IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727
PP 52-60
www.iosrjournals.org

Minimalist Ray Tracing

Dr. Kirti Wankhede[1], Karan B Pathak[2]
[1]*(Assistant Professor, KJ Somaiya Institute of Management Studies and Research, India)*
[2]*(MCA Student, KJ Somaiya Institute of Management Studies and Research, India)*

Abstract: *Ray tracing is a physics-based method for simulating photorealistic 3D scenes. Light rays are drawn from the eye through each pixel of the desired image, and the rays' interactions with the scene determine the displayed pixel color. In computer graphics, ray tracing is a rendering technique for generating realistic image on view plane from 3D object by shooting rays at every pixel. The technique is capable of producing all properties of light such as shadows, reflection and refraction which are difficult to imitate in scanline rendering. Ray tracing technique is very useful in the scenario where the display or move of the image require to be slow for example creating visual effects in film or television. But it is not suitable in the scenario where display speed is high for example video games etc.. The current paper gives the overview of ray tracing technique and its implementation by using JavaScript.*

Keywords: *depth of field, reflection, refraction, sampling, transparency.*

I. Introduction

Ray Tracing is able to simulate optical effect of light such as reflection, refraction and shadow due to which image appear more real. Ray Tracing simulates the similar behavior to sun rays in nature. In nature when sun rays hits the object on the surface three things happen absorption, reflection and refraction due to which some object reflects like shiny metal while some are transparent like glass and some objects completely absorbs light like wood, stone etc. These three properties are natural result in ray tracing while it may be much difficult in scanline rendering. But all this significant properties comes with high computation cost.

Ray Tracing is been used in movies. Pixar Animation studio with great movies like Toy Story, Monster inc, Cars has been using ray tracing for realistic effects. In fact Movie “Car” introduce ray tracing in cinematic [1].

The purpose of this paper is to demonstrate how minimal ray tracer can be setup compared to huge ray tracer used in game industry / Film.

II. Antialiasing

Computers are discrete devices that display a finite number of pixels. As such most Ray Traced images are subject to aliasing, where an alias means substitute. An Aliasing effect produces staircase jaggies as shown in Fig 1.

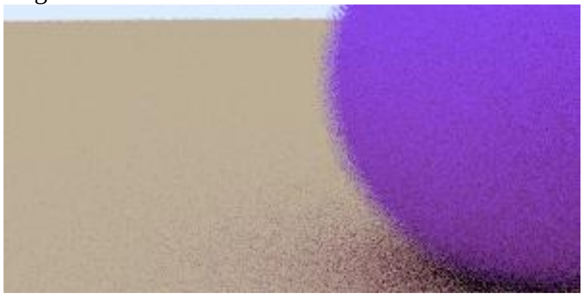


Fig 1. aliasing effect

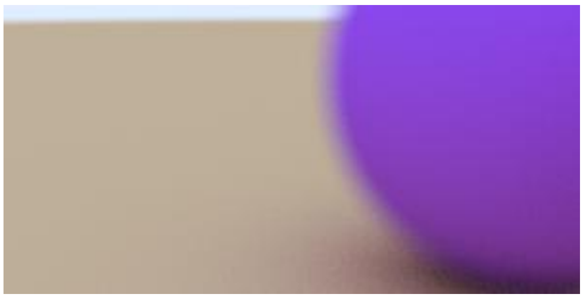


Fig 2. anti-aliasing

Remedies to aliasing: In ray tracing to resolve aliasing effect increase the sampling rate. It involves shooting more rays. Which smooth the corner edges of objects as shown in below Fig 2.

2.1 Regular Sampling

In Regular Sampling we shoot extra rays on each pixel and average the color of every pixel as shown in Fig 3a, Fig 3b is sampled with 25 rays. Here, 16 samples are yellow, and 9 are grays resulting in a mixture of color of (16/25) x (1, 1, 0) + (9/25) x (0.5, 0.5, 0.5) = (0.82, 0.82, 0.18) as in Fig 3c.

