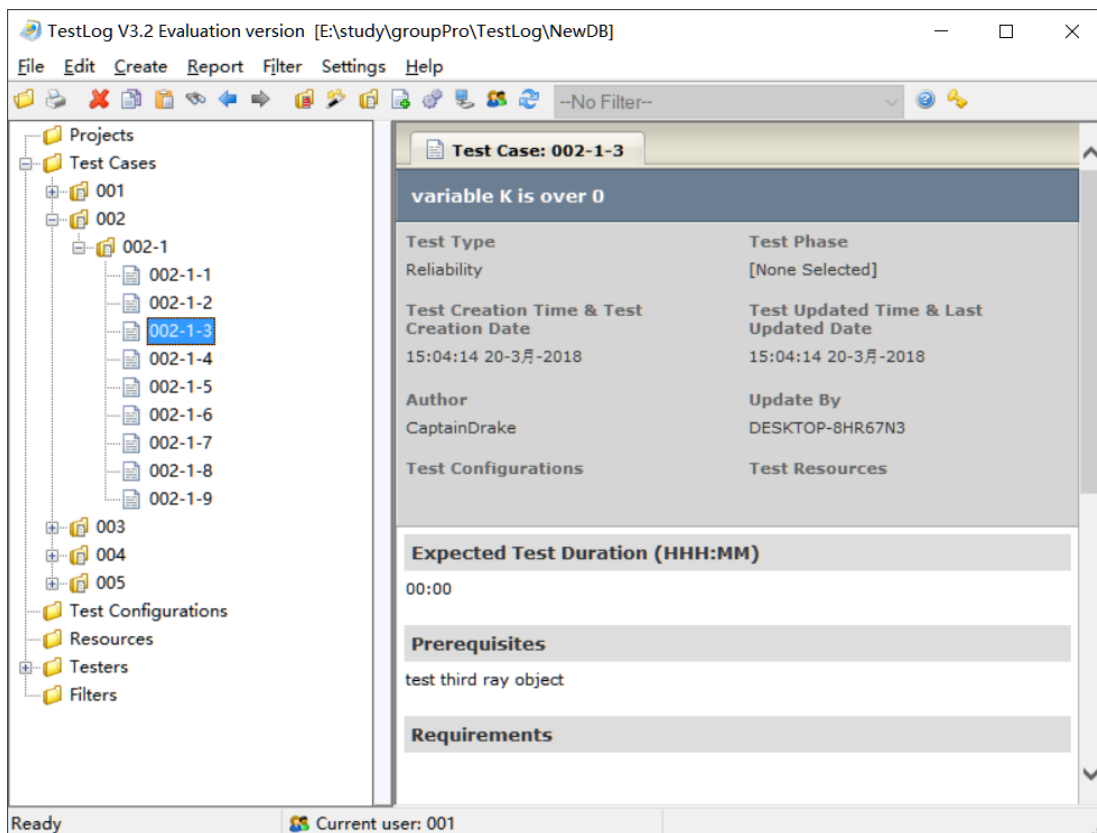


Figure 13: Black-box test case

### 5.2.3 Example input and result

Here are an example test:

- Initial:

```
{ "name": "myimage", "image": { "x": 500, "y": 500, "definition": 1080 }, "camera": { "from": { "x": 0, "y": 2, "z": 13 }, "at": { "x": 0, "y": 0, "z": 0 }, "fieldOfView": 60 }, "objects": [ { "type": "sphere", "radius": 1, "center": { "x": 0, "y": 1, "z": 0 }, "material": { "type": "marble" } }, { "type": "sphere", "radius": 1, "center": { "x": -4, "y": 1, "z": 0 }, "material": { "type": "glass" } }, { "type": "sphere", "radius": 1, "center": { "x": 4, "y": 1, "z": 0 }, "material": { "type": "metal", "color": { "x": 0.7, "y": 0.6, "z": 0.5 } } }, { "type": "rectangle", "position1": { "x": 1.3, "y": 0, "z": 0.5 }, "position2": { "x": 2.5, "y": 1.6, "z": 2.3 }, "material": { "type": "checker", "color1": { "x": 0.2, "y": 0.3, "z": 0.1 }, "color2": { "x": 0.9, "y": 0.9, "z": 0.9 } } }, { "type": "light", "radius": 2, "center": { "x": 0, "y": 7, "z": 0 } } ] }
```

