Interactive Refractive Rendering for Gemstones

LEE Hoi Chi Angie

A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of

Master of Philosophy

in

Automation and Computer-Aided Engineering

Jan, 2004


香港中文大學
圖書館系統
12 APR 2005
UNIVERSITY
LIBRARY SYSTEM
THE CHINESE UNIVERSITY OF HONG KONG

Abstract

Customization plays an important role in today's jewelry design industry. There is a demand on the interactive gemstone rendering which is essential for illustrating design layout to the designer and customer interactively. Although traditional rendering techniques can be used for generating photorealistic image, they require long processing time for image generation so that interactivity cannot be attained.

The technique proposed in this thesis is to render gemstone interactively using refractive rendering technique. As realism always conflicts with interactivity in the context of rendering, the refractive rendering algorithm need to strike a balance between them. In the proposed technique, the refractive rendering algorithm is performed in two stages, namely the pre-computation stage and the shading stage. In the pre-computation stage, ray-traced information, such as directions and positions of rays, are generated by ray tracing and stored in a database. During the shading stage, the corresponding data are retrieved from the database for the run-time shading calculation of gemstone.

In the system, photorealistic image of gemstone with various cutting, properties, lighting and background can be rendered interactively. By combining the refractive rendering technique with conventional Gouraud and Phong shading technique, jewelry designs can be rendered interactively

摘要

在現今珠寶設計工業中，用戶化扮演一個很重要的角色。爲了讓設計師及顧客能夠在互動的環境下觀看不同的設計，一個互動的珠寶虛擬圖像顯示系統是必須的。傳統的虛擬圖像顯示技術雖然能夠製造出真確的圖像，但由於圖像計算需時，因此不適合在互動環境下應用。

本篇論文提出了有關寶石顯示的折射性圖像顯示技術。因圖像的真實性往往與互動性有所抵觸，所以我們需要在兩者中取得平衡。折射性圖像顯示技術的運算模型能分爲兩個部份：前處理階段及描影階段。在前處理過程中，光線跟踪方法所獲得的資料，如光線的方向和位置將被儲存於資料庫內；在描影過程中，適合的資料將從資料庫取出，用作即時的寶石描影計算。

本篇提出的系統能夠提供具互動性的寶石圖像顯示，根據不同的寶石切割形狀、燈光和背景，製造具高度真確性的影像。通過綜合本文提出的折射性圖像顯示技術及傳統的圖像顯示技術，便能達到互動珠寶設計顯示的目標。

Acknowledgements

I would like to express my most sincere thanks to my supervisor, Prof. Hui Kin-Chuen, for his help during the past two years. His guidance and encouragement were vital elements in the completion of the research. Through the comprehensive discussion with him, my knowledge has been enhanced a lot.

I would like to thank Prof. Du Ruxu and Prof. Chung Chi-Kit for serving as my graduate committee members as well as other faulty members for their patient instruction throughout my academic study at The Chinese University of Hong Kong.

Moreover, I would like to express my thanks to the colleagues in Computer Aided Design Laboratory, Brian Wong, Y. B. Wu for their help and support. Especially, I would like to thank Y. H. Lai for his advice on my work and his effort on my programming work.

Finally, I am grateful to my family members for their endless support and love through out my master studies and during the time I have spent in CUHK.

Table of Content

List of Figures

List of Tables

1 Introduction

In the jewelry design process, the use of computer-aided design technique enhances the precision in creating new designs of jewelry. This requires a realistic display of the jewelry which plays an important role in jewelry design. A photorealistic rendering of the design provides an effective means for illustrating the design layout to the customer. Interactive rendering allow customized design to be performed in real-time.

Traditionally, gemstones rendering is performed with the ray tracing algorithm which is capable of producing highly realistic images. Because ray path tracing and intersection checking in the ray tracing process is a computation intensive process, the frame rate is too slow for interactive rendering.

The techniques proposed in this thesis focus on the rendering of reflective transparent polyhedral object with a single refractive index. The technique can be used for the interactive rendering of gemstones with high transparency and reflectance. The proposed refraction rendering technique can be applied to different gemstone cuttings with specific gemstone properties. The gemstones can be rendered interactively in a user defined environment with different lighting condition and object orientation. As realism always conflicts with interactivity, the refractive rendering algorithm provides a means for achieving a balance between interactively and image quality.

The refractive rendering algorithm can be divided into two stages: the pre-computation stage and the shading stage. In the pre-computation stage, a ray tracing process specific to gemstone rendering is performed. A database is constructed for the storage of ray-traced information. They are the ray directions and the positions of the corresponding image points. In the shading stage, the corresponding data are

retrieved from the database for the shading the gemstone by taking into consideration of the global illumination.

2 Previous Works

There are two fundamentally different paradigms to solve the problems of synthesizing images of three dimensional scenes, which are the Geometry Based Rendering (GBR) and the Image Based Rendering (IBR) [Canon97] techniques. To achieve the goal of interactive gemstones rendering, GBR such as Ray Tracing is used to provide high realism for the gemstone image, where as IBR is used to provide interactivity for refractive object rendering.

2.1 Geometry Based Rendering

Geometry Based Rendering uses geometric model data to build virtual space and objects, and then represents them using computer graphics technology. The illumination and shading of a virtual 3D world are generated by calculating the perceived color of objects in the environment. The scene consists of surfaces with various properties, such as color, reflectance, transparency, material characteristic, and the specification of the lighting condition, such as the position, direction, color and intensity of the light source.

Transparent object can be rendered by using the alpha blending technique. However, the effect of refraction is not produced in the image obtained. The following gives a review on GBR based shading for real time application.

In order to obtain a more realistic image, some other non-real time rendering techniques are developed. Several studies discussed the rendering of transparent object by means of light dispersion and volume absorption in a ray tracing model, which are essentials for the rainbow-likes color and transparency of gemstones.

2.1.1 Real-Time Shading model

Two types of illumination systems, the Local Illumination System and the Global Illumination System are usually adopted in a real-time shading model.

Local Illumination System refers to the illuminations as a result of the light emitted from the light sources to the object. There are two approaches for local illumination, the Basic Shading Model and the Interpolation Based Model. The Basic Shading Model, such as Flat shading and Lambert shading computes the shade of a model by calculating the color of the polygon stored in the polygon mesh. The concept of Interpolation Based Shading Model, such as Gouraud shading [Gouraud71] and Phong shading [Phong75] uses light intensity or normal assigned to each vertex to calculate the shading.

Most popular processors and 3D accelerators use the Interpolation Based Shading Model for real-time rendering. Popular graphics libraries, such as OpenGL uses the Phong Shading Model with alpha blending for rendering transparent object. Alpha blending is to blend the color of an object with the background according to the alpha value set for the object's material, in which refraction is not considered. Although local illumination does not give photorealistic image quality, its short computing time makes it suitable for real-time shading.

2.1.2 Ray Tracing

In a Global Illumination System, not only the light emitted from light source, but also the light reflected and transmitted through objects are considered for the illumination. Ray tracing [Arvo86] and Radiosity are examples of Global Illumination System. The principle of ray tracing is to trace light ray from the view point of the observer, through each pixel on the view plane, to the nearest intersected objects. The color of

the object at the intersection point is calculated and shown in the corresponding pixel of the view plane. Radiosity is defined as the light energy that leaves a surface per unit time and area. It actually computes the overall light propagation within a scene. The light reflected diffusely in the scene can be calculated.

In the rendering of gemstones, the highly reflective and refractive properties of the gemstone material makes the Interpolation Based Shading Model not appropriate for generating a realistic gemstone image. One of the possible methods is to use Ray tracing, while Radiosity is only suitable for rendering object with diffuse reflection. Various rendering models based on ray tracing have been developed for the visualization of transparent object.

2.1.2.1 Volume absorption

Volume absorption is a crucial factor to the appearance of transparent object. It produces continuously varying but closely related colors across the object. It depends not only on the spectral absorptivities of materials but also the path lengths over which lights travel inside the object. The objects in the two images of Figure 2.1 are identical hexagonal pyramids, but the materials are "light blue glass" and "red glass" respectively. White lights from the light source transmit through the object and become colored due to volume absorption.

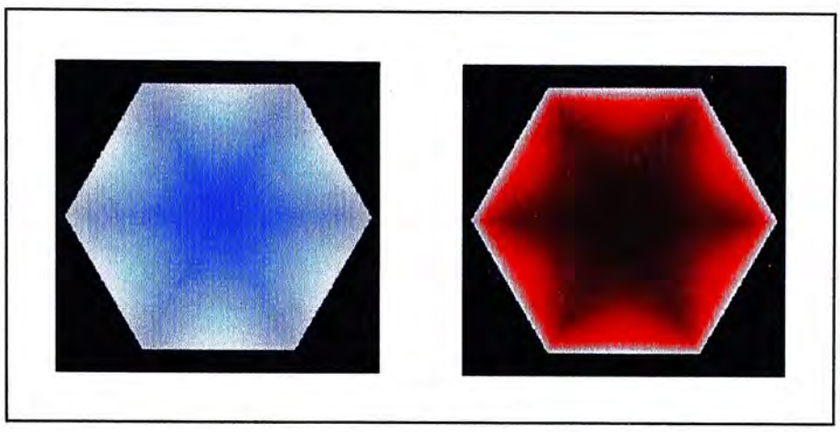

Figure 2.1 Continuous color variations generated by volume absorption [Sun99]

Sun et al. [Sun99] modeled volume absorption for homogenous transparent materials using Bouger- Lambertian law and rendered it by spectrum ray-tracer. A brilliant cut diamond shown in Figure 2.2 demonstrates the colors with piecewise variations in both intensity and saturation due to the effect of volume absorption.

Figure 2.2 An image of a brilliant cut diamond generated by volume absorption [25]

When the color of an object sensitively affected by the viewing and illumination directions, the spectral information is required for the rendering process. Conventional RGB based renderer does not consider any color change and thus fail to give a proper result. In this case, spectrum based renderer is used instead of RGB based renderer.

Spectrum based renderer takes into consideration the spectral information for modeling light-material interactions, and hence gives a more realistic image. Besides, the spectrum-based approach eliminates the errors due to the inadequacy of using the primary colors for modeling light and materials.

2.1.2.2 Light Dispersion

The prism effects and rainbows-like color variation are all familiar consequences of light dispersion phenomenon. In Figure 2.3, a narrow beam of white light passes through a prism producing a colored strip on a diffusive screen.

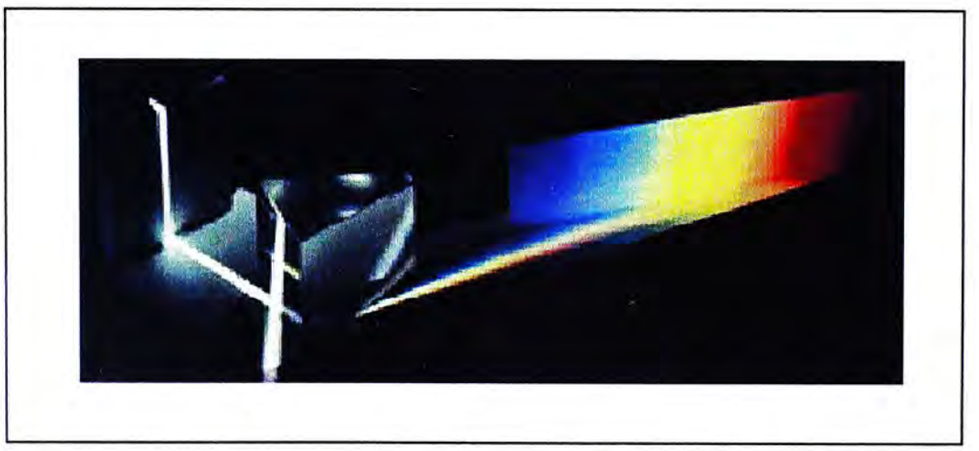

Figure 2.3 Photograph that illustrate light dispersion [Sun00b]

Yuan et al. [Yuan88] presented an approximation method which uses three simple rays adaptively to model the ray spreading caused by dispersive refraction. The method reduces the computational complexity to an order close to that of normal ray tracing.

Collins [Collins94] proposed a 2-pass ray tracing algorithm, with an adaptive light-pass followed by a standard eye-pass to render the light dispersion of crystal glass. Sun et al. [Sun00b] also proposed a composite spectral model for light dispersion. The model decomposes any spectrum into a smooth component and

spikes, and offers a convenient method for describing monochromatic lights occurring in light dispersion

Sun et al. [Sun00a] focused on generating realistic computer images of diamonds by extending a regular ray tracing techniques to incorporate the three basic physical mechanisms for rendering diamond, including fresnel reflection [Hecht87], volume absorption and light dispersion. In Figure 2.4, the white highlight triangles on the diamond are the result of fresnel reflection of the illuminating lights. The red color with various degrees of saturation is generated by volume absorption. This technique gives realistic gemstone image but is computationally intensive and is not suitable for interactive rendering.

Figure 2.4 Photographs of a red diamond [7].

2.2 Image Based Rendering

Image based rendering represents 3D space based on captured image. It does not mainly rely on the geometrical information. The rendering process is accelerated by using pre-computed image data instead of generating an image based purely on the geometric description of the objects to generate new frames. Therefore, image data in certain form must be saved in the database during the pre-computation stage.

The studies of IBR for refractive object can be divided into three aspects. They are environment mapping, light field rendering and hardware acceleration.

2.2.1 Environment mapping

Environment mapping is widely used in real time rendering of transparent object. Kay et al. [Kay79] developed linear and non-linear algorithms for intensity change and applied Snell's law on environment mapping of the refractive object. However, this technique is restricted to the rendering of object of uniform thickness.

P. Y. TS'O et al. [TS'O87] generated refracted water wave images by using texture mapping based on polygon ray tracing. Diefenbach et al. [Diefenbach94] simulates the effect of recursive reflections by the use of secondary viewpoints, and extended this method for rendering a refractive surface using a homogeneous 2-D projective image mapping. They only give the algorithms for refraction in water waves and planar surfaces.

Zongker et al. [Zongker99] introduced an "environment matting and compositing" technique which support relighting of translucent or refractive objects and changing of the arbitrary objects' background at interactive speeds. The environment matt not only describes the opacity of the foreground element at every point, but also describes how that object refracts and reflects light. The environment matt is done by capturing the light came from the backdrops in a fix viewpoint. When the foreground object is placed in a new environment, the environment compositing can then be done by simple texture mapping to form a realistic scene. However, the generated image is limited to a fixed viewpoint only.

Matusik et al. [Matusik02] proposed a rendering technique for transparent and translucent objects in which the objects can be rendered from arbitrary viewpoints.

Alpha and environment mattes of the object from multiple viewpoints are acquired for the refraction components, while surface reflectance fields using a overhead array of lights are acquired for the reflection components. However, it does not support real-time or interactive rendering as the rendering speed is about five minutes per frame on a 2GHz P4 PC with 1GB RAM.

Lensch et al. [Lensch02] presented a rendering method for translucent object, in which view point and illumination can be modified at interactive rates. The local response and global response to incoming light impinging at each surface is computed and stored in the preprocessing step. The rendering method is based on the property of highly scattering media. The light impulse response on the surface of a translucent object is factorized into a high frequency local part and a low frequency global part.

2.2.2 Light field Rendering

Levoy et al. [Levoy96] proposed an image-based rendering method for generating views from arbitrary camera positions by combining and resampling the images. It relies on interpreting the input images as 2D slices of a 4D light field function. The approach is shown in Figure 2.5. A line is parameterized by its intersections with two planes in arbitrary positions. A light field is created from a set of images. This corresponds to inserting 2D slices into the view volume of a view point. New view can be generated by extracting and resampling a slab as illustrated in Figure 2.6.

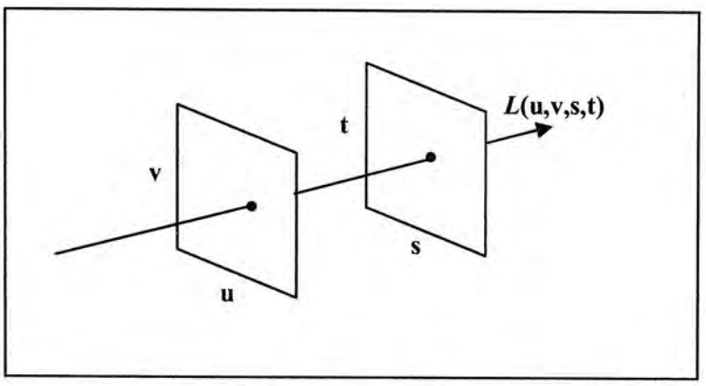

Figure 2.5 The light slab representation.

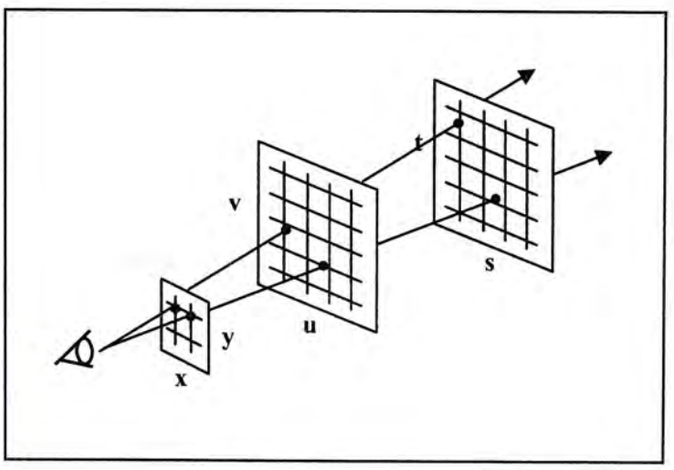

Figure 2.6 The process of resampling a light slab during display.

Gortler et al. [Gortler96] proposed a similar method called a Lumigraph. It is a subset of the complete plenoptic function [Adelson91] that describes the flow of light at all positions in all directions.

Heidrich [Heidrich99] presented a method for rendering reflections and refractions in curved objects based on light field and Lumigraph. The core of method is to separate the geometry and the illumination of object into two distinct image-based data structures. The first data structure is a 2-plane parameterized light field containing a mapping from incoming rays to outgoing rays based on the

geometry and refractive properties of the object. The illumination corresponding to this outgoing ray can either be provided in the form of an environment map or in the form of another light field. In Figure 2.7, the left image shows the color coded texture coordinates for the environment map which are extracted from the geometry light field. The color values of the pixels in an image are interpreted as texture coordinates. The right image is the result of light field rendering.

Figure 2.7 Images created by light field rendering [Heidrich99]

However, the observer is restricted to the regions of space free of occlusion. Besides, light field rendering is computational expensive and requires large memory space for storing the 4D data.

2.2.3 Hardware Acceleration

Hakura et al. [Hakura01] gave a study on producing realistic reflections and refractions based on graphics hardware. The proposed hybrid rendering, greedy ray path shading model and layered environment maps are supported by specific hardware.

Ohbuchi [Ohbuchi03] proposed the per Vertex Ray Tracing method for real-time refraction rendering and discussed the hardware implementation of the algorithm. The algorithm consists of a pre-processing phase and a rendering phase. In the pre-processing phase, the directions and ray destination points of the rays attached to each vertex in the polygons are calculated by ray tracing. In the rendering phase, the polygons in the refractive objects are rendered by using the results obtained in the pre-processing phase to compute the refracted color. However, total internal refraction of the object is not addressed. A prototype of a single-chip multimedia processor with graphics rendering hardware is developed for the implementation of his proposed refraction renderer.

In general, using IBR techniques, interactive respond can be achieved using environment mapping. However, the reality of the rendered image will be sacrificed. Photorealistic images can be generated by light field algorithm, which is computational expensive. Rendering speed can be enhanced by hardware acceleration, but specific graphic card and accelerator must be used.

In this research, an interactive gemstone rendering technique has been developed. The technique is similar to that of [Ohbuchi03] in the sense that ray information are computed and stored in the pre-computation stage. The method imitates the effect of total internal reflection in transparent object, which is an essential element in gemstone rendering.

3 Gemstones

Mostly, people characterized gemstone beauty by the two terms: fire and brilliance. Brilliance means the intensity of the internal and external reflections of white light from the crown of a polished diamond or other gemstone which are defined by the Gemological Institute of America (GIA) Diamond Dictionary [GIA03]. Fire means the portions of the light dispersed into visible rainbow colors. Both of them are affected by the hardness, refractive index, reflectivity, polish, luster, and proportions.

Based on the transparency, gemstones can be divided into three main types. They can be as transparent as water, such as Sapphire, Ruby, Topaz and Diamond; as opaque as a cliff, such as Jade, Opal and Moonstone; or anywhere in between.

As some terms mentioned later on relate to the basic optical properties and cutting of gemstones, a brief introduction of these terms are given in this chapter. The gemstone models will be discussed later in this chapter.

3.1 Basic optical properties

The basic optical property of gemstone consists of **Hue**, **Refractive index**, **Clarity** **Transparency**, and **Dispersion**.

- **Hue**

  Hue refers to the shade, tint or sensation of a color.

- **Refractive index**

Refractive index defines the bending level of the incident ray when it passes through the object. Most gemstones have only one refractive index while some gemstones may have two refractive indices.

- **Clarity**

All natural gemstones have inclusions. Clarity is simply an assessment of the degree of inclusions inside the gem crystal. Figure 3.1 shows a tourmaline with high clarity.