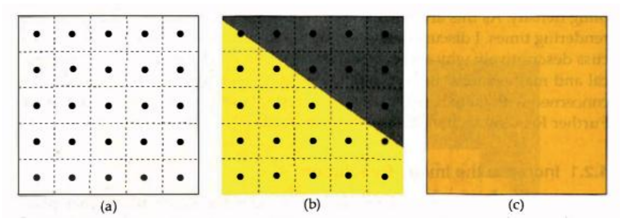


Fig 3. regular sampling [2]

Problem with Regular Sampling,

a) Aliasing is still present
b) Due to which Moire pattern appears.

2.2 Random Sampling

In random Sampling aliasing is replaced by noise, if we shoot rays that are randomly distributed over pixel surface. Noise is just a random pixel colors.

2.3 Jittered Sampling

Random Sampling is not the best way to uniformly distribute rays over surface of pixel of the following:

a) Random sample can clump together,
b) Random sample can leave gaps as shown in Fig 4a and Fig 4b.

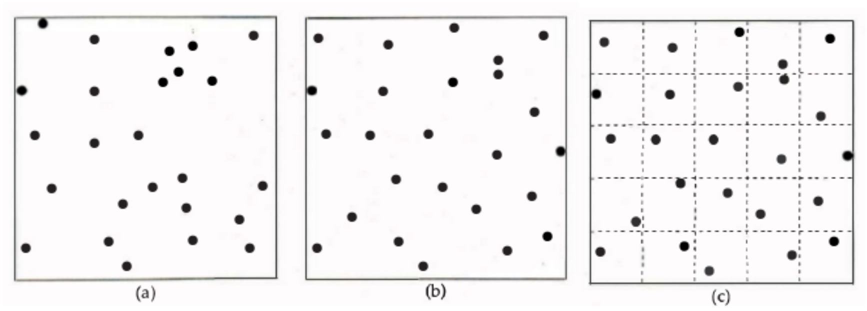


Fig 4. Jittered Sampling [2]

Uniform distribution of samples over the pixel is better way while still maintaining the randomness. To achieve Jittered Sampling:

a) Divide pixel into an n x n grid cell.
b) Select samples of n^2.
c) Randomly place single samples in each cell.

III. Sampling

Sampling are used to obtain flat edges in other words if we want to remove sharp-edges or jig-jag edges, we will have to do lot of more sampling as compared to antialiasing. Sample pattern which have below features are good Sampling: a) Points are uniformly distributed over the 2D unit square, so that clumping and gaps are minimized

b) Uneven distributions can result in parts of scenes being under sampled or oversampled, and this can increase aliasing c) Uniform distance between each sample points. Types of Sampling Pattern

3.1 n-Rooks Sampling

In this technique we place n samples in an n x n grid, so that there’s exactly one sample in each row and column and randomly shuffle them in x and y coordinate while maintaining n-rooks condition. Difference between jittered and n-Rooks

a) jittered Sampling we have n x n grid with n^2 samples
b) n-Rooks sampling we have n samples in n x n grid.

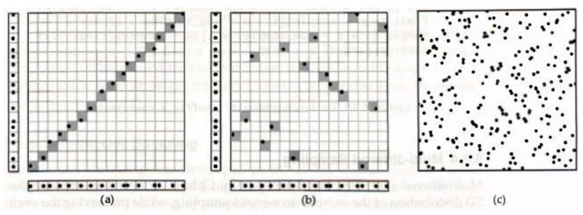


Fig 5. N-Rooks Sampling [2]

Problem with n-Rook Sampling is 1D projection has uniformly distributed samples, but the 2D projection is no better than random sampling.

3.2 Multi-Jittered Sampling

It improves 2D distribution of samples in n-Rooks sampling, while preserving the uniform distribution in 1D projection. This technique is combination of n-rooks and jittered sampling.