Figure 14: Black-box test example initial data

- Result:

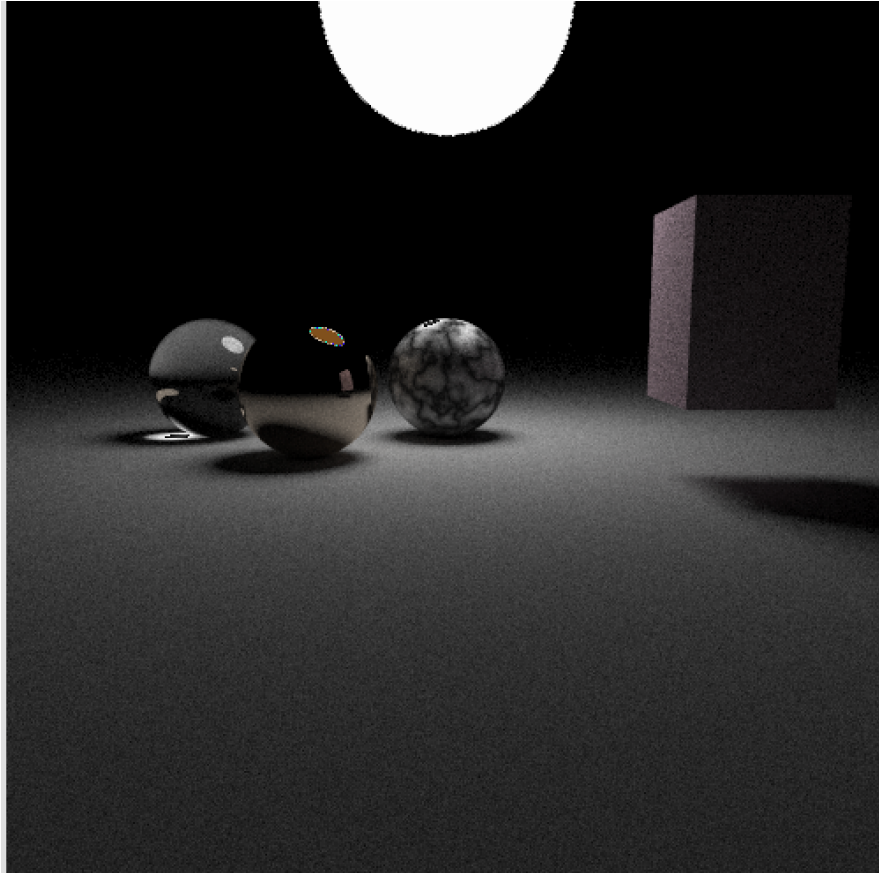


Figure 15: Black-box test example result

## 6 Responsibility distribution

Table 2: Work allocation

Part	Name	Responsibility
Back-end group	Jiawei Luo	Team Leader; Back-end algorithm implementation; Manage the project
	Qiaowei Ni	Back-end algorithm design and implementation; Back-end server implementation
	Yufei Li	Back-end algorithm design; Test management
Front-GUI group	Jinyu Hu	Object module; requirement and testing; manage the document
	Mengmeng Liu	Main page design; position module
	Shichao Fang	Preview module; Produce module

Table 3: Peer assessment

Peer assessment	
Name	Points
Jiawei Luo	17
Qiaowei Ni	19
Yufei Li	16
Shichao Fang	16
Mengmeng Liu	16
Jinyu Hu	16