Figure 3.1 Intense deep green tourmaline with high clarity

- **Transparency**

Transparency describes the degree to which light passes through the stone. It depends on the amount of inclusion since inclusion blocks the passage of light. Therefore, the more the inclusion, the less the transparency. In Figure 3.2, we can see that there is much inclusion in the emerald, so that the transparency is low.

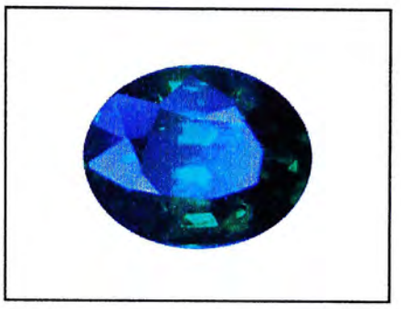

Figure 3.2 An oval emerald with low transparency

- **Dispersion**

Dispersion is the prismatic effect of the crown facet created by white light. White light is a mixed color of all spectrums with different frequencies. When white light passes through the refractive object, the spectrums will be separated apart according to its own range of frequencies, which gives the rainbow-like color of gemstones. Figure 3.3 shows the rainbow-like color variations of diamonds caused by light dispersion.

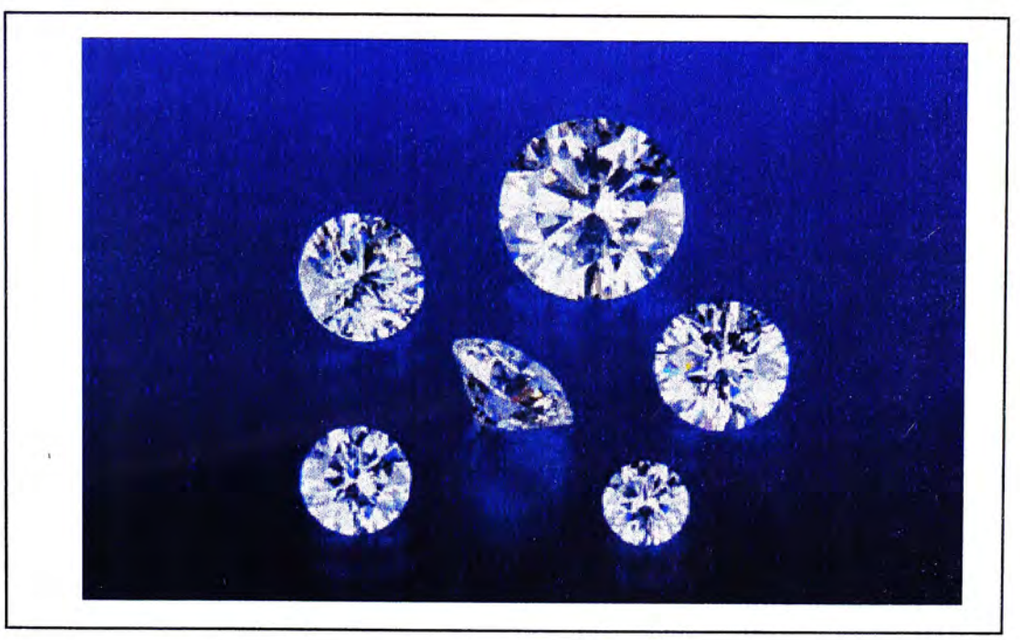

Figure 3.3 Photograph of colorless diamond [Tannous03]

When a diamond is cut to good proportions, light will reflect from one mirror-like facet to another and disperse through the top of the stone, resulting in a display of brilliance and fire.

3.2 Cutting

Cutting is an important property in all gemstones. Figure 3.4 shows the critical features for gemstone cutting. Good proportions in diamond cutting causes light to reflect from one mirror-like facet to another and disperse through the top of the stone, resulting in a display of brilliance and fire. Figure 3.5 (a) shows the ideal cutting proportion of crown and pavilion while (b) and (c) demonstrate a very deep and a very shallow stone due to poor cutting proportion of crown and pavilion respectively.

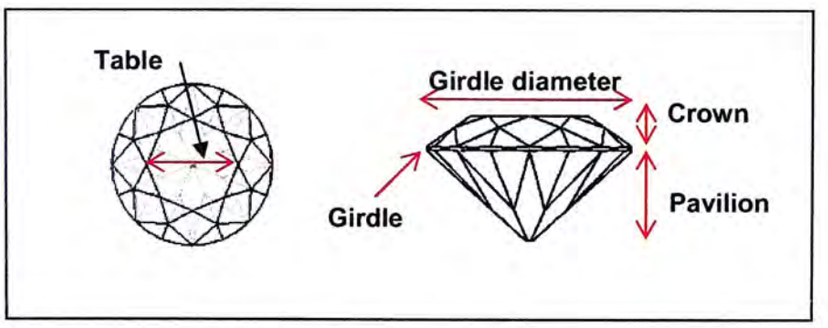

Figure 3.4 Top and front view of brilliant gemstone

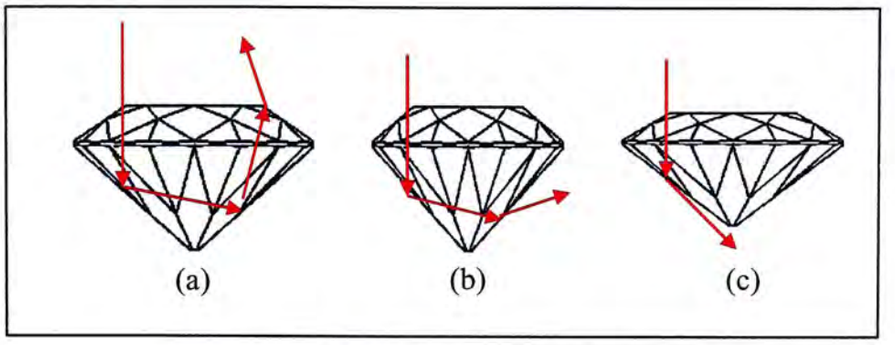

Figure 3.5 The cutting angle of gemstone

There are various types of cutting shape design, for example brilliant round, marquise, heart, oval, tear and princess, which are shown in Figure 3.6.

Figure 3.6 Various cutting of gemstone

3.3 Gemstone model in the proposed system

The proposed system provides interactive refractive rendering for polyhedral object. Specific types of gemstone with high transparency can be rendered. In the system, gemstone models are assumed to have single refractive index with ideal cutting. The dispersion effect is ignored in the system for simplicity. More details will be given in chapter 5.

4 Ray Tracing

In this chapter, the concept and algorithms of standard ray tracing will be discussed. Basically, ray tracing can be divided into two streams: Forward ray tracing and Backward ray tracing.

4.1 Forward ray tracing

Forward ray tracing traces the image from the light sources, following the light path to the viewer's viewpoint. Consider a ray from the light source to the observer's eye through the image plane (see Figure 4.1). The color of the object at the intersection point is calculated and then projected onto the corresponding pixel of the view plane. As it traces along the light path, it is capable of modeling various effects like reflection, transmission and shadow in the real world.

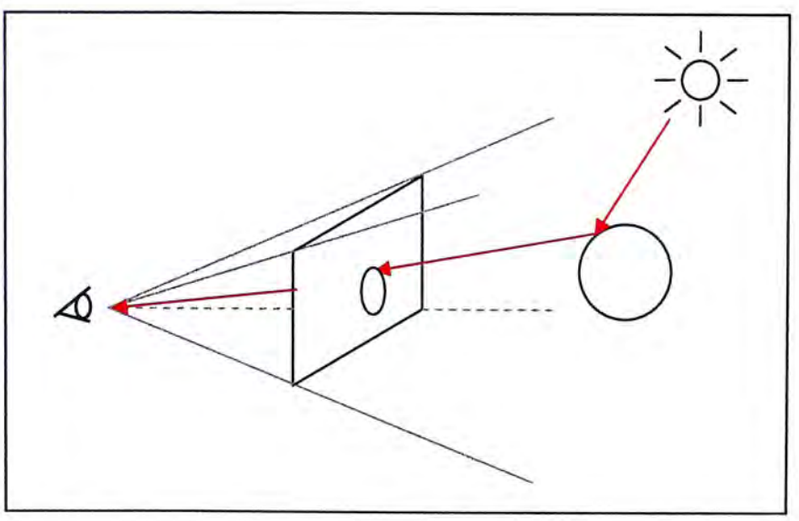

Figure 4.1 Ray tracing principle

Forward ray tracing determines the color of each object accurately. However, it is highly inefficient and expensive in computation because there may be large numbers of rays from the light source never reaching the viewer's eye, which means

they do not contribute to the final image. This makes a considerable amount of computational time being wasted.

4.2 Backward ray tracing

Backward ray tracing compute an image by tracing the rays from the viewer's viewpoint, through the view plane to the light sources. In this case, only those rays contributing to the pixel color are being computed. In this research, backward ray tracing is used. Unless specified, the term "ray tracing" mentioned here after in this thesis refers to backward ray tracing.

A ray tracing model consists of four types of ray, which are pixel ray, illumination ray, reflected ray and transmitted ray. They are shown in Figure 4.2.

- **Pixel Ray**

Pixel ray is the part of the ray that carries light from an object to the observer. In the Figure 4.2, a pixel ray is shown as black line.

- **Illumination Ray**

A ray originating from a light source is called an illumination ray. In the Figure 4.2, illumination rays are represented as dotted lines.

- **Reflected Ray**

When a ray hits a surface, a specular reflection takes place. They are shown as red lines in Figure 4.2.

- **Transmitted Rays**

A transmitted ray undergoes refraction when it passes from one media to another. In Figure 4.2, they are shown as blue lines.

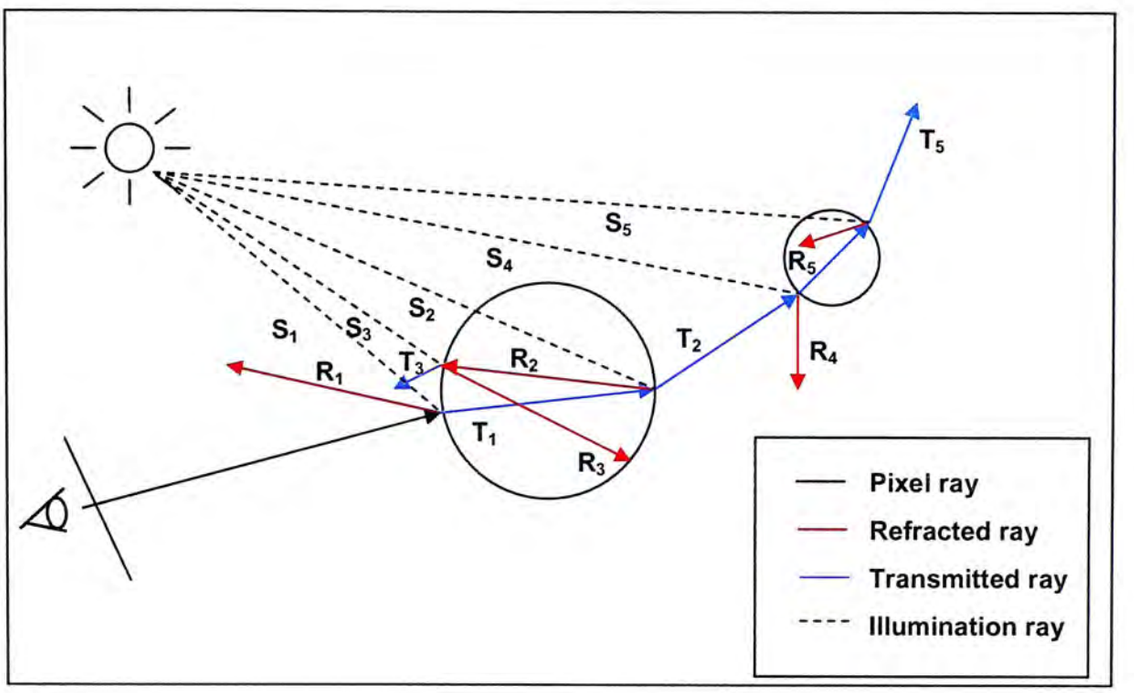

Figure 4.2 Backward ray tracing diagram

4.3 Recursive ray tracing

Recursive ray tracing is the recursive application of ray tracing to calculate the shading of objects, and is adopted to imitate the reflection and transmission of light rays in gemstone (See Figure 4.2). Starting with a single pixel ray, when the ray hits the surface of a transparent object, two new rays are produced, which are the reflected ray and the transmitted ray. Each of the two generated rays can be viewed as a new starting point of another ray tracing process. Traced rays for each pixel can be arranged in a tree-structure called "ray-tree". A ray-tree contains a root node, branches and leaves, which are shown in Figure 4.3. The root node of the tree is the

point visible to the viewer, that is, the nearest intersection of the pixel ray and the object being rendered. Each branch carries a weight ranging from 0 to 1 representing the contribution of a ray to the overall illumination of the surface. Going down the tree, the contributions become less significant. The branch without a node (open end) means that there is no intersection with objects anymore. If the ray intersects the light source, intensity of the light source will be used to determine to the illumination. Otherwise, background will be allocated to the corresponding pixel point.

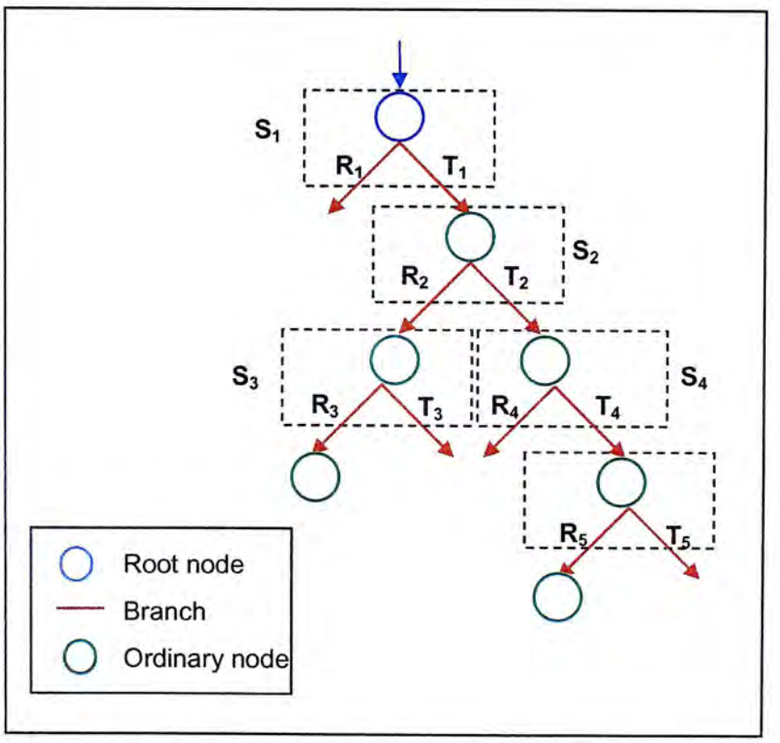

Figure 4.3 A ray-tree corresponding to rays in Figure 4.2

4.4 Ray tracing Algorithm

A ray is defined with two vectors: one describing a position through which the ray passes and one describing the direction of the ray. Equation (4.1) is the parametric form of ray, p_0 represents the point through which the ray passes, $(p_1 - p_0)$ represents the direction of the ray.

$$p = p_0 + t(p_1 - p_0) \qquad 0 \le t < \infty \qquad (4.1)$$

4.4.1 Ray/ Plane Intersection Calculation

Given a ray r_i and the equation of a plane

$$r_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \begin{pmatrix} X_0 + X_d * t \\ Y_0 + Y_d * t \\ Z_0 + Z_d * t \end{pmatrix} \qquad (4.2)$$

where $p_0 = (X_0, Y_0, Z_0)$ and $p_1 - p_0 = (X_d, Y_d, Z_d)$

$$AX + BY + CZ + D = 0 \qquad (4.3)$$

The intersection cam be computed by substituting the explicit form of the ray parametric equation into the plane equation, so that

$$A*(X_d*t + X_0) + B*(Y_d*t + Y_0) + C*(Z_d*t + Z_0) + D = 0 \qquad (4.4)$$

hence,

$$t = \frac{-(A*X_0 + B*Y_0 + C*Z_0 + D)}{A*X_d + B*Y_d + C*Z_d} \qquad (4.5)$$

By putting the value t of into the parametric ray equation, the intersection point can be calculated by Equation (4.2).

4.4.2 Shading Equation

Shading is the process of assigning color to the object point where a pixel ray intersects an object. All illumination, either direct or indirect to the object is considered in the shading process. A shading equation is constructed by combining the contributions of four different light transport mechanisms: ambient lighting,

diffuse reflection, specular reflection and specular transmission. They will be discussed later in this chapter.

The equations listed below are to be applied for monochromatic light. They can be applied for colored light by resolving the light into RGB components which are then considered separately.

4.4.2.1 Ambient Lighting

Ambient Lighting is used to compensate for inter-object indirect illumination. The intensity of ambient lighting can be calculated by Equation (4.6). I_o represents the maximum intensity of object color, k_a determines the percentage of ambient light reflected by the surface.

$$I_a = k_a I_o \qquad (4.6)$$

4.4.2.2 Diffuse Reflection

Diffusely reflected light is reflected in all directions with equal intensity. The intensity of diffuse reflection can be calculated by Equation (4.7). I_l represents the intensity of the light, k_d represents the diffused reflection constant which determines the percentage of diffuse light reflected by a surface, N represents the normal of the object's surface and L represents the direction of the light.

$$I_d = I_l k_d I_o (N \bullet L) \qquad (4.7)$$

4.4.2.3 Specular Reflection

Specular reflection is exhibited by smooth surfaces. Specular reflection produces highlight on the surface of an object as a small bright light spot. As the angle of

incidence is always equal to the angle of reflection, the reflected ray can be calculated by Equation (4.8).

$$R = 2N(L \bullet N) - L \qquad (4.8)$$

The intensity of specular reflection can be calculated by Equation (4.9), which is illustrated in Figure 4.4, where V and L represent a ray to the eye and a ray to the light source respectively, N is the surface normal, and P is the point in consideration. In Equation (4.9), h represents the polishing level of an object's material, k_s represents the specular reflection constant which determines the percentage of specular light in the scene reflected by the surface. The higher the value of k_s, the faster specular highlight falls off. The higher the value of h, the higher polish a surface has. The reflectance decreases rapidly with increasing viewing angle.

$$I_s = I_l k_s (R \bullet V)^h \qquad (4.9)$$

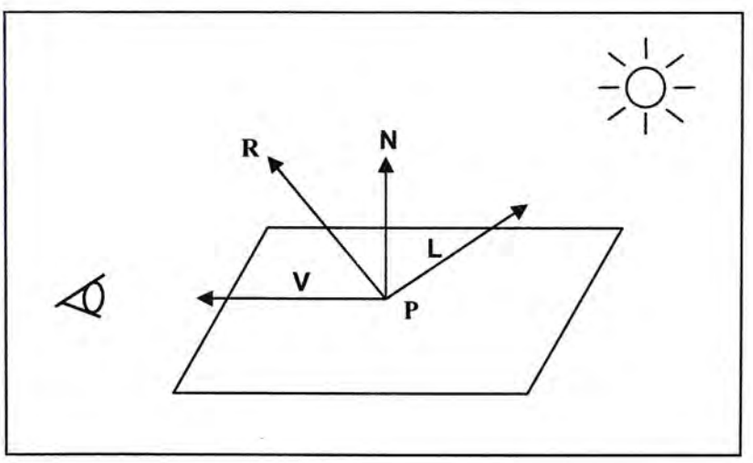

Figure 4.4 Specular Reflection

4.4.2.4 Specular Transmission

Specular transmission is exhibited by transparent object. In general, a light ray is refracted when it passes from one medium to another with different densities. Each medium has its own "index of refraction" or "IOR". IOR is a measure of the relationship between the speed of light through the media and the speed of light through vacuum space. The Equation (4.10) relating the angle of the incident ray (θ_1) and transmitted ray (θ_2) is the Snell's Law. In the equation, n_1 is the IOR of medium 1 with respect to vacuum, n_2 is the IOR of medium 2 with respect to vacuum, and n_{21} is the IOR of medium 2 with respect to medium1.

$$\frac{\sin(\theta_1)}{\sin(\theta_2)} = n_{21} = \frac{n_2}{n_1} \qquad (4.10)$$

The direction of the transmitted ray T is calculated by Equation (4.11) which is derived from Snell's Law. Again, V, L and N are used as mentioned above, and n is the IOR.

$$T = n_{12}V + (n_{12}(V \bullet N) - \sqrt{(1 + n_{12}{}^2((V \bullet N)^2 - 1))})N \qquad (4.11)$$

There is a phenomenon called "Total Internal Reflection" or "TIR". It occurs only when light passes from a dense medium to a medium of lower density. If the quantity $(1 + n_{12}{}^2((V \bullet N)^2 - 1)))$ is less than zero, TIR will occur. In this case, there will be no transmitted ray. Only reflected ray is reflected off the internal surfaces.

The calculation of the intensity for specular transmission is similar to that of specular reflection and can be expressed as Equation (4.12).

$$I_s = I_t k_{st} (T \bullet V)^h \qquad (4.12)$$

4.4.3 Fresnel equations

The Fresnel equations describe the behaviour of light when moving between media of differing refractive indices. When light moves from a medium of a given refractive index n_1 into a second medium with refractive index n_2, both reflection and transmission of the light may occur. The fraction of the incident light that is reflected from the interface is given by the reflection coefficient C_R, and the fraction of the incident light is given by refracted by the transmission coefficient C_T. The Fresnel equations may be used to calculate C_R and T_R in a given situation.