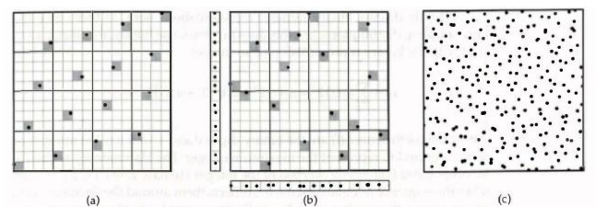


Fig 6. Multi-Jittered Sampling [2]

Fig 6 contains 16 x 16 grids which is superimposed by 4 x 4 grid. We randomly generate a single sample in each 4 x 4 grid cells. This is jittered distribution with respect to 4 x 4 grids, but also satisfies the n-Rooks condition on sub-grid. Therefore multi-jittering uses benefits of both n-Rook and jittered sampling, n-Rooks sampling gives good 1D projections, while 4 x 4 jittering results in a better 2D distribution.

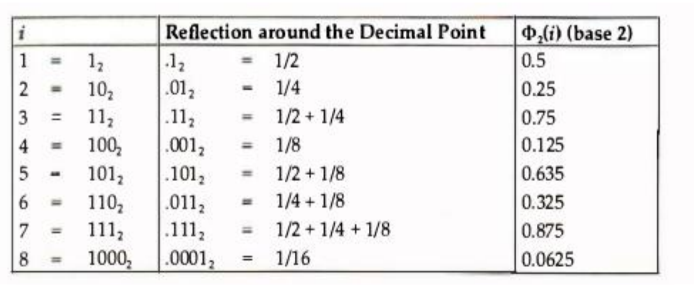
3.3 Hammersley Sampling

It is based on computer representation of number in various prime number bases. For an n x n unit square, the Hammersley samples Pi (Radical Inverse formula)

:$P_i = (x_i, y_i) = [1 \div n, \phi_2(i)]$(1)

$\phi_2(i) = \sum_{j=0}^{n} a_j(i) 2^{-j-1}$(2) [2]

Table 1. Radical Inverse function [2]



i			Reflection around the Decimal Point			$\Phi_2(i)$ (base 2)
1	=	1_2	$.1_2$	=	1/2	0.5
2	=	10_2	$.01_2$	=	1/4	0.25
3	=	11_2	$.11_2$	=	1/2 + 1/4	0.75
4	=	100_2	$.001_2$	=	1/8	0.125
5	=	101_2	$.101_2$	=	1/2 + 1/8	0.635
6	=	110_2	$.011_2$	=	1/4 + 1/8	0.325
7	=	111_2	$.111_2$	=	1/2 + 1/4 + 1/8	0.875
8	=	1000_2	$.0001_2$	=	1/16	0.0625

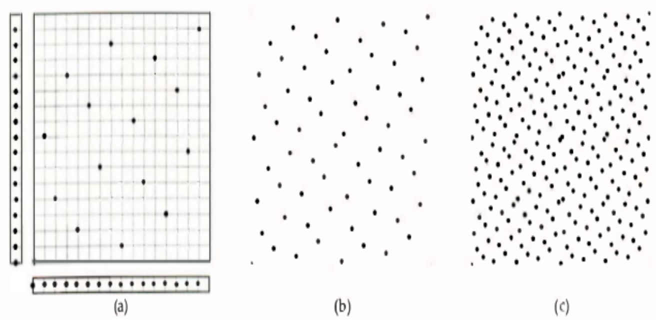


Fig 7. Hammersley Sampling [2]

Fig 7. shows hammersley sequences of 16, 64 and 256 samples. Since all samples are in unit square. Therefore 1D projections are regularly spaced. This can lead to aliasing. The advantage is sample points are well-distributed in 2D, with a minimum distance between the samples. That's something none of sampling pattern has.

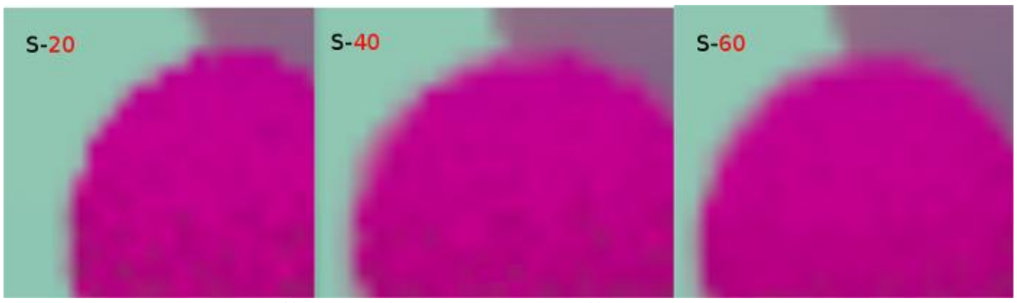


Fig 8. show difference after Sampling at different rate: S-20, S-40, S-60

IV. Virtual Pinhole Camera

It is just a box with small hole on one side and film on the other side. Images produced by pinhole cameras are much simpler and less expensive. The pinhole camera known as virtual pinhole camera has significance in 3D application and video games.