## References

- [1] Appel, Arthur. "Some techniques for shading machine renderings of solids." Proceedings of the April 30–May 2, 1968, spring joint computer conference. ACM, 1968.
- [2] Auburns(2016).FastNoise.[online] GitHub. Available at: <https://github.com/Auburns/FastNoise> [Accessed 22 Mar. 2018].
- [3] Cheng Z D, He Y L, Cui F Q, et al. Comparative and sensitive analysis for parabolic trough solar collectors with a detailed Monte Carlo ray-tracing optical model[J]. Applied Energy, 2014, 115: 559-572.
- [4] Conn, Richard W. "Digitized photographs for illustrated computer output." Proceedings of the April 18-20, 1967, spring joint computer conference. ACM, 1967.
- [5] Gao, Jerry, H-SJ Tsao, and Ye Wu. Testing and quality assurance for component-based software. Artech House, 2003.
- [6] Gennadiycivil (2017). google/googletest. [online] GitHub. Available at: <https://github.com/google/googletest> [Accessed 23 Mar. 2018].
- [7] Glassner A S. Spacetime ray tracing for animation[J]. IEEE Computer Graphics and Applications, 1988, 8(2): 60-70.
- [8] Kindler, Eugene, and Ivan Krivy. "Object-oriented simulation of systems with sophisticated control." International Journal of General Systems 40.3 (2011): 313-343.
- [9] Nohmann (2017). json. [online] GitHub. Available at: <https://github.com/nlohmann/json> [Accessed 23 Mar. 2018].
- [10] Pharr, Matt, Wenzel Jakob, and Greg Humphreys. Physically based rendering: From theory to implementation. Morgan Kaufmann, 2016:23-24.
- [11] Safyan, M. (2011). What are Google Test, Death Tests. [online] Stackoverflow. Available at: <https://stackoverflow.com/questions/3698718/what-are-google-test-death-tests> [Accessed 23 Mar. 2018].
- [12] Schmittler J, Wald I, Slusallek P. SaarCOR: a hardware architecture for ray tracing[C]//Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware. Eurographics Association, 2002: 27-36.
- [13] Scratchapixel. (2017). Global Illumination and Path Tracing. [online] Available at: <https://www.scratchapixel.com/lessons/3d-basic-rendering/global-illumination-path-tracing> [Accessed 23 Mar. 2018].
- [14] Suffern K. Ray Tracing from the Ground up[M]. CRC Press, 2016.
- [15] Tomdalling.com. (2014). Explaining Homogeneous Coordinates and Projective Geometry. [online] Available at: <https://www.tomdalling.com/blog/modern-opengl/explaining-homogenous-coordinates-and-projective-geometry/> [Accessed 24 Mar. 2018].

- [16] Whitted, Turner. "An improved illumination model for shaded display." ACM Siggraph 2005 Courses. ACM, 2005.
- [17] Williams, Laurie. "White-Box Testing." PDF): 6061 69 (2006).
- [18] Dirksen, Jos. Learning Three.js: the JavaScript 3D library for WebGL. Packt Publishing Ltd, 2013.
- [19] Hinckley, Ken, et al. "Usability analysis of 3D rotation techniques." Proceedings of the 10th annual ACM symposium on User interface software and technology. ACM, 1997.
- [20] Crockford, Douglas. "The application/json media type for javascript object notation (json)." (2006).
- [21] Tilkov, Stefan, and Steve Vinoski. "Node.js: Using JavaScript to build high-performance network programs." IEEE Internet Computing 14.6 (2010): 80-83.
- [22] Di Benedetto, Marco, et al. "SpiderGL: a JavaScript 3D graphics library for next-generation WWW." Proceedings of the 15th International Conference on Web 3D Technology. ACM, 2010.
- [23] Marchant, Benoit, Pierre Frisch, and Dimitri Dupuis-Latour. "Web-based animation." U.S. Patent No. 8,643,653. 4 Feb. 2014.