The calculations of C_R and C_T depend on polarisation of the incident ray. Figure 4.5 shows the reflection and transmission of a light wave in a plane. In the figure, k_i is the incident wave, k_r the reflected wave, k_t is the transmitted wave, θ_i is the angle of incident and θ_t is the angle of transmission.

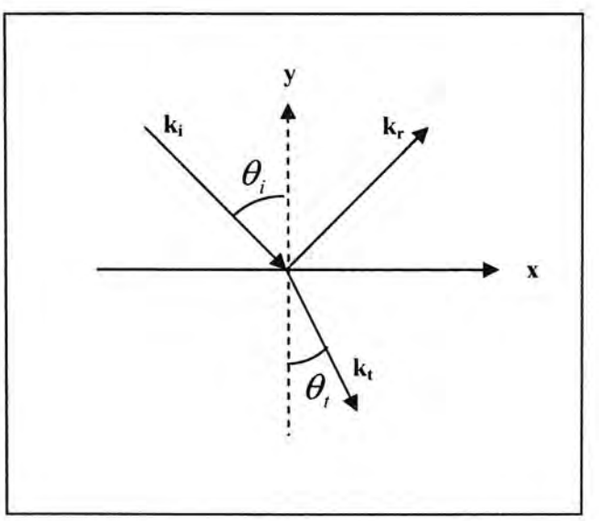

Figure 4.5 Reflection and transmission of a light wave in a plane

If the light is polarised with the electric field of the light perpendicular to the plane shown in Figure 4.5, the reflection coefficient $C_{R,\perp}$ and the transmission coefficient $C_{T,\perp}$ are given by Equation (4.13) and (4.14).

$$C_{R,\perp} = -\frac{\sin(\theta_i - \theta_t)}{\sin(\theta_i + \theta_t)} \tag{4.13}$$

$$C_{T,\perp} = \frac{2\sin\theta_t \cos\theta_t}{\sin(\theta_i + \theta_t)} \tag{4.14}$$

If the incident light is polarised in the plane shown in Figure 4.5, the reflection coefficient $C_{R,//}$ and the transmission coefficient $C_{T,//}$ are given by Equation (4.15) and (4.16).

$$C_{T,//} = \frac{2\sin\theta_t \cos\theta_t}{\sin(\theta_i + \theta_t)\cos(\theta_i - \theta_t)} \tag{4.15}$$

$$C_{R,//} = \frac{\tan(\theta_i - \theta_t)}{\tan(\theta_i + \theta_t)} \tag{4.16}$$

For the light which is unpolarized, the reflection coefficient C_R and the transmission coefficient C_T are calculated by Equation (4.17) and (4.18).

$$C_R = \frac{\left|C_{R,\perp}\right|^2 + \left|C_{R,//}\right|^2}{2} \tag{4.17}$$

$$C_R = \frac{\left|C_{T,\perp}\right|^2 + \left|C_{T,//}\right|^2}{2} \tag{4.18}$$

5 The Pre-computation stage

In the pre-computation stage, positions and directions of rays and the positions at which the rays intersect the transparent object are calculated by the ray tracer recursively. In Figure 5.1, a ray is traced from viewpoint to a transparent object. **P** is the ray /object intersection point and **D** is the outward ray direction at **P**. They are calculated by the ray tracer and stored in the database for the illumination calculation in the shading stage. In this chapter, the proposed ray tracer and the data structure for representing $\mathbf{P}_i$ and $\mathbf{D}_i$ will be discussed.

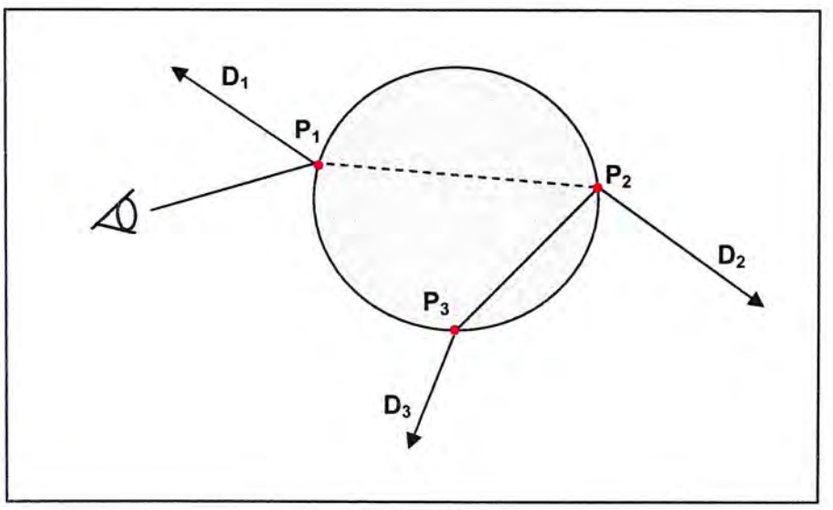

Figure 5.1 Illustration of **P** and **D**

5.1 Ray tracer

The proposed ray tracer is built upon a recursive ray tracing algorithm for gemstone rendering. The output data (**P** and **D**) of the tracer is stored and is used in the shading stage. To achieve the interactive rendering, some non-critical factors involving intensive computation are simplified. As gemstone and jewelry are usually small in size, it is difficult to notice the difference between the outputs of a complete model

and a simplified model by human perception. Four simplifications below are made in the tracing algorithm.

5.1.1 Simplifications

- **Removal of frequency-dependent term**

In the light and surface interaction, the frequency of the light factors is taken into the consideration. Dispersion of gemstone is caused by the frequency components of the spectrums of white light being bended with different amounts. The omission of frequency terms means dispersion cannot be model and other subtle lighting effects will be lost.

- **Inter-object reflections and transmissions**

Inter-object reflections and transmissions can be computed by radiosity method as discussed in chapter 2. However, the radiosity method is computation intensive and is not suitable for interactive application. Therefore they are not being considered in the tracer.

- **Fresnel equations**

In the system, T-factor and R-factor, which are the proportion of light being transmitted and reflected, are introduced to replace the Fresnel coefficient (C_R and C_t) for reflections and transmissions mentioned in chapter 4. More details can be found in chapter 6.

- **Distance**

Light traveling through space is attenuated as square of the distance. Light traveling twice as far is one-quarter as intense. Our tracer takes neither the distance between the light source and objects, nor the distance between objects that provide reflected light to another object, into consideration.

In generally, the path for a light ray passing through a gemstone with ideal cutting to the viewer's eye is a line consisting of five segments. That's, a light ray is transmitted or reflected five times before it reaches the eye of viewer (See Figure 5.2). Therefore in the proposed system, the first five segments of a ray are traced. Further segments along the ray are neglected.

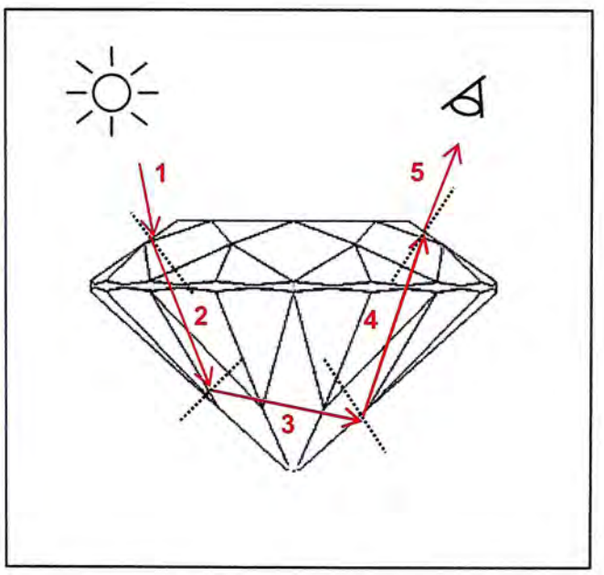

Figure 5.2 Ray traced in gemstone

5.2 Representing Position and Direction of Rays

The ray directions and the positions at which they intersect the object are computed in the ray tracing process and are stored for the illumination calculation in the shading stage. In Figure 5.3, lines in different colors represent the rays corresponding to different pixels in the image plane.

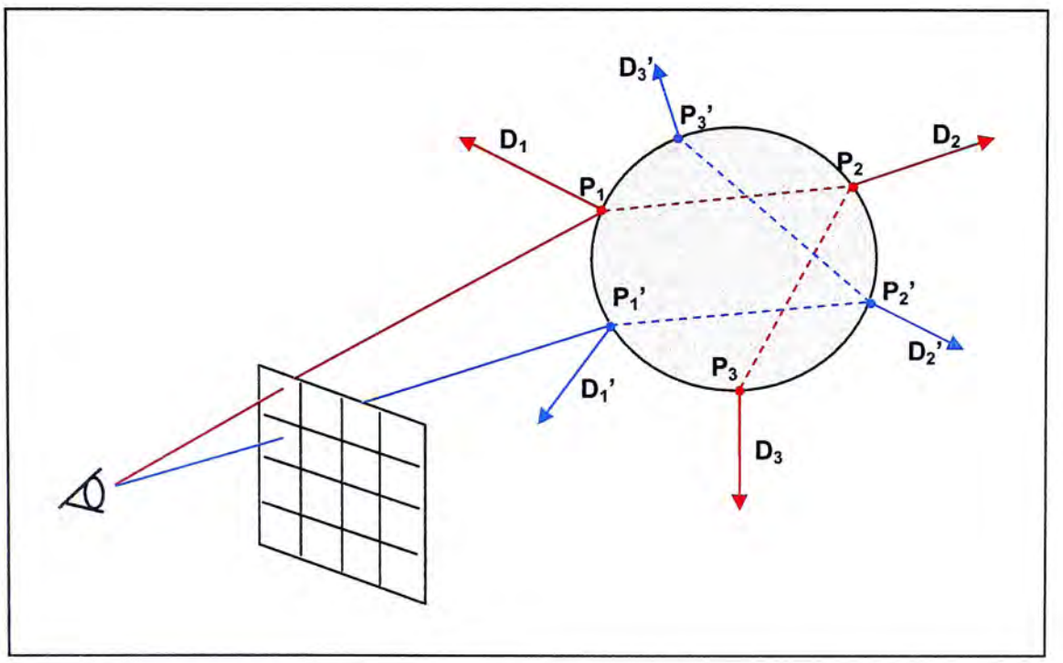

Figure 5.3 Illustration of ray tracing process from viewer through different pixels on image plane to the object

A ray-segment can be represented as

$$R_{i,j} = \{(P_k, D_k)\} \quad \text{where} \quad k = 0,1,2...n \qquad (5.1)$$

In proposed system, ray segments ($\mathbf{P_k}$ and $\mathbf{D_k}$) generated from ray tracer is stored in a ray-table. Each ray-table is associated with a particular gemstone cutting and a particular image size and position related to the position of viewer's eye. Figure 5.4 illustrates the idea of ray-table.

A ray-table contains w × h ray-lists where w and h are the number of rows and columns of the image in pixels.

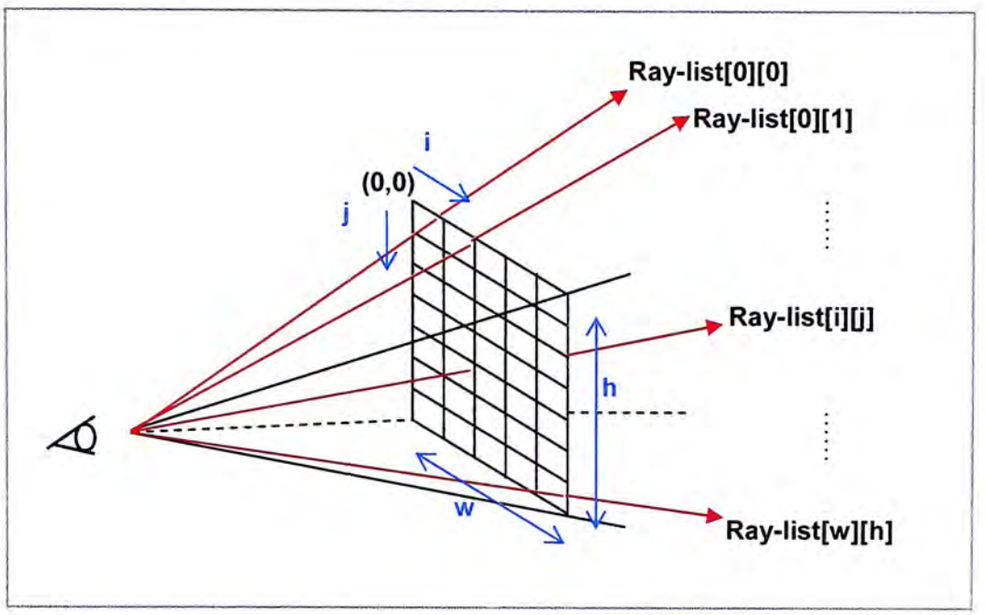

Figure 5.4 A ray-list buffer

5.2.1 Ray-list

A ray-list contains the data for the illumination calculations in the shading process. Each ray-list is associated with a pixel in image plane. The ray-list corresponding to the pixel at the i^{th} row and j^{th} column of the image plane is represented by ray-list as [i][j] . A ray is defined by its starting point and direction. A ray-list is a link-list of rays.

To minimize the data size, only the rays relevant to the shading stage are of stored. A ray-tree (see Figure 4.3) containing all the traced rays is reduced to a ray-list as will be discussed in the following paragraphs. Figure 5.5 illustrates the ideal ray paths traced in a gemstone with an ideal cutting which gives the highest brightness. In the figure, pixel rays are drawn as orange lines, reflected rays are drawn as blue lines,

transmitted rays are drawn as red line and the intersection points are drawn as green spots. Only five segments of a ray are traced as discussed previously.

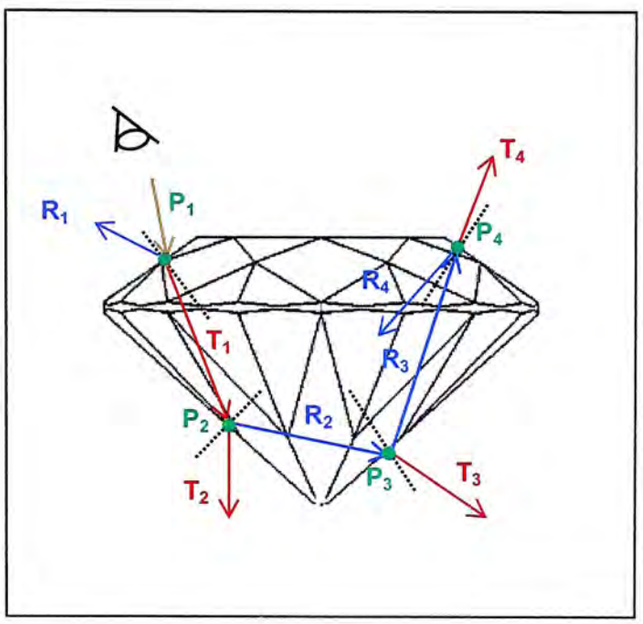

Figure 5.5 Reflected and transmitted rays traced in gemstone

Figure 5.6 shows the ray-tree of rays traced corresponding to Figure 5.5. The black line represents the pixel ray, the red branch represents the transmitted ray, the blue branch represents the reflected ray and the node represents the intersection point. The root node and branches in dotted line bounded by rectangular grids represent the rays that contribute to the calculation of shading directly, which will then be stored in a ray-list (a link-list of rays). The intersection point and direction of a useful ray are represented by a node and its selected branch, for instances P_1 and R_1; P_2 and T_2; P_3 and T_3; P_4 and T_4. The useful rays (bounded by rectangular grids in Figure 5.6) are then extracted from the ray-tree and stored in a ray-list as shown in Figure 5.7.

In case of total internal reflection, no transmitted ray is emitted, and therefore only the reflected ray is taken into account. When a pixel ray does not have any

intersection with the object, which means the ray does not hit any object, the whole ray-list will be empty.

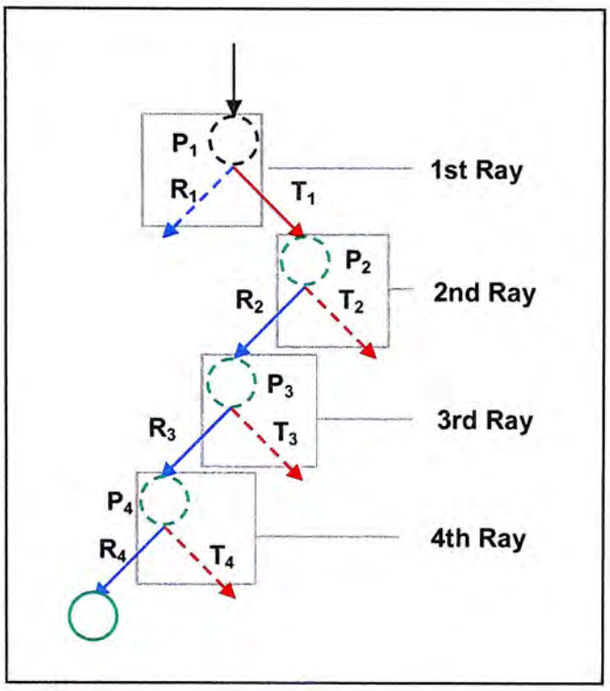

Figure 5.6 The ray tree of ray traced in Figure 5.3.

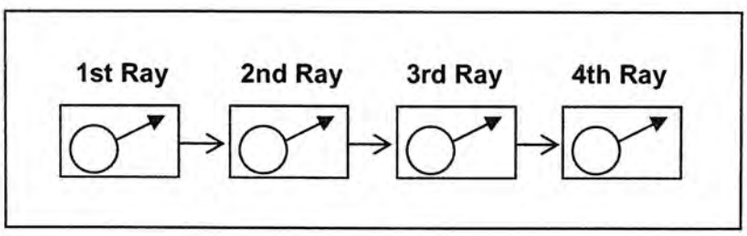

Figure 5.7 A ray-list structure

In the shading stage, the 1st Ray of ray-list is used for the calculation of specular reflection, which accounts for the shininess of the object; other rays of ray-list are used to calculate the specular transmission, which contribute to the fires in gemstone.

6 The Shading stage

During the shading stage, gemstone is rendered using data obtained from the pre-computation stage. In the system, the shading process is carried out in another program independent of the pre-processing part. The data retrieved from the ray-table are used to compute the shading taking into consideration parameters such as light source and the environment, the transparency and color of the gemstone.

6.1 Data retrieving process

To use the calculated data from the pre-computation stage, data must be retrieved from database first. Each ray-table is associated with a specific position of the viewer, the position and size of the image plane, and the gemstone cutting model. The ray-list is retrieved from the ray-table once the illumination equations of the corresponding pixel ray are in process.

6.2 Illumination equation

The illumination model is divided into two parts: illumination from light source and illumination from environment. The illumination from light source is determined by the relative positions of the light source and the ray; the illumination from the environment is obtained by refraction mapping of the environment as will be discussed in section 6.2.3.2. The illumination model applies to monochromic light only. Colored light has to be resolved into its RGB components before applying the model. R-factor and T-factor are introduced first as they play important roles in the illumination model.

6.2.1 R-factor and T-factor

R-factor and T-factor describe the amount of light ray that is reflected and transmitted across a boundary. A R-factor of 1 indicates that 100% of the light is reflected, where as a R-factor of 0 indicates that no light is reflected. Every ray has its own R-factor or T-factor. The contribution of a ray to the whole ray-tree is represented by its R-factor or T-factor. Figure 6.1 shows a ray-tree. F_r and F_t represent respectively, the R-factor and T-factor of the reflected and transmitted ray in the first level of the tree ($0 < F_r, F_t < 1$). Going down the tree, the contribution of the rays to the pixel intensity decreases, and becomes insignificant after a number of reflections and transmissions.

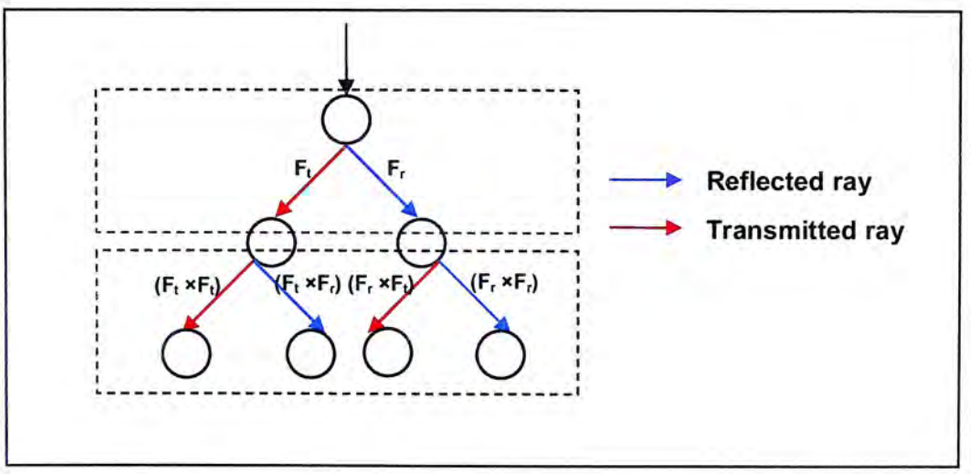

Figure 6.1 R-factor and T-factor in a ray tree

6.2.2 Illuminations from light source

By using data stored in the ray-table, the illuminations from the light source can be calculated as below.