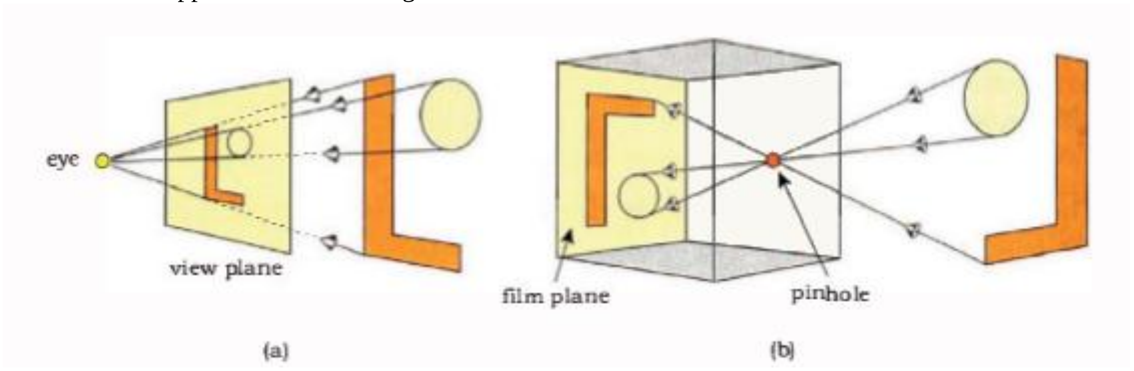


Fig 9. Virtual Pinhole Camera [2]

As shown in Fig 9a(virtual pinhole camera) and Fig 9b(real pinhole camera) there is important difference between real pinhole camera and virtual pinhole camera. In virtual pinhole camera view plane is between eye and object and in real pinhole camera, pinhole is between the objects and film.

The virtual pinhole camera implements:

a) The eye point e.
b) The look-at point l.
c) The up vector up (y_v).
d) The view plane distance d.

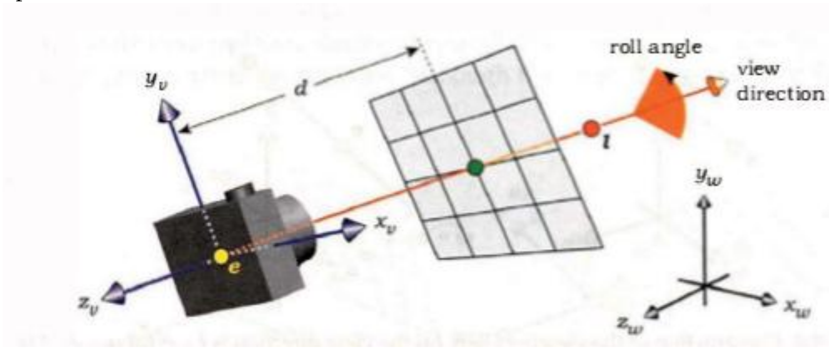


Fig 10. Component of Virtual Pinhole Camera [2]

The view direction is defined as l-e (look at point - eye point).To orient camera and view plane about z-axis we use up vector (y_v).viewing coordinate is based on orthonormal coordinates (u, v, w) and is also known as camera coordinate , vector w is calculated by normalizing view direction in opposite direction,

: $w = (e-l) / \| e-l \|$ (3)
Up vector takes default unit vector, so $v = (0,1,0)$ and
: $u = up \otimes w$ (4)

4.1 Depth of Field

The depth of field of a camera is the range of distance the object appears to be in focus.