6.2.2.1 Diffuse reflection

The intensity of diffuse reflection can be calculated by Equation (6.1). However, for the high reflective property of gemstones, diffuse reflection is small enough which can be neglected in our illumination model.

$$I_d = I_l k_d I_o (N \bullet L) \qquad (6.1)$$

6.2.2.2 Specular reflection

The 1st ray segment stored in the ray-list is a reflected ray. In Figure 6.2, reflected ray is in blue color, which is used to calculate the specular reflection. R-factor is assigned to the corresponding reflected ray. The specular reflection can be calculated using Equation (6.2). L represents the direction of light source, which is determined by the position of light source relative to the ray/ object intersection point, R represents the reflected ray, I_l represents the intensity of the light source, h represents the polishing level of the object's material, k_s represents the specular absorption constant, F_r represents the R-factor of the reflected ray.

$$I_{sr} = I_l k_s F_r (R \bullet L)^h \qquad (6.2)$$

6.2.2.3 Specular transmission

Starting from the 2nd ray segment stored in the ray-list, the ray segments are transmitted ray. In Figure 6.2, transmitted ray is shown in red color. Transmitted ray is used to calculate the specular transmission. A T-factor is assigned to a transmitted ray indicating the amount of light being transmitted.

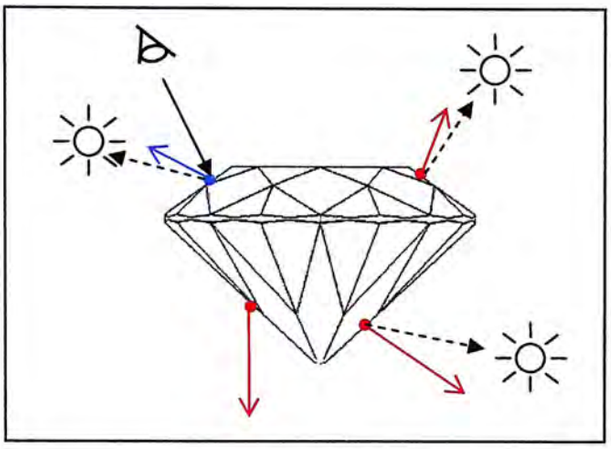

Figure 6.2 Rays from ray-list

The specular transmission can be calculated by Equation (6.3). L represents the direction of the light source, T_i represents the transmitted ray, I_l represents the intensity of light source, I_o represents the intensity of the object's color, h represents the polishing level of the object's material, k_{st} represents the specular absorption constant, F_{nt} represents the T-factor of the n^{th} transmitted ray.

$$I_{st} = \sum I_l I_o k_{st} F_{nt} (T_n \bullet L)^h \qquad (6.3)$$

The intensity of the object's color in the equation is used to compensate the error due to the omission of dispersion factor. There is an approximation that, when white light pass through a refractive object, the refractive ray will be affected by the color of object. Figure 6.3 shows that when a white light pass through a red glass, the color of the transmitted ray will change to red.

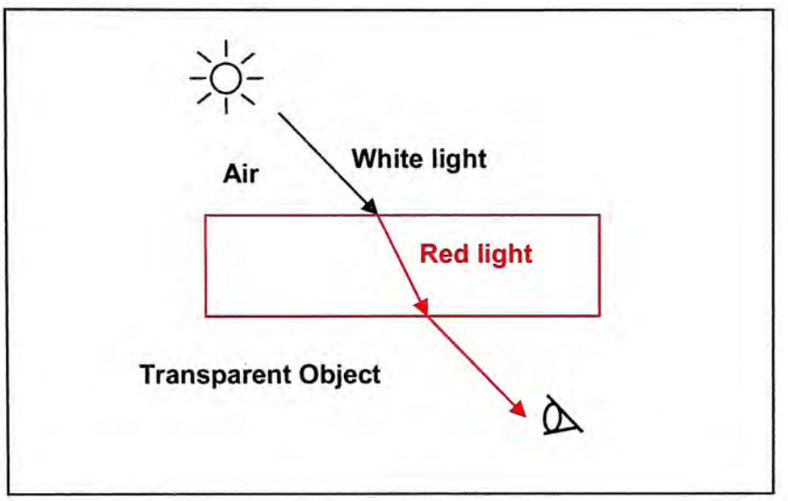

Figure 6.3 Approximation of the color of transmitted ray

6.2.2.4 Light Obstruction Test

There are two conditions that the light source does not have any contribution to the shading. In the system, these conditions must be checked before having the specular reflection and transmission computation.

Condition 1: Angle between L and R is greater than 90 degree. See Figure 6.4(a).

Condition 2: The plane containing the exit ray is blocked by other planes from the light source. Verification can be completed by creating a light ray from the light source to a center point of a plane. Find the light ray/ plane intersection with other surface. Compare the line connecting the light source and the center point of the plane with the line connecting the light source to the resultant point of light ray/ plane intersection. If it is larger, the light source of certain

plane is blocked by other planes. Figure 6.4(b) shows that L is blocked by other surface before it reaches the red plane.

As ray-list does not provide any information about the object geometry, the corresponding gemstone model has to be stored for the light obstruction test.

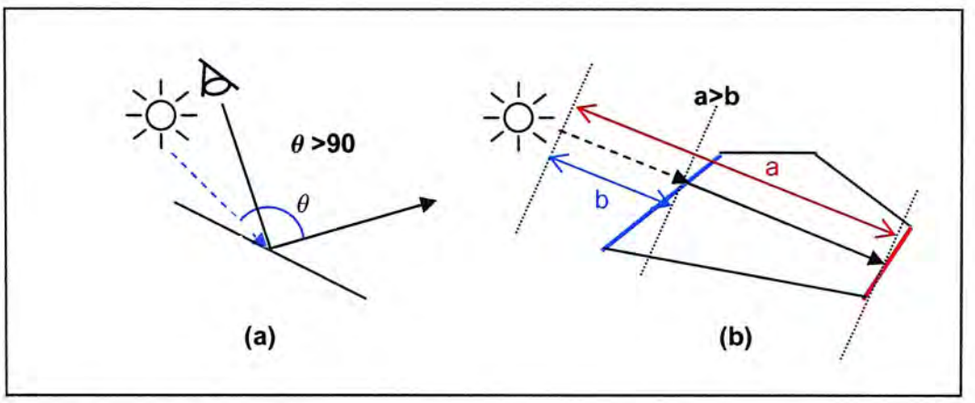

Figure 6.4 Cases of light source with no contribution to shading

6.2.3 Illumination from environment

6.2.3.1 Virtual Cube

In order to imitate the effect of the environment on the rendering of gemstone, we adopted the environmental mapping technique based on a virtual cube is adopted.

A gemstone model is enclosed by a virtual cube as shown in Figure 6.5. The center of the gemstone is aligned with the center of the virtual cube. The environment is then projected onto the six planes of the cube, which forms the environment textures.

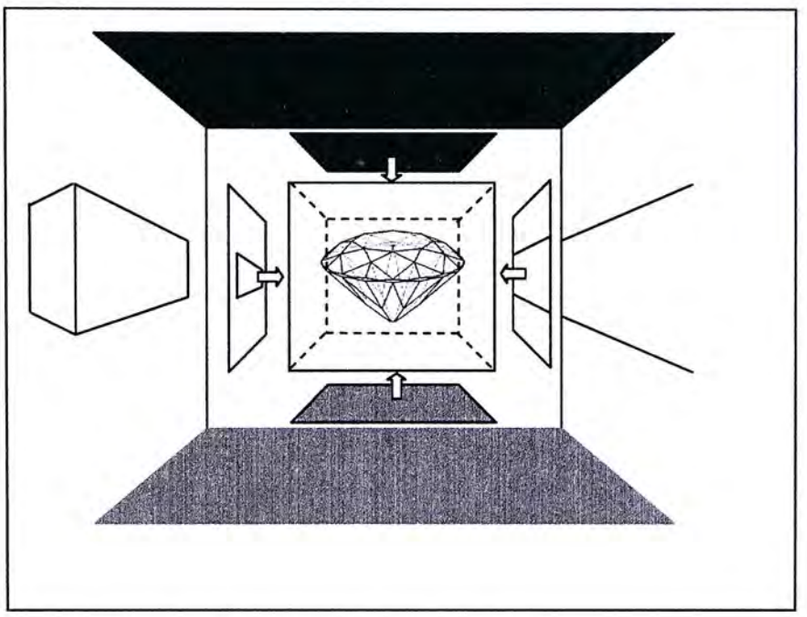

Figure 6.5 A virtual cube in the environment

The ray-table is used to calculate the effect of the environment on the final image. For each pixel of the final image, its corresponding ray-list is retrieved from the ray-table. For each ray, its intersection with the virtual cube is computed. The corresponding pixel on the environment texture is then retrieved and applied to the final image. An example is shown in Figure 6.6. Ray/ plane intersection calculation as discussed in Section 4.4.1 is used to locate the intersection between the rays and cube faces.

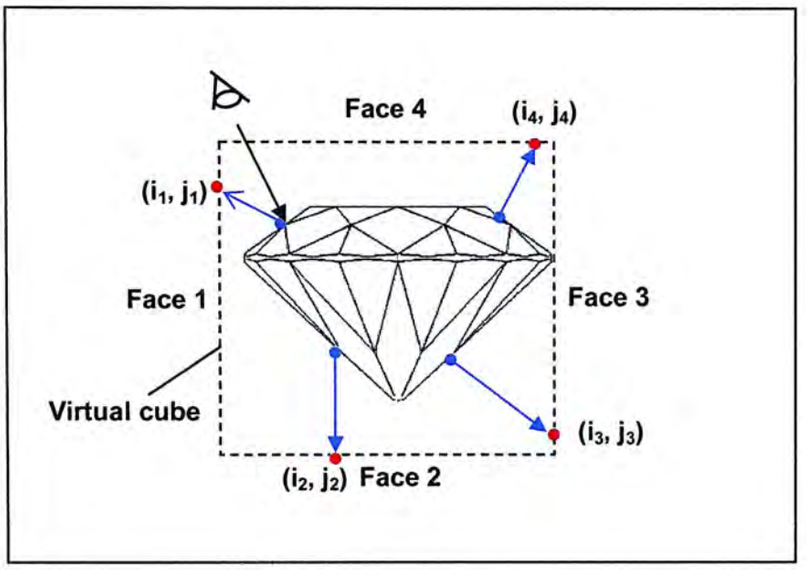

Figure 6.6 The intersections of rays and virtual cube

6.2.3.2 Refraction map of the environment

The refraction map of the environment is computed based on Equation (6.4). In Equation (6.4), the alpha value α denotes the transparency ratio of a gemstone. The intensity of the pixel at position (i_m, j_m) corresponding to the virtual cube is denoted as $I_{(i_m, j_m)}$, where m is the face's index. The intensity of the pixel corresponding to the intersection of the nth ray in a ray-list and the virtual cube is denoted as I_n such that $\forall I_k \in I_{(i_m, j_m)}$.

$$I_r = (I_1\alpha + I_o(1-\alpha))F_{1r} + \sum_{k=2}^{n} (I_k\alpha + I_o(1-\alpha))F_{kl} \qquad (6.4)$$

where I_o is the object color intensity, F_{nt}/F_{nr} is the T-factor /R-factor of the nth transmitted /reflected ray, I_r indicates the total intensity at the first ray/object intersection point.

If there is no ray segment in a ray-list the ray does not hit the object (Figure 6.7). The intensity of the texture image pixel corresponding to the ray/cube intersection will be allocated to the find image pixel.

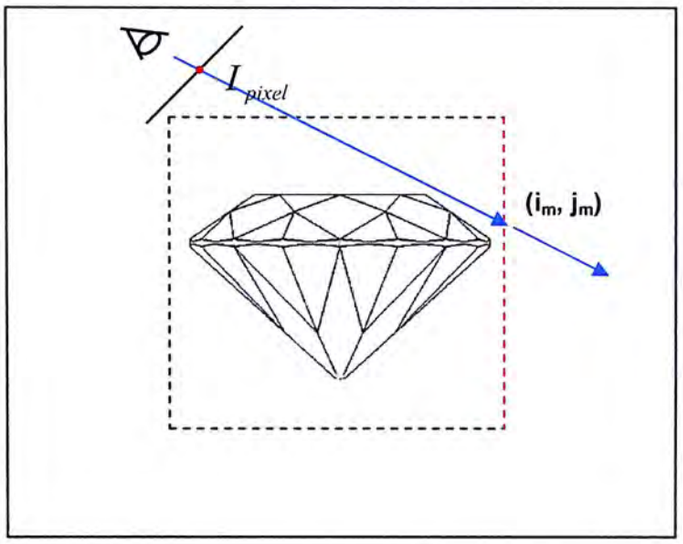

Figure 6.7 No intersection between the pixel ray and the object.

6.2.4 Total illumination

The total illumination is the sum of the illumination from the light source and the illumination from the environment, as illustrated in Equation (6.5). Intensity of the final image pixel is represented by I_{pixel}

$$I_{pixel} = I_{sr} + I_{st} + I_r \qquad (6.5)$$

7 Implementation

In the proposed system, the refractive rendering of gemstone can be performed interactively in a 3D environment with changing lighting condition and object orientation. All programs developed in the system are implemented in Visual C++; the graphic rendering tasks are handled by OpenGL™ library. The system is composed of three different modules, namely, a module for the import of gemstone model, a module for pre-computation stage, and a module for shading.

7.1 The Gemstone model

Gemstone cutting models in brilliant round, marquise, heart, oval, tear and princess are built using common CAD software, such as SolidWorks, Rhinoceros, 3DStudio MAX. See Figure 7.1. The geometric data file of the gemstone model in ASC format can be imported into the system.

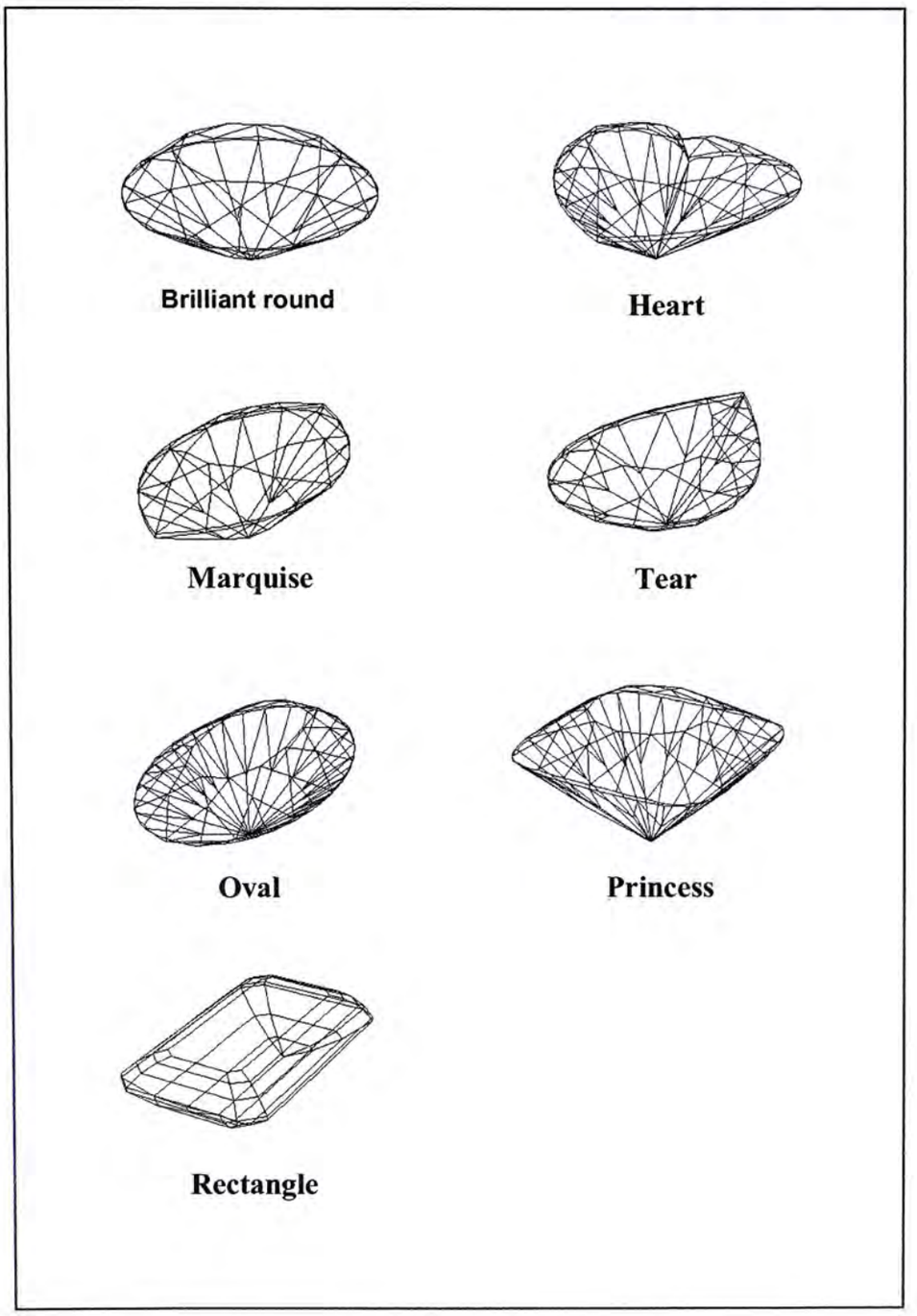

Figure 7.1 Cuttings of gemstone

7.2 Pre-computation Stage

In the pre-computation stage, a series of ray-table for different object orientation are computed using conventional ray-tracing method.

Given a gemstone model in ASC format of the desired cutting, the refractive index, and other parameters such as position of viewer, position and size of view plane are specified before the ray-tracing process is involved. The results of the ray-tracing process are stored in a ray-table. A series of ray-table for different object orientation are created by alternating the angle of rotations about the principle axes (i.e. x, y, z axes) of the object at the girdle center point P_c. Figure 7.2 shows the position of point P_c which lies on the girdle center of a gemstone with brilliant cut. In the system, angle between successive orientations of the gemstone model is 0.1 Radian as shown in Figure 7.3.

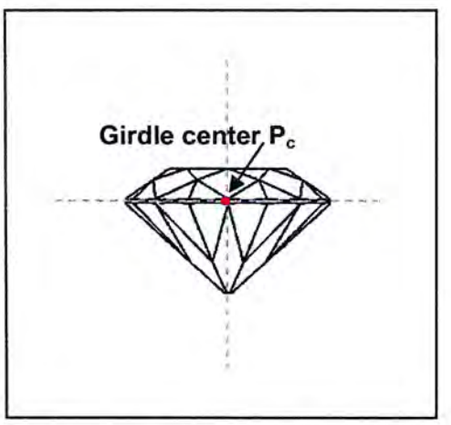

Figure 7.2 The position of point P_c lies on the girdle center of the brilliant cut

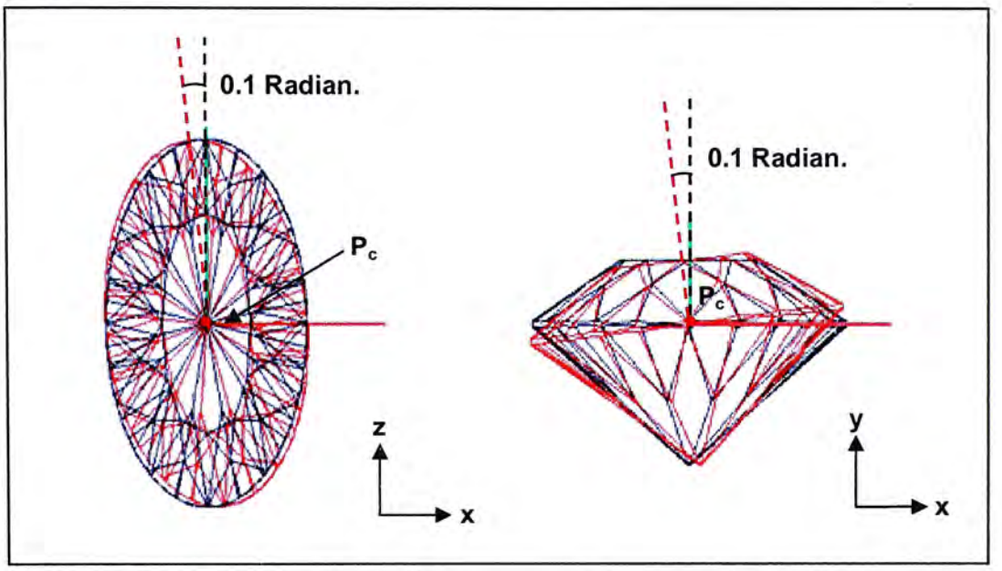

Figure 7.3 Ray-list buffer is captured for every 0.1 Radian

Each ray-table is stored in binary format with a file extension (.OUT). Binary format is used instead of simple text file format because the file size of the ray-table can be reduced about four to five times. Ray-tables of different object orientations are created for gemstone model of a specific cutting, refractive index, and viewing parameters. They are then grouped together to form a ray-table database.

7.3 Shading Stage

In the shading module, some of the parameters can be changed by user interactively. They are:

- ***Light Source***:

  Position and Color Intensity

- ***Property of gemstone***:

  Transparency and Color Intensity

- ***Orientation of gemstone***:

  Rotation about the principle axes at the girdle center

- ***Change in Environment:***

  Background image

Firstly, all parameters in the program are set by the user. The geometry of the gemstone is imported into the system for the light obstruction test. Once an object orientation is specified by a user, a searching process will be involved to find the corresponding ray-table located is retrieved from the database for use in the rendering process.

In the rendering process, the ray-list associated with each image pixel is retrieved. Each ray-segment in the ray-list makes intersection with the virtual cube. The intensity at the intersection point is used for computing the intensity of the image pixel using the illumination model discussed in chapter 6. A flow chart describing the process is shown in Figure 7.4

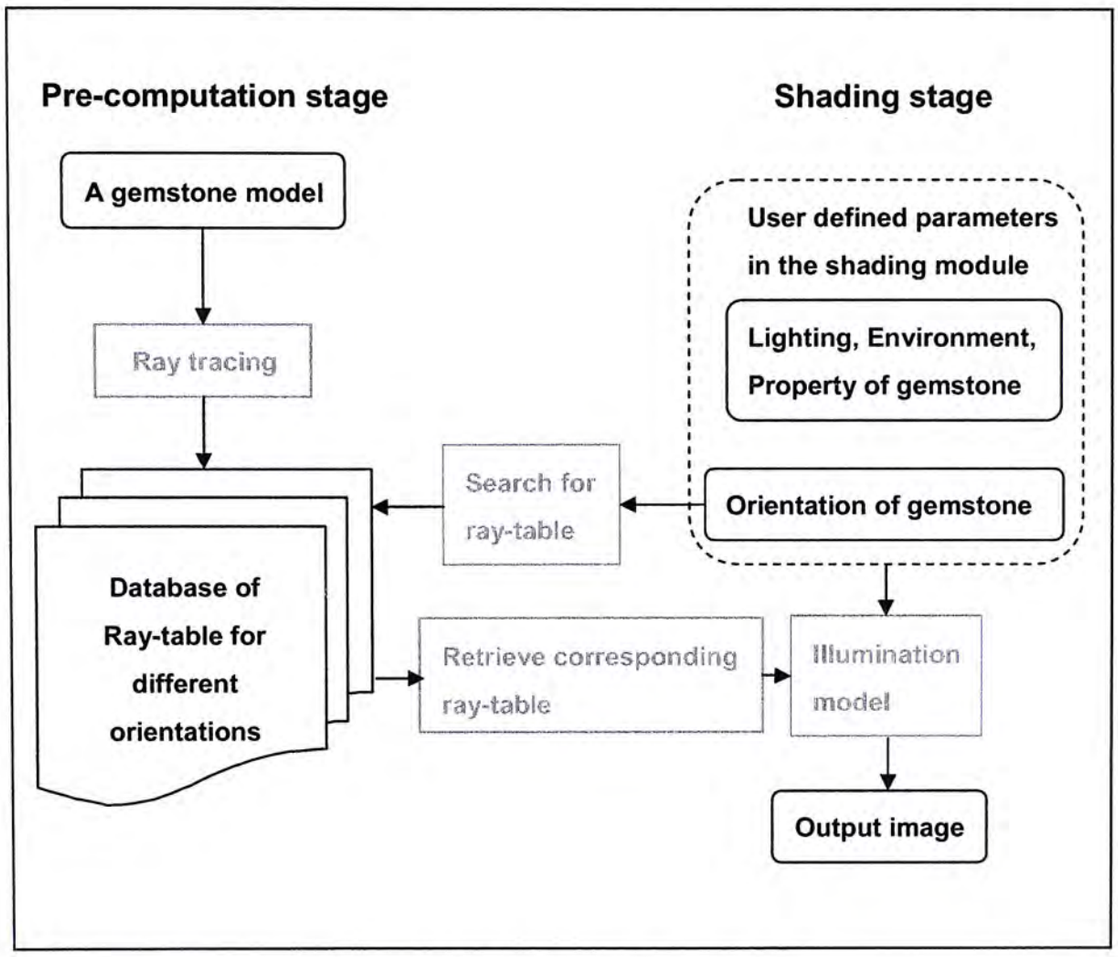

Figure 7.4 Flow chart of our system

The method of gemstone rendering is discussed in the above section. Using this method an image of the rendered gemstone as shown in Figure 7.6 can be obtained. In order to render a jewelry design interactively, jewelry component must be rendered together with the gemstone. In the following section, the extension of the above rendering method for display 3D jewelry design is discussed.

Figure 7.5 Image of a diamond generated by our system

In order to render a jewelry design, such as diamond ring, a few more steps are required in the shading stage. Before rendering the gemstone, the 3D environment, including the jewelry components, are projected onto the virtual cube so that the effect of 3D environment is taken into account. The images of the six cube faces are produced using OpenGL functions. It is performed by setting up six different view points, and for each view point, the projection of the 3D environment is drawn in the auxiliary buffer supported by OpenGL. The images are then created using the pixel information inside the buffer. The six images are updated in runtime when there is any change in the environment. Figure 7.6 shows the projected images on the faces of the virtual cube corresponding to the gemstone's orientation.

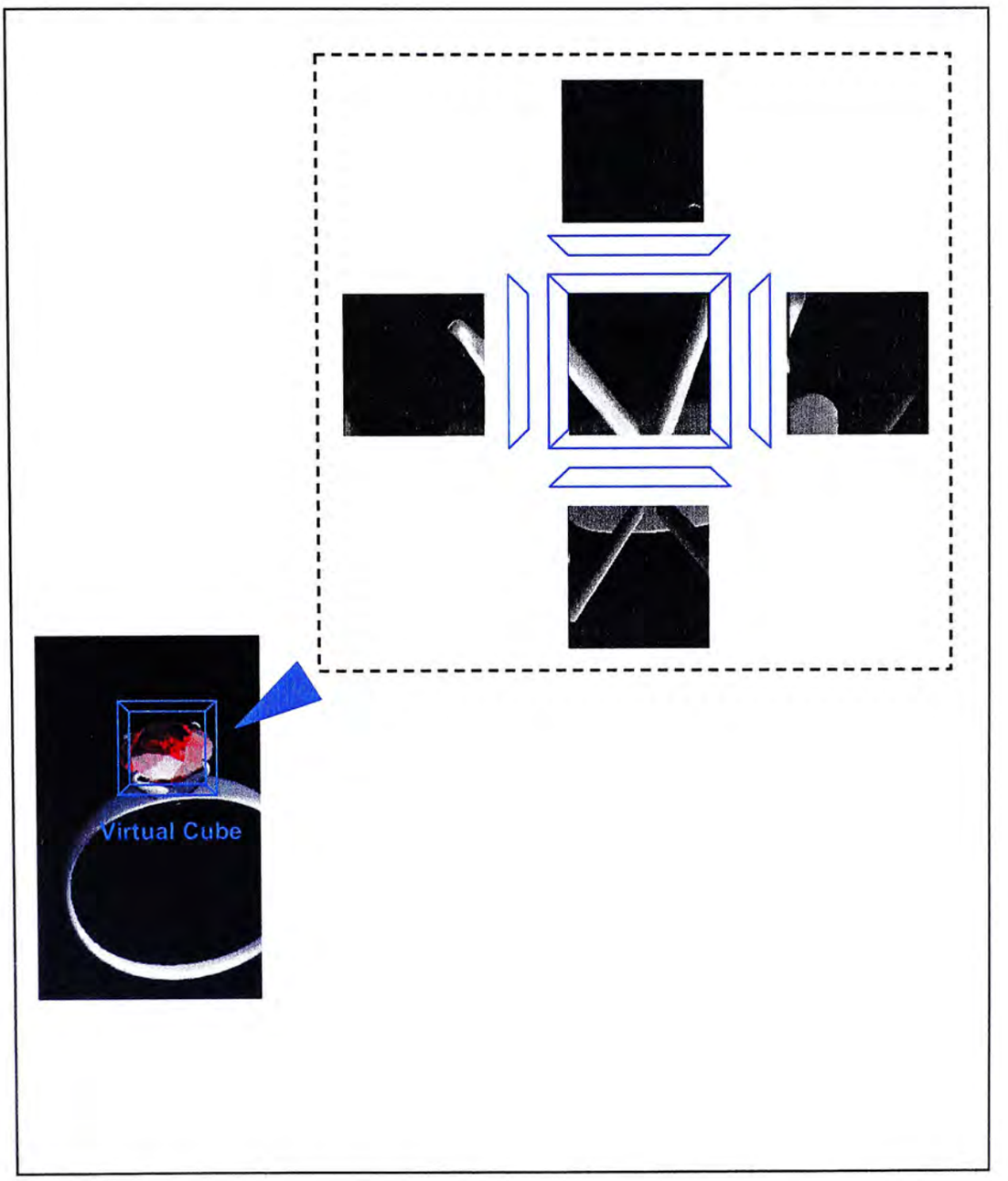

Figure 7.6 Image projection on the faces of the virtual cube

Once the projected images on the virtual cube are constructed, the rendering of the gemstone begin. The image of a rendered gemstone is created as a bitmap and is texture mapped onto a plane in 3D space. Figure 7.7 shows the plane with a gemstone texture located in 3D space. The point $\mathbf{P_c}$ on the texture is always set as (0, 0 ,0) in 3D space.

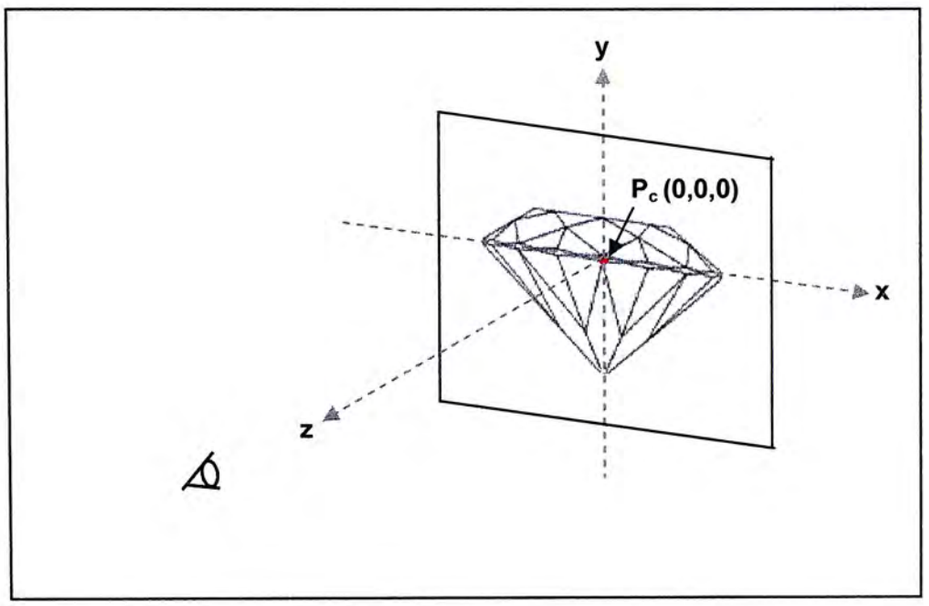

Figure 7.7 The plane with a gemstone texture is located in 3D space

In Figure 7.8, the ring in 3D space and is rendered using OpenGL functions. The image of a diamond is mapped onto the plane as indicated.

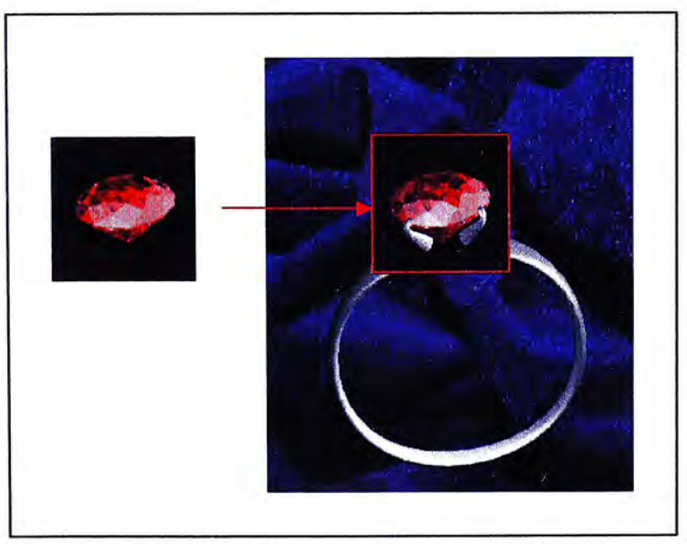

Figure 7.8 A texture mapping of a diamond on a plane in 3D space.

Alpha blending is used to blend the diamond image with the image of the ring. The background of the diamond image is set as transparent and is merged with the

ring and the whole environment. As a result, an image of diamond ring can be produced, which is shown in Figure 7.9.

Figure 7.9 Image of a diamond ring generated by our system

Since the image plane of the diamond is always normal to the viewer, whenever there is a change in object orientation, a new diamond image is created, and the above process is repeated.

8 Results

8.1 Variables setting

In the system, the generated images are controlled by a set of variables. The effects of different variables are shown below. In each case, all parameters are kept constant except the one which is to be examined.

8.1.1 R-factor and T-factor

Figure 8.1 shows the effect due to variation of R-factor on the final image. From left to right, the R-factor of the model is 0.2, 0.4, 0.6 and 0.8 respectively. The result shows that the specular effect is more significant with a higher R-factor. This is because the first reflected ray is the only ray contributing to the specular reflection of the object. A higher R-factor results in more light being reflected in the first reflection.

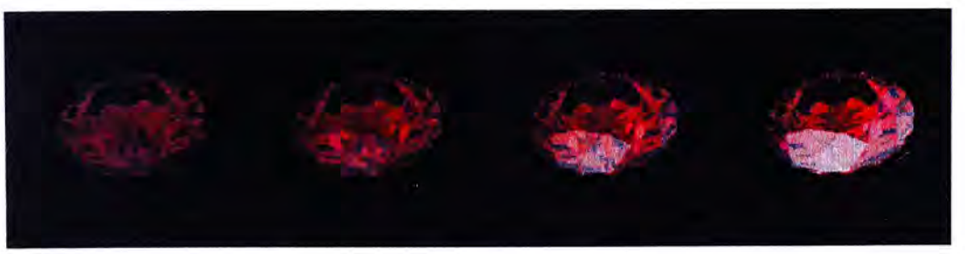

Figure 8.1 Variation of R-factor

Figure 8.2 shows the effect of a variation in T-factor on the final image. From left to right, the T-factor of the model is 0.2, 0.4, 0.6 and 0.8 respectively. The result shows that the internal pattern due to transmission is brighter with a higher T-factor while

the specular reflection on the surface remains constant. This is because the T-factor accounts for the amount of light passing through a gemstone.

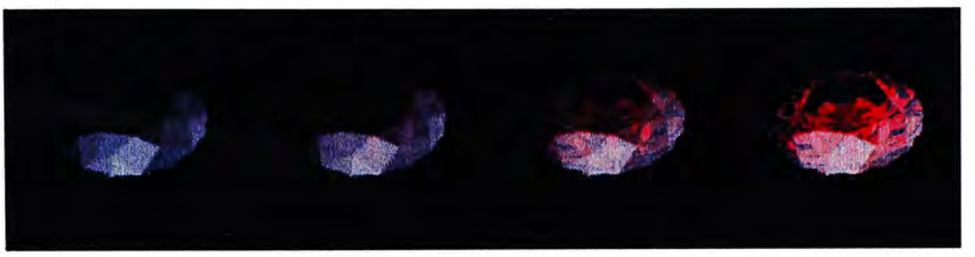

Figure 8.2 Variation of T-factor

8.1.2 Specular reflection constant k_s

Figure 8.3 shows the effect due to variation of k_s on the final image. The value k_s change from 0.6 to 0.9. The higher the value of k_s, the brighter the specular reflection on the surface.

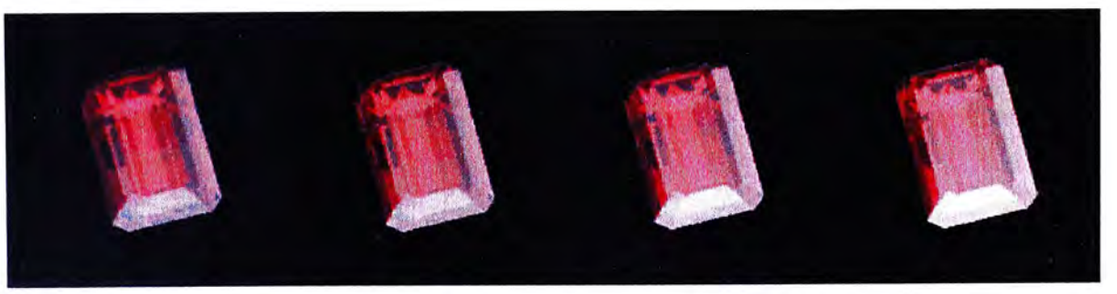

Figure 8.3 Variation of k_s from 0.6 to 0.9

8.1.3 Color and orientation

Figure 8.4 shows the variation of color and orientation on the images of diamond. The diamond model is a diamond with brilliant round cutting and a refractive index of 2.44. They are in red, green, blue and white.

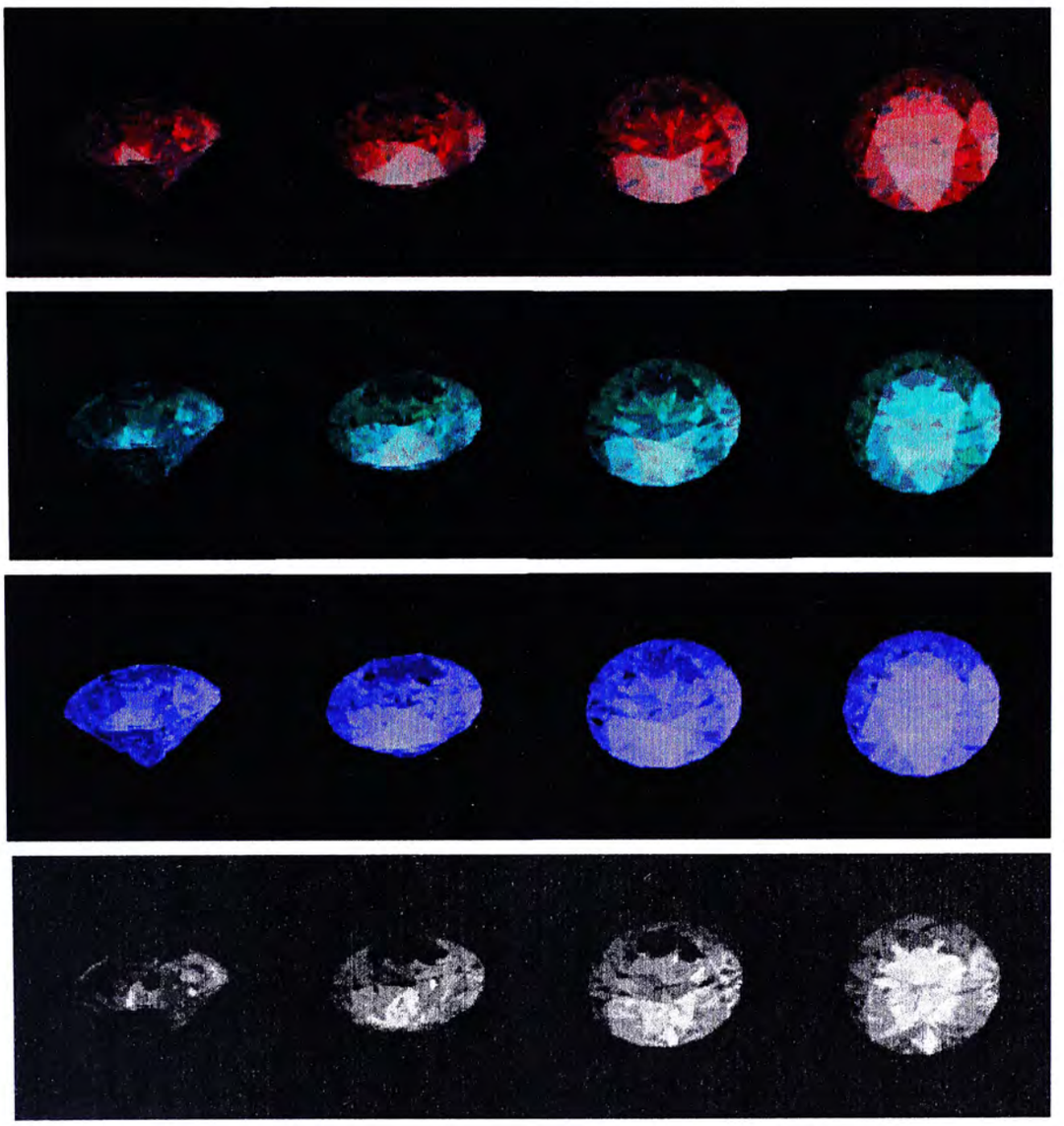

Figure 8.4 Variations of the color and orientation

8.1.4 Lighting

Figure 8.5 shows the variation of lighting on the rendering images of an intense red garnet. The garnet model is a garnet with marquise cutting and a refractive index of 1.70. In the figure, the position of one of the light source is changed from left to right.