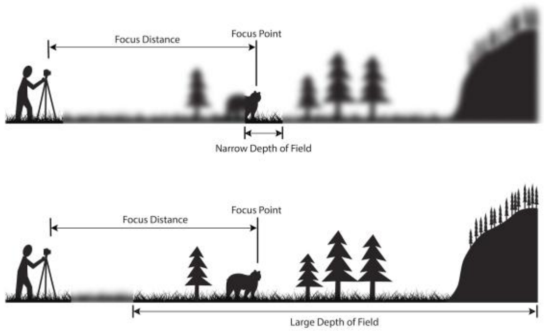


Fig 11. Depth of Field [4]

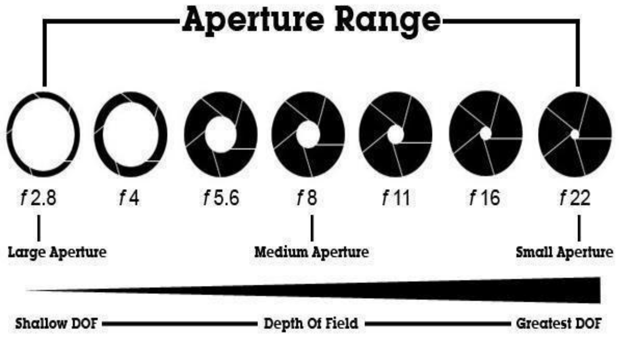


Fig 12. Aperture range in camera [4]

Factors that affect Depth of Field
Aperture - is a small opening in camera through which lights travel inside. Aperture is also known as f /stop. The smaller the f /stop the larger the aperture opening.
Focal Length - is the distance between the camera and lens. The longer the focal length the more shallow will be depth of field and vice versa.
: f-stop = Focal Length ÷ Aperture Diameter (5)

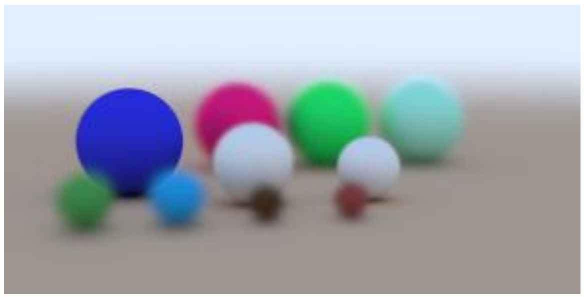


Fig 13. Shallow Depth of field (Images not in focus are more blurred in shallow then deep depth of field)

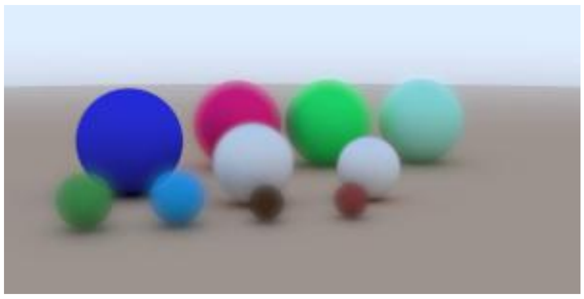


Fig 14. Deep Depth of field (Foreground and Background appears more focus and clear then shallow depth of field.)

4.2 Thin Lens Theory

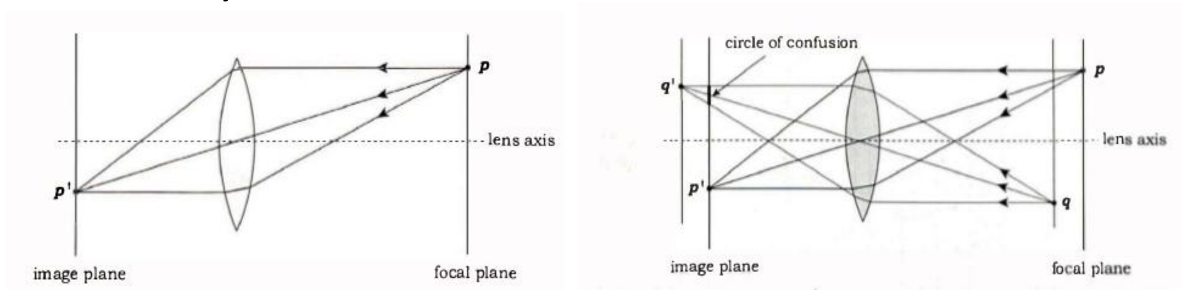


Fig 15. Thin lens theory [3]