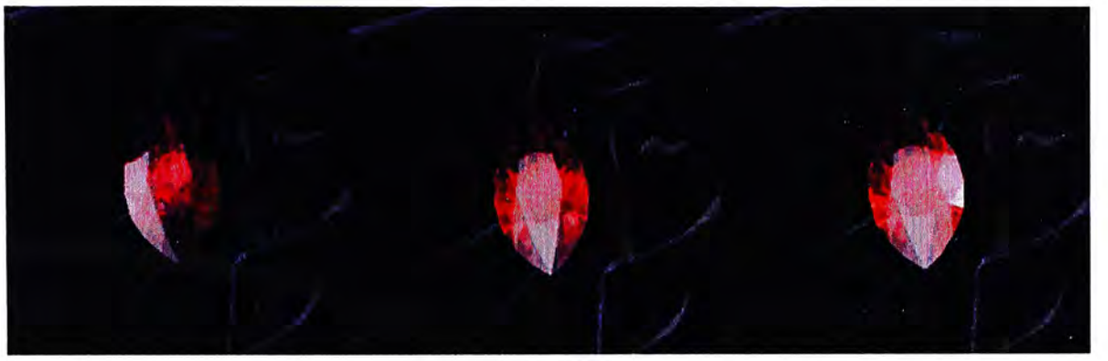

Figure 8.5 Variation of light source position from left to right

8.1.5 Refractive index

Figure 8.6a and b shows the variation of refractive index on the images of an object with rectangular cutting and tear cutting respectively. The refractive index of first object is 1.40 (i.e. glass); the second object is 1.70 (i.e. sapphire); the third object is 2.44 (i.e. diamond). The higher the value of the refractive index, the more sparkling are the facets in the object. This explains the formation of fires in diamond because of its high refractive index.

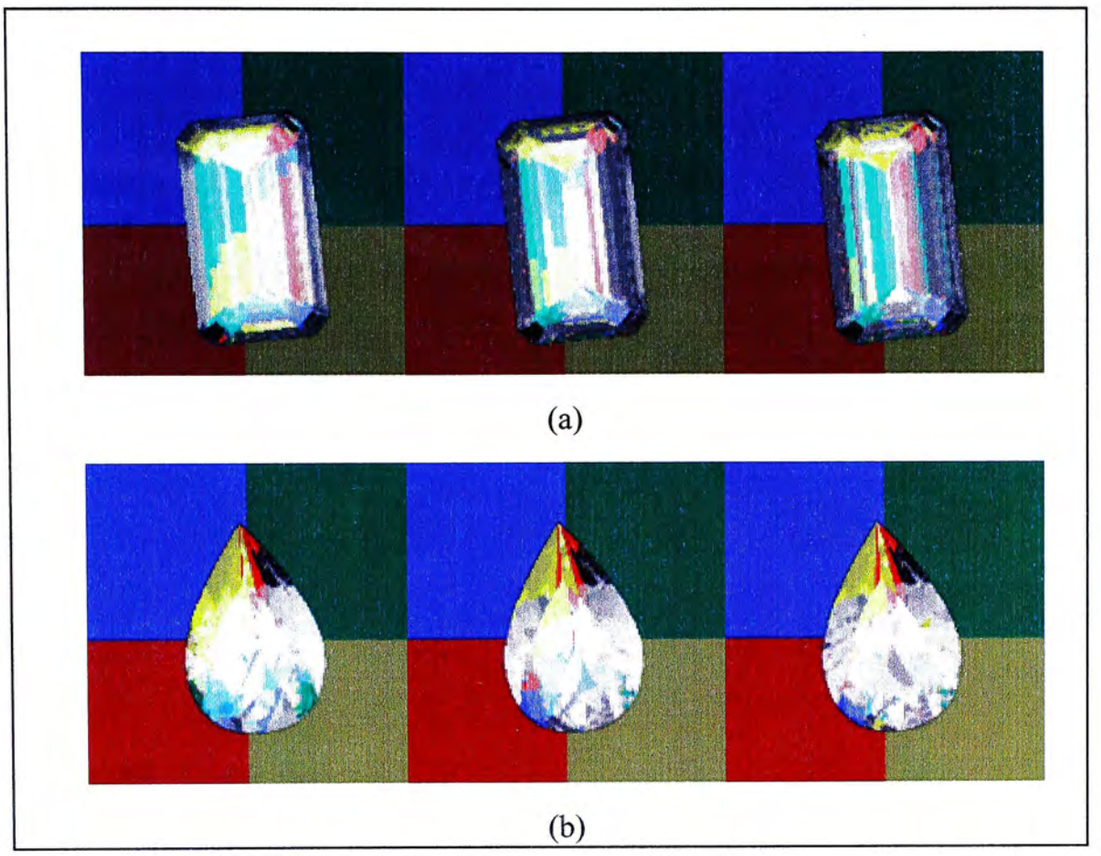

Figure 8.6 Variation of increasing value in refractive index

8.1.6 Transparency

Figure 8.7 shows the variation of transparency on the rendering of a sapphire with heart cutting. The transparency value of the sapphire changed from 0.0 (opaque) to 1.0 (transparent).

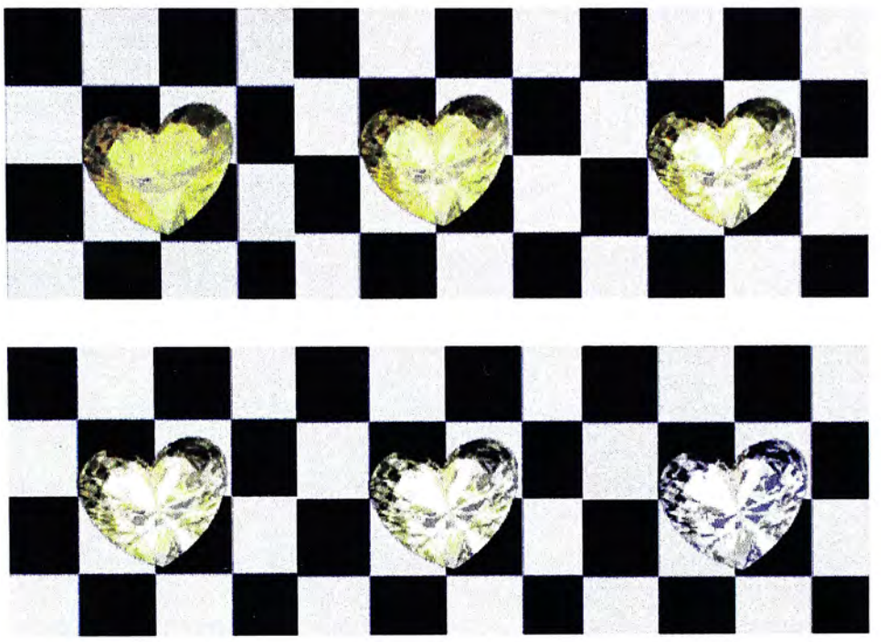

Figure 8.7 Variation of transparency value in heart

8.1.7 Background

Figure 8.8 shows the variation of the background on the rendering of a sapphire with marquise cutting. The result shows that the colors of the sapphire are affected by the background.

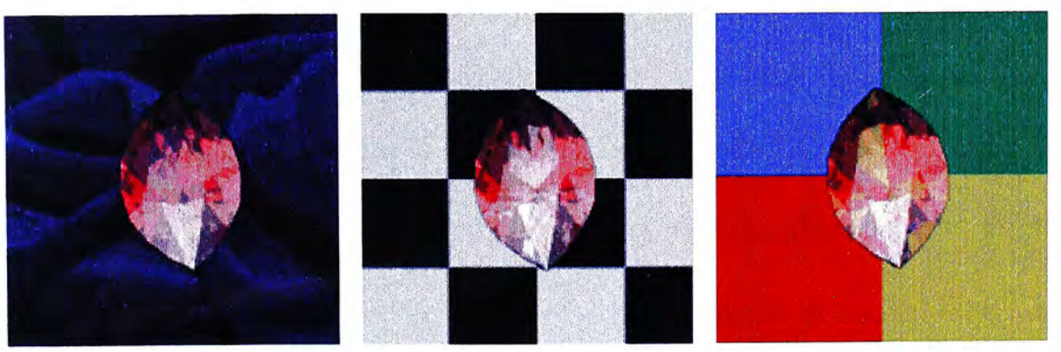

Figure 8.8 Variation of background in marquise cut

8.2 Computational speed

8.2.1 Analytical results

The performance of the system is mainly affected by the time (T_p) required for the construction of ray-tables in the pre-computation stage and the rendering time (T_s) in the shading stage. Both of them depend on the number of polygons (p), the image size (s) and the number of the internal reflection (r). In the system, the objects are represented as polygons. Image size is the number of pixels in the image plane as used in the ray tracing process. The number of the internal reflection corresponds to the number of times a ray is traced inside an object.

8.2.1.1 Pre-computation stage

In the pre-computation stage, the most time consuming process is the ray/ plane intersection. Since the number of ray/plane intersection is proportional to the number of polygons composing the object, the pre-processing time $T_p(p)$ is $O(p)$.

With an increase in image size, more pixel rays are generated. More rays have to be traced which means that more ray/ plane intersection process is required. Since the number of rays generated is proportional to the number of pixel in the image plane, the pre-processing time increase is proportional to the image size as well. And for each ray generated, all p polygons may have to be processed. Hence, T_p is $O(ps)$. As each ray has to be traced r times, where r is the number of internal reflection, the overall preprocessing time is of $O(psr)$.

8.2.1.2 Shading stage

During the shading stage, the most time consuming processes are the light obstruction test and the illumination calculation. The processing time of the two processes is denoted by T_{lot} and T_i respectively.

In the light obstruction test, a line connecting the light source and the face center of a given polygon is constructed. Intersections between this line and other polygons of the object are detected. Hence, given a light source, p intersection tests have to be performed for each polygon. The process is repeated for every polygon. Thus, $T_{lot}(p)$ is $O(p^2)$. However, as obstruction test is not performed under Condition 1 mentioned in 6.2.2.4, the order of $T_{lot}(p)$ is $O(p^n)$, where 1<n<2.

Consider the illumination calculation with an image size of s pixels, s ray-lists are to be processed, the time spend on the illumination calculation is thus $O(s)$. As one internal reflection increases the number of ray-exit point by one. T_i is of $O(sr)$.

8.2.2 Experimental results

The experimental system was implemented on a 3.2GHz PC with 2GB of RAM with a GeForcesFX 5200 graphic card. Experiments were performed to measure the performance of the system with changes in the number of polygons, image sizes and number of internal reflections. The experimental results are then compared with the analytical results.

8.2.2.1 Varying number of polygons

Table 8.1 shows that when the number of polygons of an object increases, the pre-processing time increases and the frame rates of rendering process decreases. The

models being tested are of the same shape and size, with different number of polygons. The resolution of the image plane for the ray- table (image size) referring to Table 8.1 is 150×150 and the storage size are around 1MB. Table 8.2 lists the results for others gemstone models.

No. of polygons	Pre-processing time (second)	Frame rate per second
232	26.771	1.620
638	67.092	0.345
728	75.383	0.263
1034	110.623	0.1425
1626	159.393	0.0771

Table 8.1 Results for rendering a gemstone with different number of polygons

Gemstone model	No. of polygons	Pre-processing time (second)	Frame rate per second
Brilliant round cut	184	23.890	1.597
Marquise cut	126	15.828	1.544
Tear cut	174	16.312	1.508
Heart cut	386	48.546	0.662

Table 8.2 Results for rendering other gemstone models

Figure 8.9 shows the graph of the number of polygons against the pre-processing time , and is approximately linear.

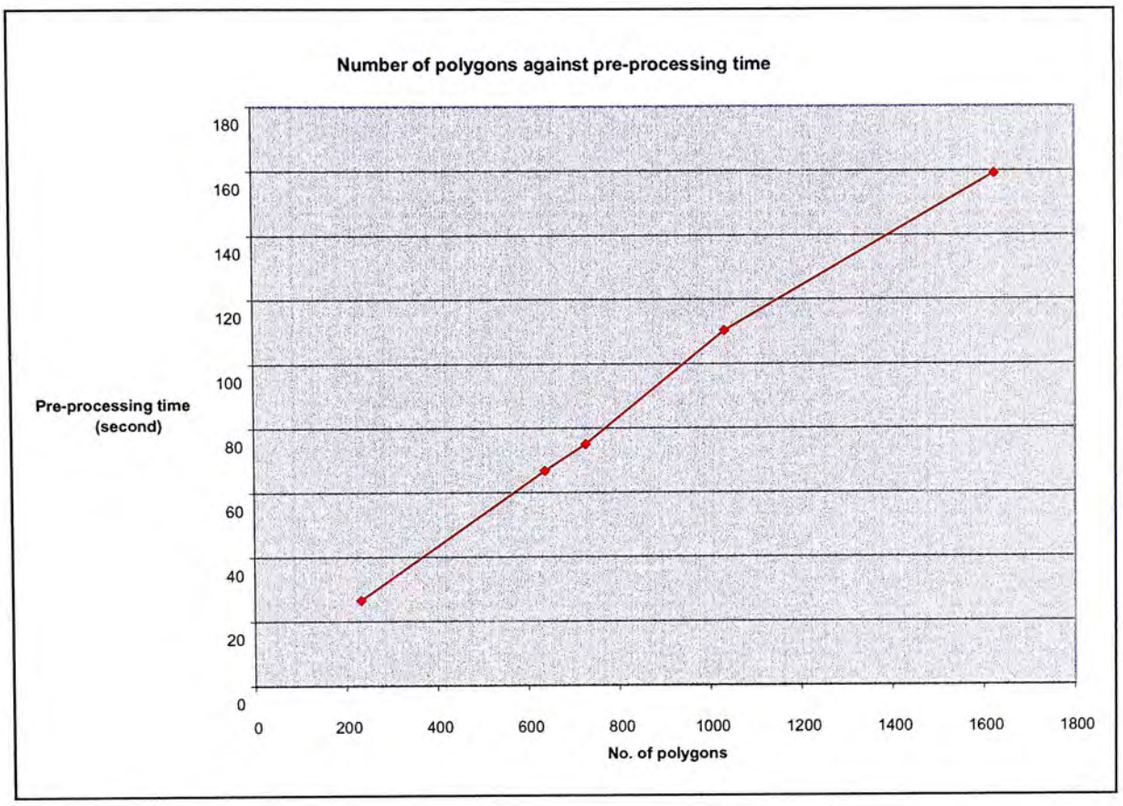

Figure 8.9 Number of polygons against pre-processing time

Figure 8.10 shows the graph for the number of polygons against the frame rate. Figure 8.11 is the graph in $\log_{10}$ scale. The slope of the graph in Figure 8.11 is -1.8 which means the order of the number of polygons against frame rate is around -1.834. As the time require for light obstruction test dominates the total shading time, so we can say that the order of the number of polygons against T_{lot} is around 1.8.

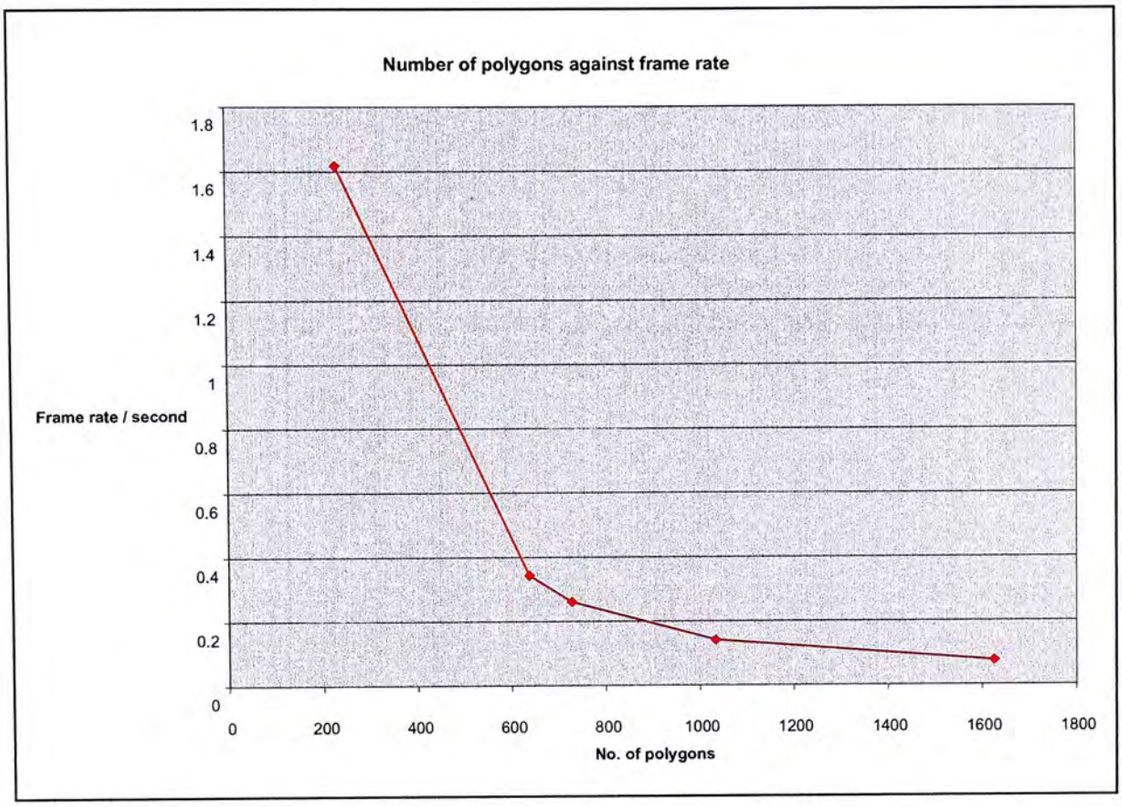

Figure 8.10 Number of polygons against frame rate

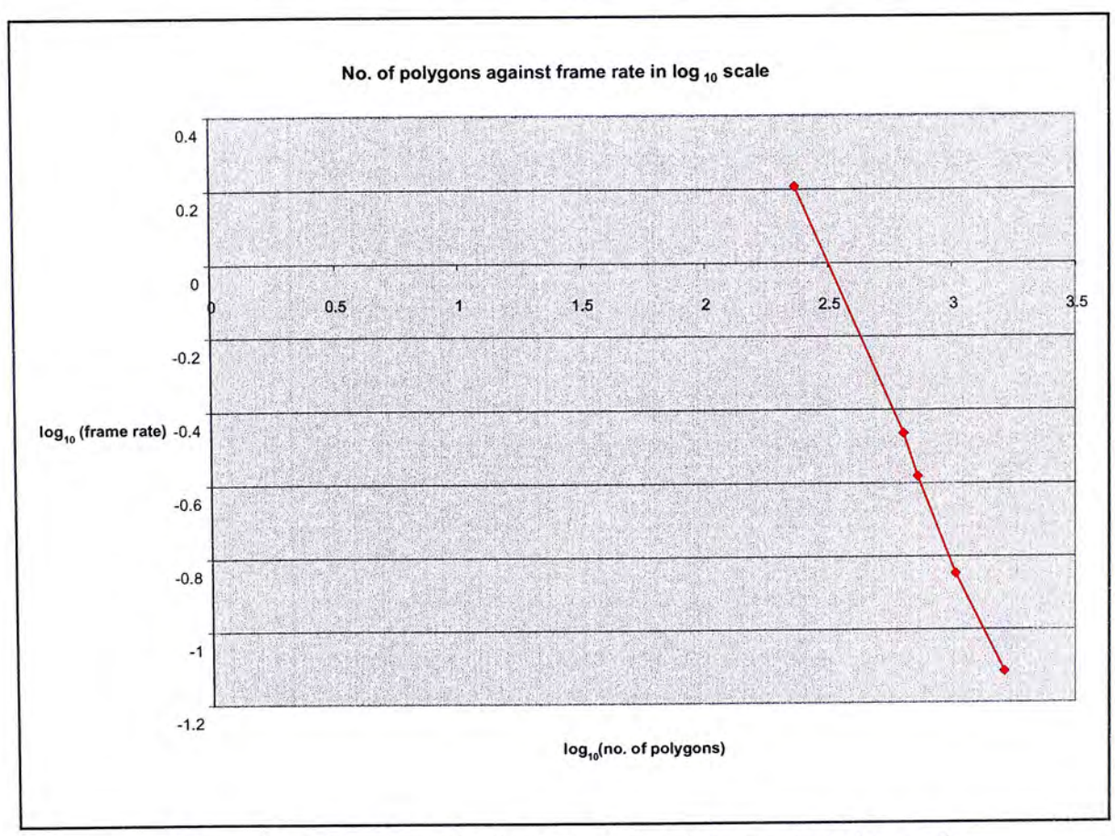

Figure 8.11 Number of polygons against frame rate in log scale

For the rendering of jewelry, the frame rate is lower than that in the rendering of gemstone. It is because additional resources are required for the projection of images onto the cube faces. Figure 8.3 lists the results for the rendering of a ring with a brilliant round diamond ring and a rectangular garnet ring.

Ring model	No. of polygons	Pre-processing time (second)	Frame rate /second
Ring with a brilliant round diamond	184	23.890	0.690
Ring with a rectangular garnet	126	20.147	0.808

Table 8.3 Results for the rendering of jewelry designs

8.2.2.2 Varying the image size

Table 8.4 shows that when the image size increases, the pre-processing time increases and the frame rates in the rendering process decreases. The models being tested are of the same shape and with the same numbers of polygons (232 polygons).

Image size	Pre-processing time (second)	Frame rate / second
50×50	2.503	1.634
100×100	9.738	1.508
150×150	21.400	1.348
200×200	37.639	1.129
250×250	58.582	0.972

Table 8.4 Results of a rendering gemstone with different image sizes

Figure 8.12 shows the graph for the image size against pre-processing time. Figure 8.13 shows the graph for image size against frame rate respectively. The graph in Figure 8.12 shows that the relation between the image size and the pre-processing time is linear. The graph in Figure 8.13 shows the relation between the image size and the rendering speed is linear.