Thin lens will focus all points from focal plane to image plane. Every point p on the focal plane has a corresponding image p|. So if the lens was real camera and film were the image plane, the focus plane would be in focus, but everything else would be out of focus.
If lens was real camera lens then more than one ray can travel via lens and there can exist more than one matched pair of focal and image plane and shown in Fig 15. The rays which hits image plane and make small circular on image plane is known as circle of confusion. The q would be out of focus if we place film on it. The further q goes from plane of p (on the either side), the larger circle of confusion becomes and more out of focus q becomes.

V. Ray-Geometry Intersection

Ray-Geometry intersection method is used to display 3d objects on pixel screen. Shapes are of different types sphere, disk, cubes or boxes these shapes are mathematically described by equation and we use this equation to compute whether they are intersected by rays.

5.1 Ray-Sphere intersection

Ray can be expressed as
: O + tD (6)

Where O is a origin of ray, D is direction vector and t is parameter of function. By varying t we can compute any point on line.if t > 0 then point on ray is in front, if t < 0 point on ray is behind the ray origin which is not needed. When t ==0 the point and ray origin are same.

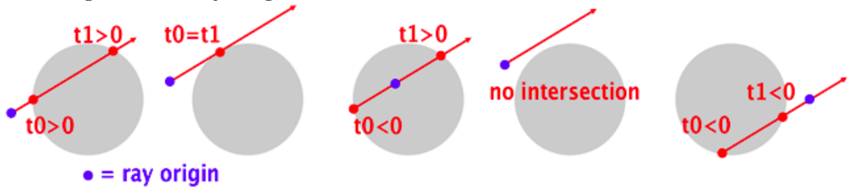


Fig 16. Sphere Intersection [5]

Sphere can described as
: $x^2 + y^2 + z^2 = R^2$ (7)
If we assume x, y, z as coordinate of point P then
: $P^2 - R^2 = 0$ (8)
Substitute (7) in (8) we get
: $|O + tD|^2 - R^2 = 0$(9)
on expanding this (9) we get
: $O^2 + (Dt)^2 + 2ODt - R^2$ (10)
: $f(x) = ax^2 + bx + c$(11)
Where $a = D^2, b = 2OD, c = O^2 - R^2$. Since (11) is quadratic function its root can be easily found using
: $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ (12)
: $\Delta = b^2 - 4ac$ (13)

The letter Δ (Greek letter delta) is called the discriminant. When $\Delta > 0$ there is two roots(In that case, the ray intersects the sphere in two places (at t0 and t1)), when $\Delta = 0$ there is one root (The ray intersects the sphere in one place only (t0=t1) grazing point or tangent to sphere).and $\Delta<0$ there is no root which mean ray doesn't intersect sphere.

VI. Shading

It is the process to compute pixel color of object in 3D scene.

In computer graphics, a shading function is defined as a function which yields the intensity value of each point on the body of an object from the characteristics of the light source, the object, and the position of the observer [5]. Typically, light emitted by light source which falls on objects and is reflected towards viewer is known as direct lightning or direct illumination. Whereas a light emitted from light source is reflected by many surface before reaching to viewer is known as indirect illumination.

6.1 Diffuse Surface

In Diffuse Surface when light hits the surface at point P, not all energy is reflected some of the energy are absorbed and some of it is reflected. They reflect equal energy in all direction.

6.2 Lambert's Cosine Law:

The amount of light that a surface receives is directly proportional to the angle between the surface normal N and the light direction L[5]

: $\cos\theta = N.\,L$ (14)

Albedo (ρd): The albedo is the ratio of reflected light over the amount of incident light.

Therefore Amount of Reflected Light from Diffuse Surface(P)

: $\int_{\Omega} \rho_d * incidentLight * \cos\theta dw$ (15)

The integral symbol only means to sum up all the light energy across the whole surface of the hemisphere

VII. Shadows

Shadows are everywhere in the real world. Shadows are useful to determine how many light source are there, and from what direction their illumination come from. Without shadows image appears unnatural.

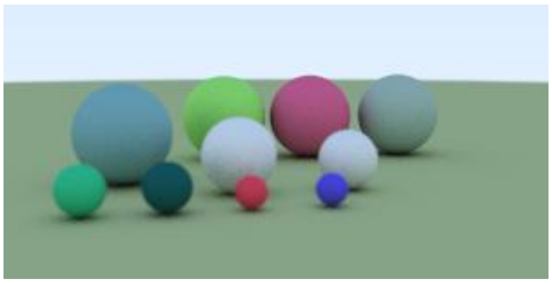


Fig 17. Shadows

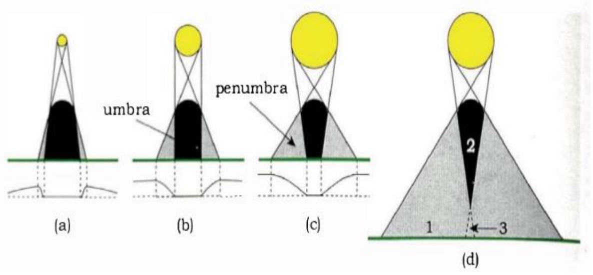


Fig 18. Shadows Rays (umbra, penumbra) [2]

Shadows are defined as the visible part of the scene that doesn't receive light from any light source because one or more objects block the illumination. The scene where none of light is visible is called umbra and where some of light is visible is called penumbra as shown in Fig 18.

The only way to evaluate shadow ray is to shoot ray from hit point towards a light source and see if the ray hits anything between point p and light source.

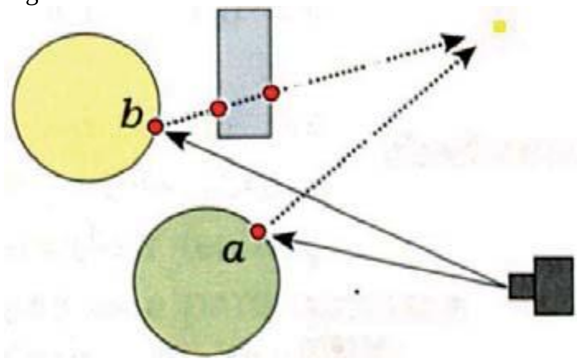


Fig 19. Working of Shadows Rays [2]

As shown in Fig 19 above Point a is shaded with ambient illumination + light source direct illumination because it shadow rays doesn't hits any object. In case of point b we shade it with only ambient illumination because its shadow ray is hit by an object which in between point b and light source.

VIII. Reflection and Refraction

Reflection and refraction are very common in the real world. Glass or water is two very similar materials which exhibit both properties. Light can travel through them and it can be reflected back[5].

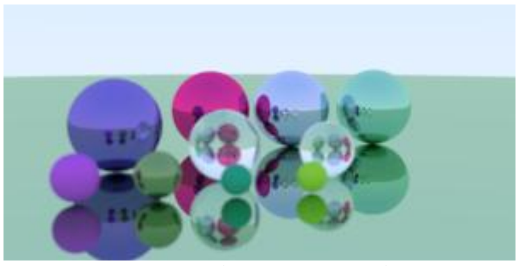


Fig 20. Reflection and Refraction

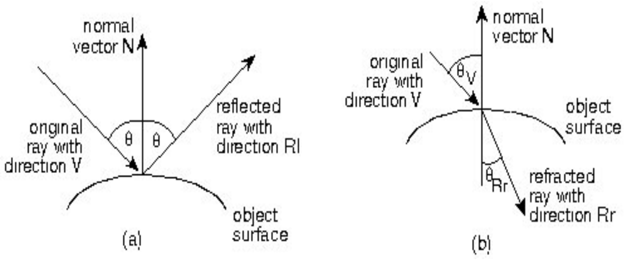

normal vector N
original ray with direction V
θ
θ
reflected ray with direction Rl
object surface
(a)
normal vector N
original ray with direction V
θV
object surface
θRr
refracted ray with direction Rr
(b)