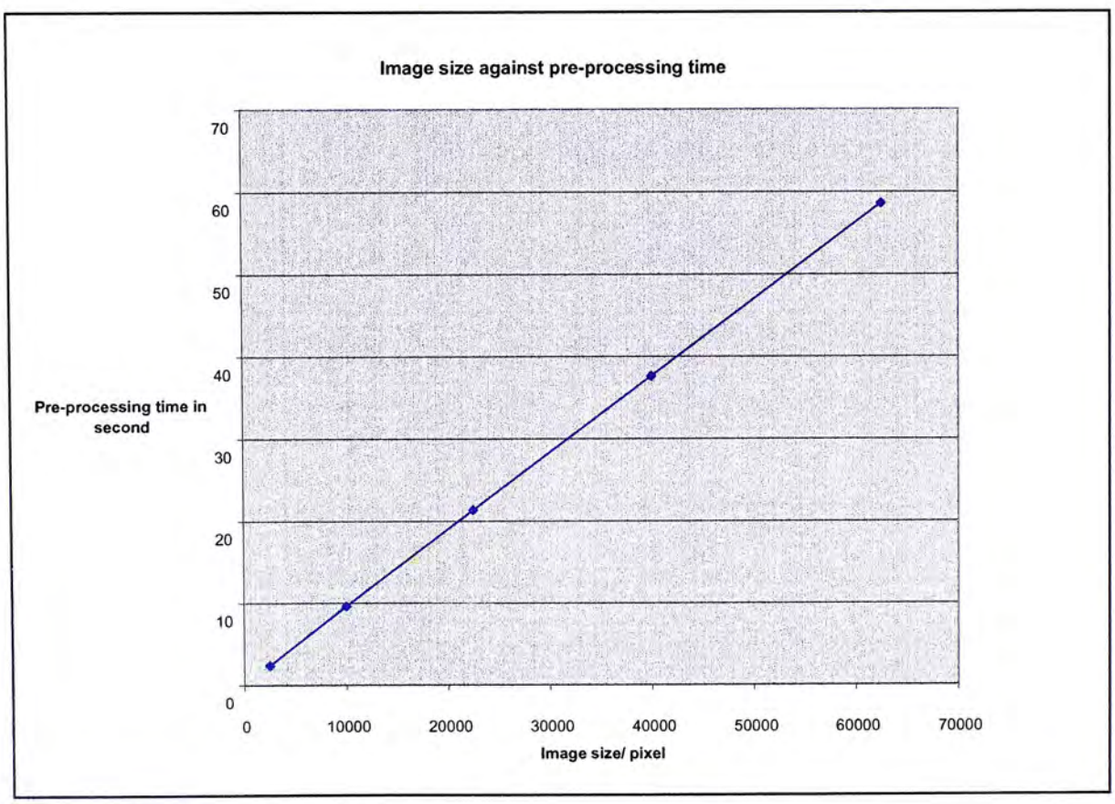

Figure 8.12 Image size against pre-processing time

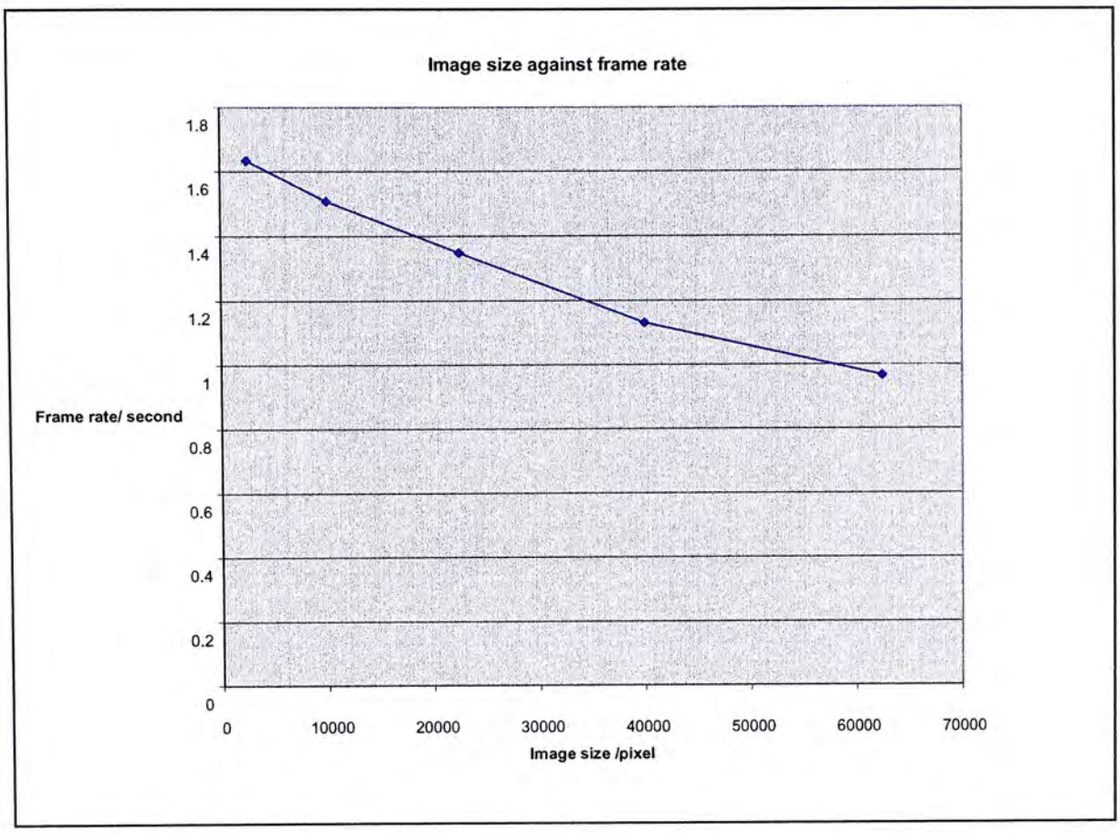

Figure 8.13 Image size against frame rate

8.2.2.3 Varying the number of internal reflection

Table 8.5 shows that when the number of internal reflection traced in the object increases, the pre-processing time increases and the frame rates in the rendering process decreases. The models being tested are of the same shape, same numbers of polygons and with the same image size. The object is composed of 232 polygons and the resolution of the image plane for the ray-table buffer (image size) is 150×150.

Number of counts	Pre-processing time (second)	Frame rate / second
3	18.723	1.405
5	21.400	1.348
7	24.031	1.302
9	28.908	1.263

Table 8.5 Results of rendering a gemstone with different number of internal reflection

Figure 8.14 and Figure 8.15 show the graph for the number of internal reflection against the pre-processing time, and the number of internal reflection against the frame rate respectively.

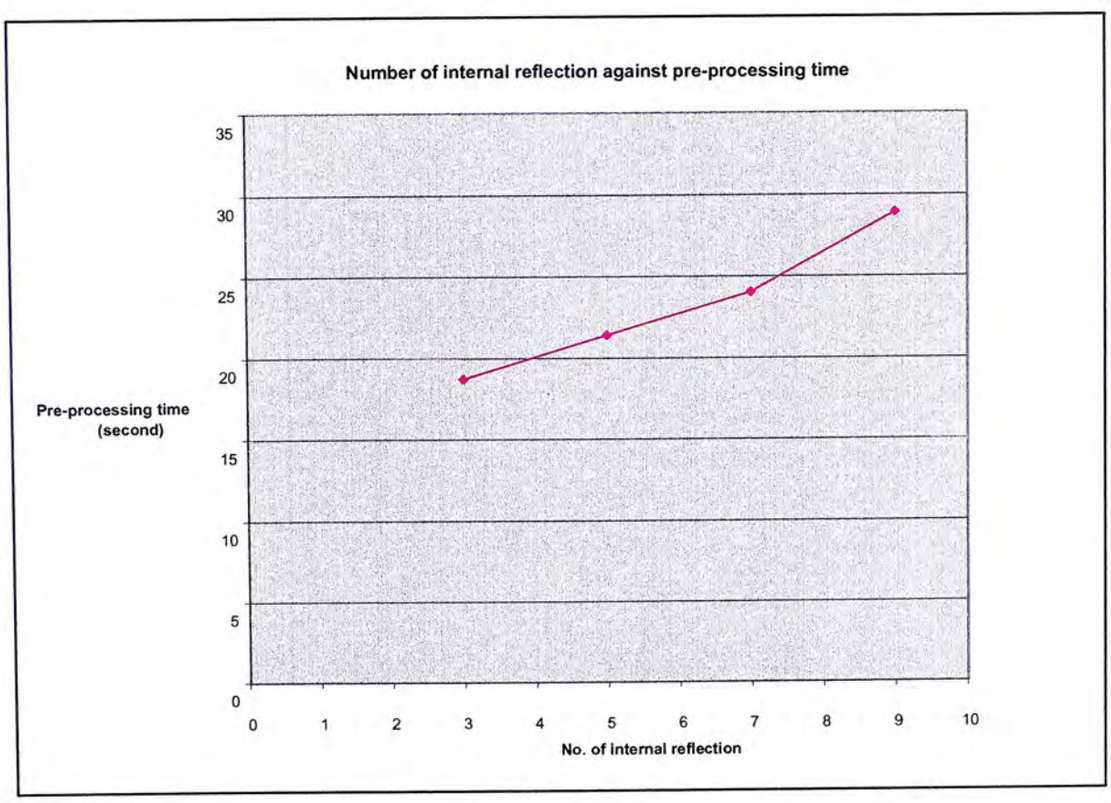

Figure 8.14 Numbers of counts against pre-processing time

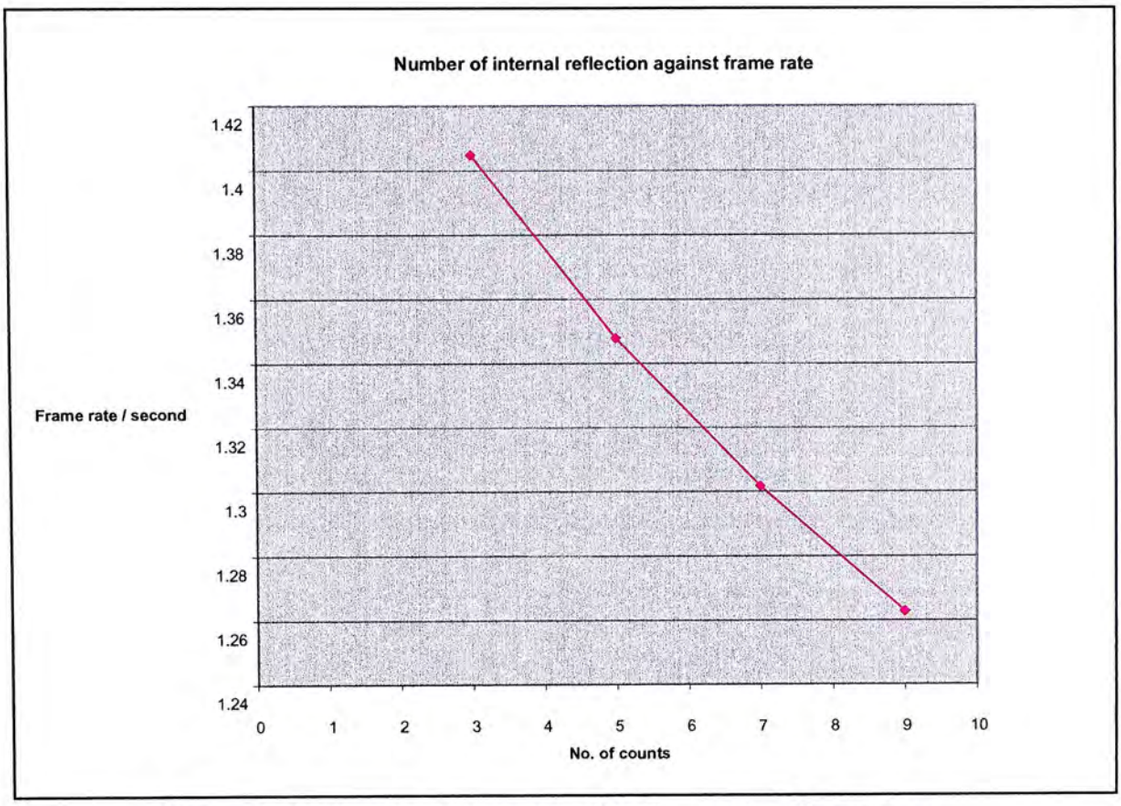

Figure 8.15 Number of internal reflection against frame rate

In conclusion, the above experimental results match quite well with the analytical results.

8.3 Comparison

In this section, the proposed technique is compared with different rendering methods in terms of rendering speed and image quality. Parameters of these rendering methods are set similar to each other. These parameters include refractive index, lighting, background, gemstone cutting and orientation of gemstone model.

8.3.1 Comparing with real images of gemstone

In the figures below, images on the left are the images of real gemstone, while images on the right are the images generated by the system. The lighting conditions of the images are not exactly the same as it is difficult to imitate the lightings and some other effects of a real environment. Figure 8.16a shows the images of a garnet. Figure 8.16b shows the images of a sapphire. Figure 8.16c shows the images of a tourmaline. Figure 8.16d shows the images of a spinel.

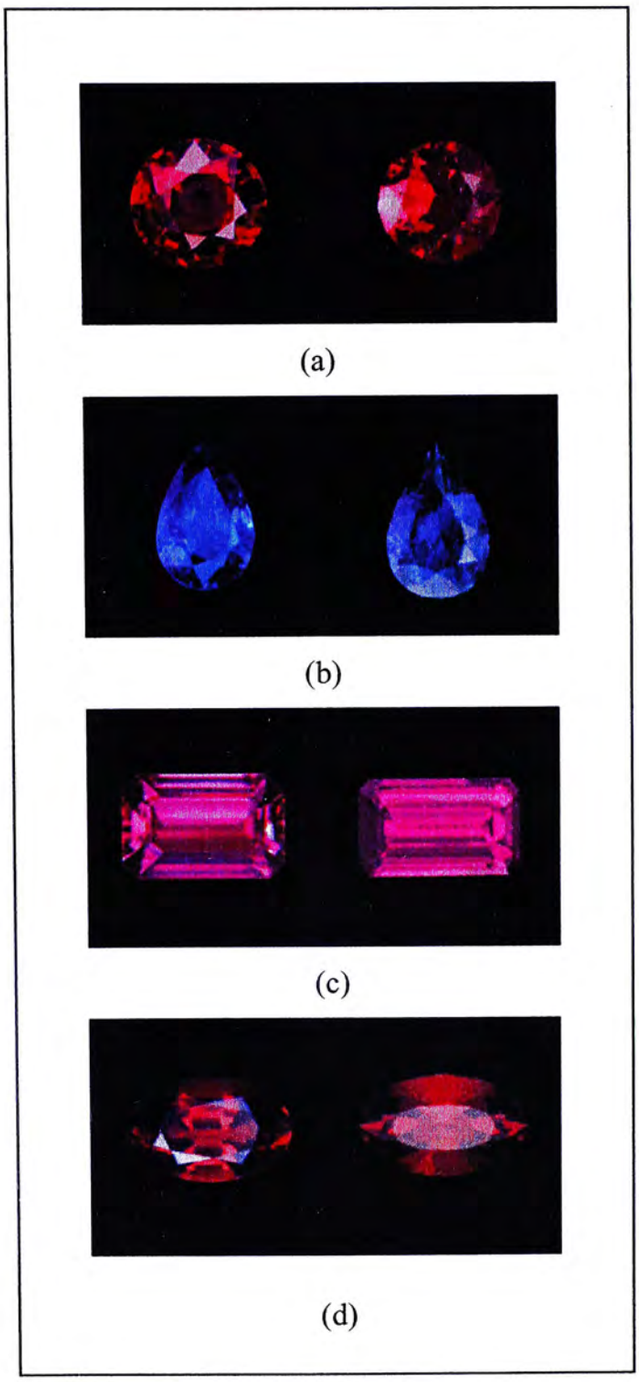

Figure 8.16 Comparison with the real images of gemstones

8.3.2 Comparing with images created by other renderers

For comparison purpose, parameters of different rendering software are set similar to each other. These parameters include the refractive index, lighting, background, gemstone cutting and the orientation of the gemstone model.

8.3.2.1 Phong shading model

Phong shading model is a standard rendering technique for real-time application (discussed in Chapter 2). It relies on the use of an alpha value to handle the rendering of refractive object. In Figure 8.17, images generated by the system are shown on the left, while images generated by Phong shading are shown on the right.

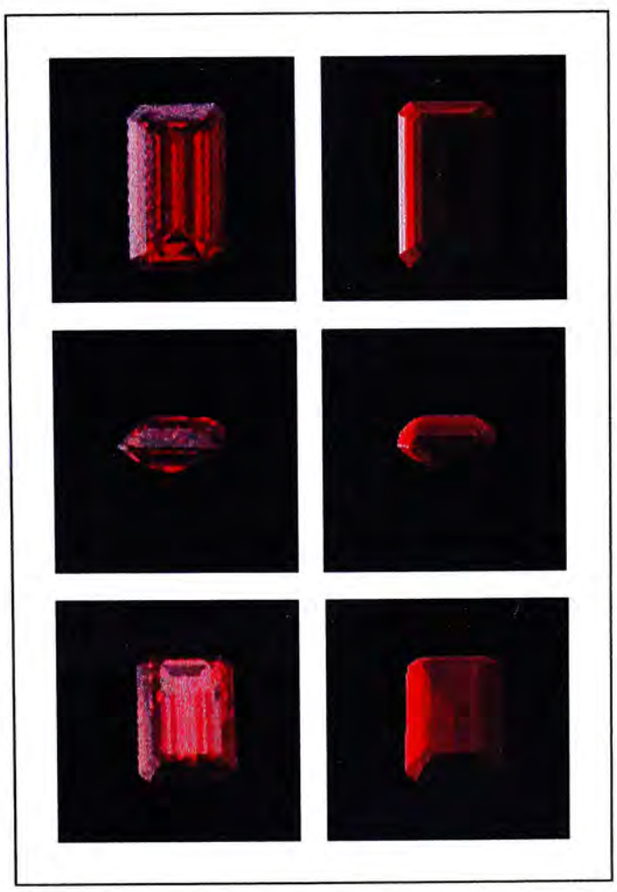

Figure 8.17 Comparing with the images generated by Phong shading

8.3.2.2 Design software

Flamingo is a ray tracing renderer used with Rhinoceros 3D design software which supports refraction rendering. In Figure 8.18, images generated by the proposed system are shown on the left, while images generated by ***Flamingo*** are shown on the right. Figure 8.19 shows a gemstone image generated by ***TechJewel. TechJewel*** is a rendering software specific for jewelry design, which is based on the ***Flamingo***.

In terms of rendering time, the proposed system takes about 0.6 second for generating an image in the shading stage, while Flamingo requires about 3 second for generating an image. The proposed technique not only enhances the rendering speed by up to 4 – 5 times faster comparing with Flamingo, but also improves the image quality.

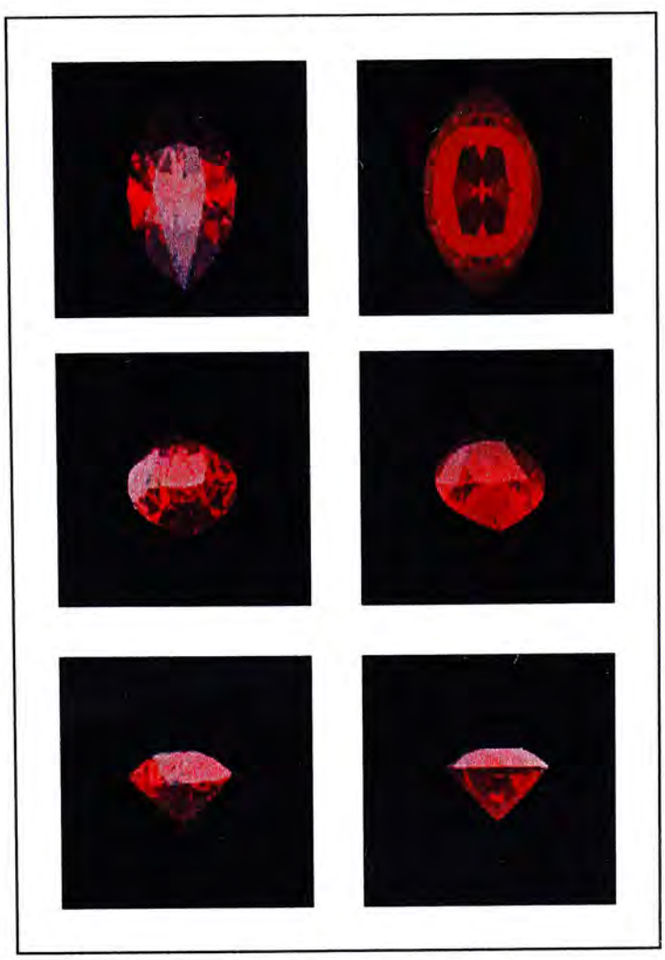

Figure 8.18 Comparing with the images generated by ***Flamingo***

Figure 8.19 Image generated by ***TechJewel*** [Flamingo02]

In Figure 8.20, an image generated by the proposed system is shown on the left, while an image generated by ***trueSpace*** are shown on the right. ***trueSpace*** is a 3D design software widely used by jewelry designers.

Figure 8.20 Comparison with the image generated *trueSpace*[3DJeweler03]

8.3.2.3 Web application software

Cult3D is a software which allows interactive viewing of objects on the Web. Figure 8.21 shows a comparison between a diamond ring produced by Cult3D and a ring

rendered using the proposed system. In Figure 8.22, the green and red patterns are used to imitate the dispersion effect. However, when there is a change in viewing direction, the patterns are retain the same. So, there is a loss in realism, especially when the ring is animated.

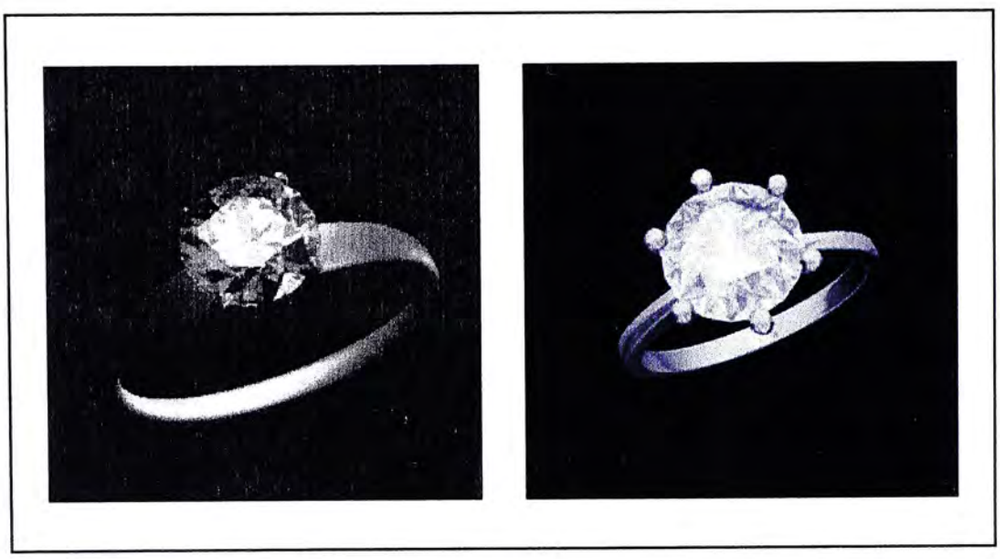

Figure 8.21 Comparison with the images generated by ***Cult3D***

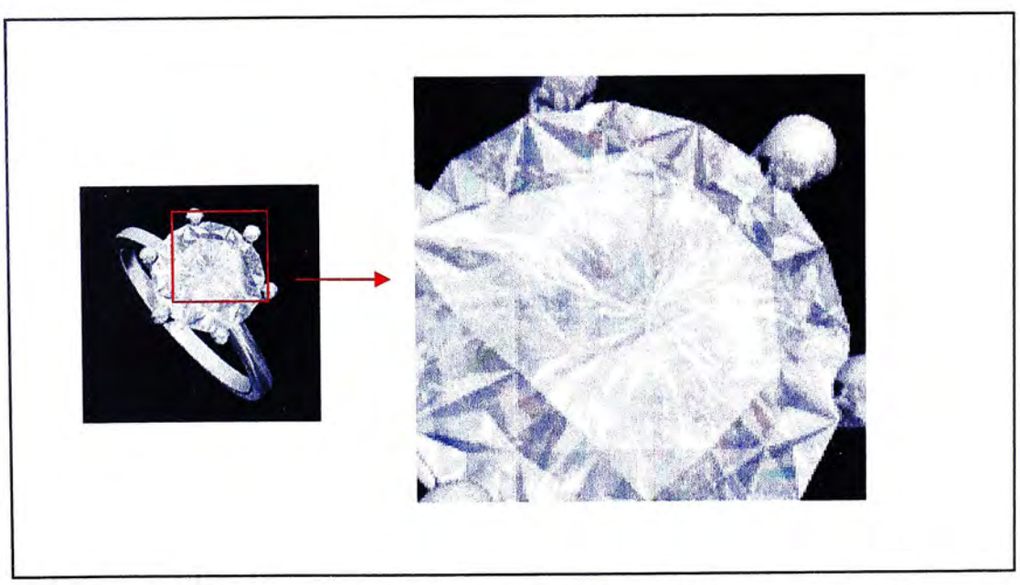

Figure 8.22 A diamond ring generated by Cult3D [Wikipedia03]

9 Conclusion and Future works

9.1 Conclusion

In this thesis, we have presented an interactive refractive rendering technique is proposed for rendering gemstone. To the best of our knowledge, this is the first refractive rendering method for gemstone which provides interactivity.

The technique relies on the construction of a ray-table using the ray tracing algorithm. Photorealistic image of gemstone with various cutting shape, gemstone properties, lighting and background can be rendered. In order to achieve interactivity, a pre-computation process is adopted for storing information for rendering a gemstone model with pre-defined orientation in the form of ray-list. Although real-time animation cannot be achieved at the moment, the rendering speed is significantly improved comparing with ordinary ray-tracing technique. In terms of image quality, the proposed technique system showed good performance on the rendering of colored gemstones. Gemstone images rendered using our system, as shown in chapter 8, are comparable to real images, while interactivity is achieved.

To further extend the use of this technique, the rendering algorithm is merged with existing techniques for displaying virtual 3D environment as described in chapter 7. The effect of a change in the 3D environment on the gemstone image can be seen immediately. The technique is particularly useful in the interactive visualization of jewelry designs.

One limitation of the proposed technique is the large amount of data needed to be stored. Ray-list data has to be stored up for each position and orientation relative to the view point. A large amount of ray-tables have to be stored to obtain smooth animation at different views.

9.2 Future work

In the system, the direction of viewpoint $\vec{v}$ is always perpendicular to the image plane which is shown in Figure 9.1, that is, a ray-table is associated with a given view plane and viewing position. Therefore, when there is a change in viewpoint, the corresponding ray-table matching the new viewpoint has to be retrieved.

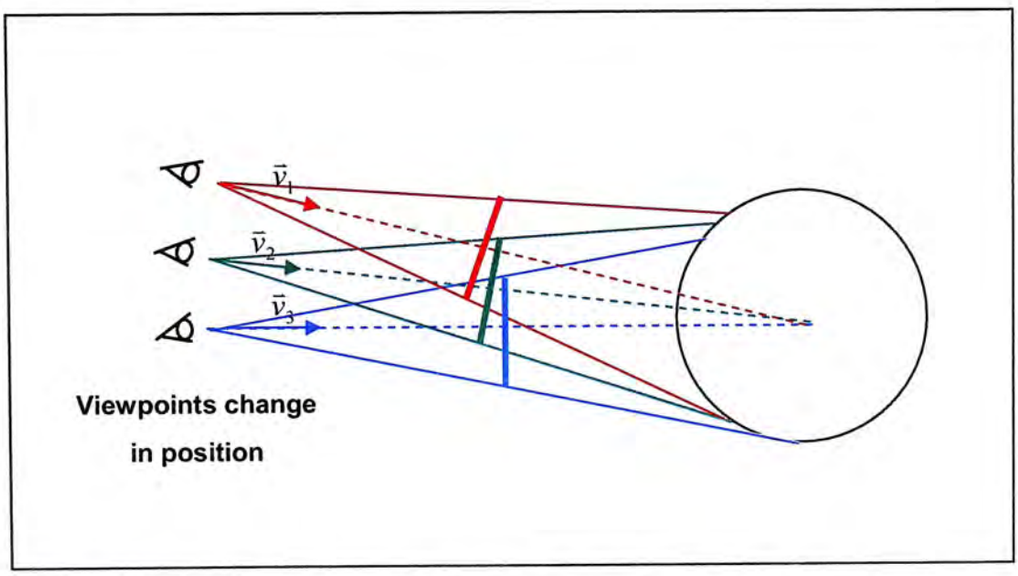

Figure 9.1 Viewpoint's direction is always perpendicular to the image plane

To reduce the size of the database, a ray-list is associated with a viewing direction. A set of ray-list associated with a pixel on an imaginary image plane eliminates the requirement of storing ray-tables for different viewing distance. In Figure 9.2, although there is a change in viewpoint positions, the position and orientation of the image plane is fixed.

9 Conclusion and Future work

9.1 Conclusion

In this thesis, we have presented an interactive refractive rendering technique is proposed for rendering gemstone. To the best of our knowledge, this is the first refractive rendering method for gemstone which provides interactivity.

The technique relies on the construction of a ray-table using the ray tracing algorithm. Photorealistic image of gemstone with various cutting shape, gemstone properties, lighting and background can be rendered. In order to achieve interactivity, a pre-computation process is adopted for storing information for rendering a gemstone model with pre-defined orientation in the form of ray-list. Although real-time animation cannot be achieved at the moment, the rendering speed is significantly improved comparing with ordinary ray-tracing technique. In terms of image quality, the proposed technique system showed good performance on the rendering of colored gemstones. Gemstone images rendered using our system, as shown in chapter 8, are comparable to real images, while interactivity is achieved.

To further extend the use of this technique, the rendering algorithm is merged with existing techniques for displaying virtual 3D environment as described in chapter 7. The effect of a change in the 3D environment on the gemstone image can be seen immediately. The technique is particularly useful in the interactive visualization of jewelry designs.

One limitation of the proposed technique is the large amount of data needed to be stored. Ray-list data has to be stored up for each position and orientation relative to the view point. A large amount of ray-tables have to be stored to obtain smooth animation at different views.

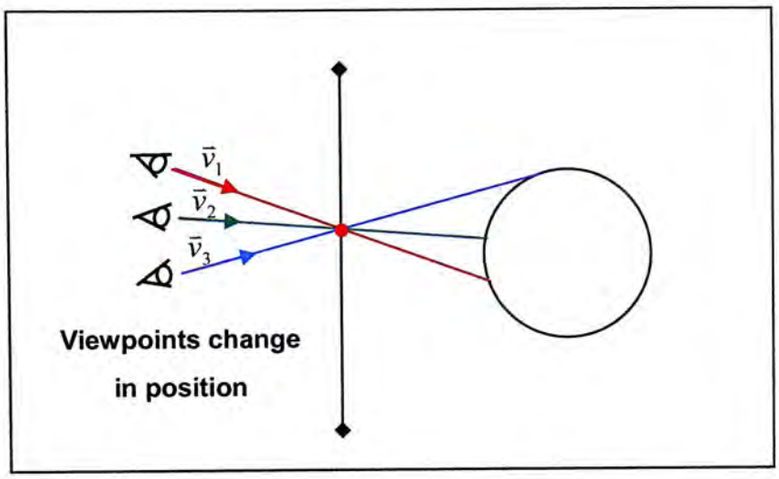

Figure 9.2 Viewing directions are independent of the image plane

When there is a change in viewpoint position, ray-list data corresponding to the direction of required pixel ray and the position of the pixel $I_{(i,j)}$ in the image plane is retrieved. Figure 9.3 shows the relationship among viewpoint positions, the directions of pixel rays and the pixel $I_{(i,j)}$ in the image plane. In this case, whenever there is a change in viewpoint position, e. g. from A to B as shown in Figure 9.4, the required ray-list can be retrieved from the revised ray-table. This eliminates the need for storing different ray-table for different viewing distance. Therefore, the size of the database can be reduced.

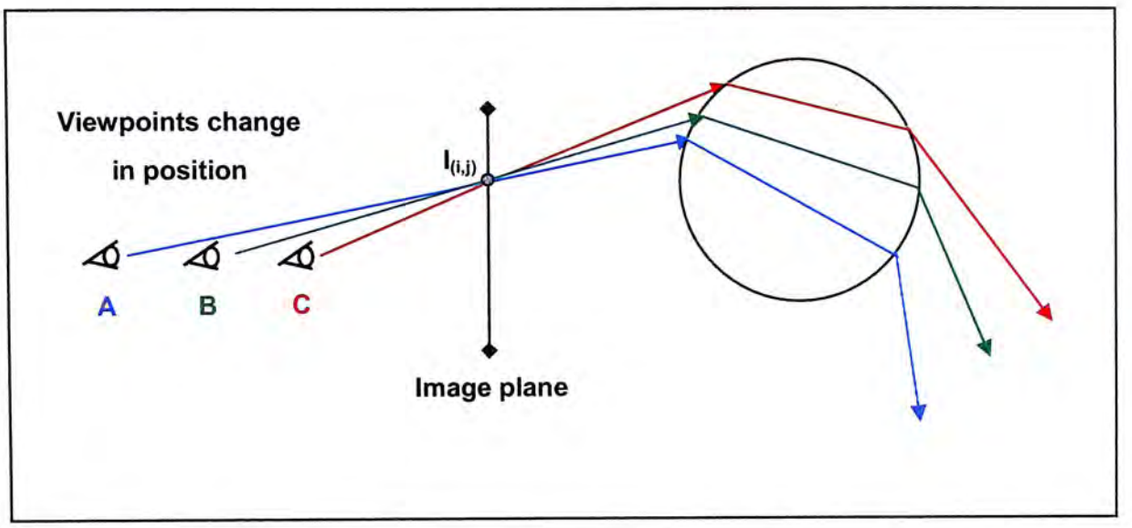

Figure 9.3 The relationship among viewpoint positions, pixel ray directions and the pixel location $I_{(i,j)}$ in the image plane

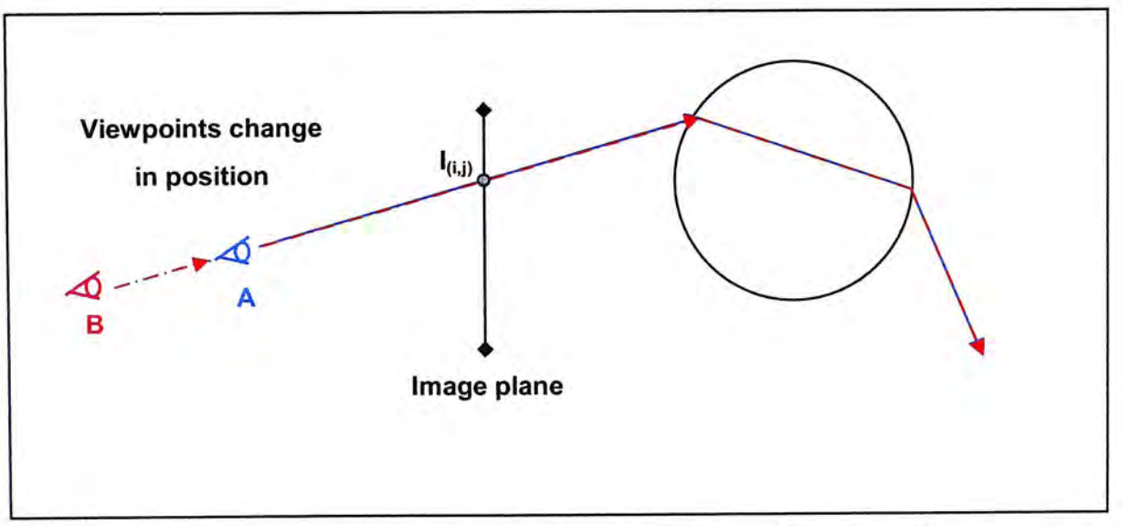

Figure 9.4 The position of viewpoint change from A to B

Besides, by using the rendered image of a gemstone as the background texture for another rendering process, it is possible to extend the technique for rendering multiple gemstones which is essential for the jewelry design. In addition, by taking into consideration the effects of dispersion, the technique can be expected to generate a more realistic gemstone images with interactive responds

Reference

[Adelson91] E. H. Adelson and J. R. Bergen, "The Plenoptic Function and Elements of Early Vision ". *Computational Models of Visual Processing*, pages 3-20,1991.

[AHO87] A. Aho, J. E. Hopcroft and J. D. Ullman, " Data Structures and Alogrithms", Addison-Wesley, 1987.

[Arvo86] J. Arvo, Backward ray tracing. *Siggraph '86 Developments in Ray Tracing seminar notes*, Aug. 1986.

[Ashdown94] I. Ashdown, "Radiosity – A programmer Perspective", John Wiley & Sons, Inc., 1994.

[Canon97] http://www.x-zone.canon.co.jp/CyberMirage/index2-e.html

[Clark97] C. Clark, "Tropical Gemstones or Malaysia & Southeast Asia". Periplus Editions, 1997.

[Collins94] S. Collins, "Rendering Crystal Glass". *Proceedings of the 2nd Irish Workshop on Graphics*, Coleraine, Nov 1994.

[Diefenbach94] P. J. Diefenbach and N. I. Badler. "Pipeline Rendering: Interactive Refractions, Reflections, and Shadows". *Display*

(Special Issue on Interactive Computer Graphics), Vol. 15, No. 3, pages 173-180, 1994.

[Euyar03] http://www.euyar.com/customize/3D.asp

[Flamingo02] http://www.flamingo3d.com

[GIA03] http://www.virtualcampus.gia.edu/giavc/didictio.cfm

[Glassner89] A. S. Glassner, "An introduction to ray tracing" Academic Press, London, 1989.

[Gortler96] S. H. Gortler, R. Grzexzczuk, R. Szelinski, and M. F. Cohen, "The Lumigraph". *In Computer Graphics, SIGGRAPH 96 Proceedings*, pages 43-54, August 1996.

[Gouraud71] H. Gouraud. "Continuous Shading of Curbed Surfaces". *IEEE Transactions on Computers*, C-20(6):623, June 1971.

[Hakura01] Z. Hakura and J.Snyder. "Realistic Reflections and Refractions on Graphics Hardware with Hybrid Rendering and Layered Environment Maps". *In Proceedings of the 12th Eurographics Workshop on Rendering*. London, UK, June 2001.

[Harlow98] G. E. Harlow, "The nature of diamonds". Cambridge University Press in association with the American Museum of Natural History, Cambridge, U.K; New York, NY, USA, 1998.

[Hecht87] E. Hecht, "Optics", Second Edition, Addison-Wesley, 1987.

[Heidrich99] W. Heidrich H. Lensch, M. Cohen, and H. Seidel. "Light Field Techniques for Reflections and Refractions". *In Rendering Techniques '99*, pp187-196. Springer, Wien, Granada, Spain, June 1999.

[Kay79] D. S. Kay and D. P. Greenberg, "Transparency for Computer Synthesized Images". *In Computer Graphics SIGGRAPH 79 Proceedings*, vol. 13, pages 158-164, August 1979.

[Lensch02] H. Lensch, M.Goesele, P. Bekaert, J. Kautz, M. Magnor, J. Lang, H.-P. Seidel , "Interactive Rendering of Translucent Objects". *Proceedings of Pacific Graphics '02*, pages 214-224, October 2002.

[Levoy96] M. Levoy and P. Hanrahan, "Light field rendering". *In Computer Graphics, SIGGRAPH 96 Proceedings*, pages 31-42, August 1996.

[Matusik02] W. Matusik, H. Pfister, A. Ngan, R. Ziegler and L. McMillan, "Acquisition and Rendering of Transparent and Refractive Objects". *Eurographics Workshop on Rendering (EGRW)*, pages 267-278, June 2002.

[Ohbuchi03] E. Ohbuchi, "A real-time refraction renderer for volume objects using a polygon rendering scheme". *2003 Computer Graphics International*, pages 190, July 2003.

[Phong75] B.T. Phong. "Illumination for Computer Generated Pictures". *Communications of the ACM*, Vol.18, No.6, pages 311-317, June 1975.

[Sun99] Y. Sun, F. D. Fracchia, and M. S. Drew, "Rendering the phenomena of Volume Absorption in Homogenous Transparent Material". *the 2nd Annual IASTED Interational Conference on Computer Graphics and Imaging (CGIM'99)*, pages 283-288, Oct, 1999.

[Sun00a] Yinlong Sun, F. David Fracchia, and Mark S. Drew, "Rendering Diamonds". *Western Computer Graphics Symposium 2000*, March, 2000.

[Sun00b] Yinlong Sun, F. David Fracchia, and Mark S. Drew, "Rendering Light Dispersion with a Composite Spectral Model". *First International Conference on Color in Graphics*

and Image Processing, CGIP'2000, pages 51-56, Saint-Etienne, France, October 1-4, 2000.

[Tannous03] http://www.gia.org/research/156/gia_on_diamond_cut.cfm

[TS' O87] P. Y. TS' O and B. Barsky, "Modeling and Rendering Waves: Wave Tracing Using Beta-Splines and Refractive and Refractive Texture Mapping". *ACM Transaction on Graphics*, Vol. 6, No. 3, pages 191-214,1987.

[Yuan88] Y. Yuan, T. L. Kunii, N. Inamoto, and L. Sun, "Gemstone Fire: Adaptive Dispersive Ray Tracing of Polyhedrons". *Visual Computer*, Vol. 4, No. 5, pages 259-270, Nov.1988.

[Webster75] R. Webster, "Gems their sources, descriptions and identification". Newnes- Butterworths Ltd, London, 1975.

[Wikipedia03] http://en.wikipedia.org/wiki/Fresnel_equations

[Wilt94] N. Wilt. "Object- oriented ray tracing in C++". Wiley, New York,1994.

[Zongker99] D. Zongker, D. Werner, B. Curless, and D. Salesin, "Environment Matting and Compositing." *In Computer*

Graphics, SIGGRAPH 99 Proceedings, pages 205-214, August 1999.

[3DJeweler03] http://3djeweler.com/3Dwelcom.htm.