Fig 21. Working of Reflection and Refraction [8]

To compute reflection rays we need to know ray direction when it was reflected by closest surface.(The incident direction), you need to know the location that the ray which was reflected by the surface (the surface location), and you need to know the normal of the surface at the intersection point.[6]
ReflectRayLocation = SurfaceLocation
ReflectRayDirection = IncidentDirection - 2 * SurfaceNormal * DotProduct(IncidentDirection, SurfaceNormal)
:$R = I - 2 (I \cdot N) \cdot N$ (16) [6]

When light travel from one medium to another medium they change direction. Every medium has refraction index (water = 1.3, glass=1.5, diamond=1.8).Refraction of rays depends on angle of incidence and refraction index.

Refraction indices for any medium is
:$\eta = \frac{c}{v}$ (17)

Refraction is defined by Snell's law
:$\frac{\sin\theta_1}{\sin\theta_2} = \frac{\eta_2}{\eta_1}$(18)

Snell's law is the relationship between the angles of incidence and refraction, when light travels through different media, such as water, glass, or air[7].The refracted ray direction R_r is given by n1 = index of refraction of original medium
n2 = index of refraction of new medium
η= n2 / n1
: $c_1 = \cos (\theta) = N \cdot I$ (19)
: $c_2 = \sqrt{1 - \eta^2 * (1 - c_1^2)}$(20)

If c_2 is greater then zero it's an total internal reflection wherein energy distributed to transmitted ray is zero.
If c_1 is greater it means incident ray has struck the water surface from inside or else its outside.
Therefore from (19) & (20) we get Transmitted rays as
$T = (\eta * I) + (\eta * c_1 - c_2) * N$ (21)

IX. Implementation of System

Demo example for Ray Tracing has been implemented by using JavaScript. Uploaded code on github and demo on website are available on following web link

- Demo: http://raytracing.surge.sh/
- Code on github: https://github.com/CY5/rayTracing

X. Conclusion

Even though Ray Tracing is not used for small devices due to it requires very high computation time to simulate light properties but it does create realistic visual which are not possible through scanline algorithm.
In fact Ray Tracing can be speed up if implemented parallel in GPU. For example Nvidia Ray Tracing engine does uses GPU to implement Ray Tracing which very fast compared to traditional ray tracing [9].

References

Journal Papers:

[1]. Per H. Christensen, Julian Fong, David M. Laur, Dana Batali , *Ray Tracing for the Movie 'Cars',*

Books:

[2]. Kevin Suffern , *ray tracing from the ground up* (Boca Raton Florida, CRC Press September 6, 2007).

Chapters in Books:

[3]. Antialiasing, Sampling Techniques, Ray-Object Intersections, Practical Viewing System, in Kevin Suffern (Ed.), *ray tracing from the ground up* (Boca Raton Florida, CRC Press September 6, 2007).

Websites:

[4]. Elizabeth (April 25, 2017) *Understanding Depth of Field – A Beginner's Guide* Retrieved October 29, 2017 https://photographylife.com/what-is-depth-of-field

[5]. Scratchapixel *Reflection, Refraction(Transmission) and Fresnel* Retrieved October 29, 2017 https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-shading/reflection-refraction-fresnel

[6]. Alan Wolfe (January 9, 2017) *Raytracing Reflection, Refraction, Fresnel, Total Internal Reflection, and Beer's Law* Retrieved October 29, 2017 https://blog.demofox.org/2017/01/09/raytracing-reflection-refraction-fresnel-total-internal-reflection-and-beers-law/

[7]. Wikipedia (6 November 2017) *Snells Law* Retrieved October 29,2017 https://en.wikipedia.org/wiki/Snell%27s_law

[8]. Paul Rademacher *Ray Tracing: Graphics for the Masses* Retrieved October 29,2017 http://www.cs.unc.edu/~rademach/xroads-RT/RTarticle.html

[9]. Nvidia *NVIDIA® OptiX™ Ray Tracing Engine* Retrieved October 29, 2017 https://developer.nvidia.com/optix