# 

# Comparing Whitted-Turner to Path tracing algorithm in the context of CPU raytracing

Minh Tri Tran

Graduation Work in the Programme Digital Arts and Entertainment, Game Development

Howest, University College West Flanders

Kortrijk, Belgium 2019

Abstract

Light transport algorithms are used to simulate the complex behavior of light in order to generate realistic images from a three-dimensional description of an artificial scene. Whitted Turner algorithm and Path tracing are two of those algorithms, used as an approximation of the rendering equation. However, these algorithms are significantly different in multiple aspects.

In this paper, the two algorithms are implemented and presented with a detailed description. Our contributions include a performance-oriented and rendering correctness comparison. Additionally, implementation details for each technique will be provided.

In the comparison, it is apparent that overall Whitted Turner algorithm performs faster relative to Path tracing. However, in regard the visual quality, Path tracing exceeds the former far more, since it can simulate many more physical phenomena.

Table of Contents

Chapter I. Introduction 4

Chapter II. Research 7

Radiometry 7

Rendering Equation 12

Whitted Turner Algorithm 13

Path tracing Algorithm 15

Research Question 16

Hypothesis 16

Chapter III. Case Study 18

Implementation 18

Performance Comparison 19

Samples Per Pixel 19

Resolution 23

Scenes 27

Visual Accuracy Comparison 29

Chapter IV. Conclusion 34

Further Work 34

References 36

Appendix 38

Lambertian BRDF derivation: 38

Complete render of the visual accuracy test 40

## Chapter I. Introduction

Rendering or image synthesis is the process of generating two-dimensional images from a collection of models, or often called a scene. Typically, a scene contains objects, camera, lightning and shading information as a description of a virtual environment. The data of a scene is given to a rendering software, which will process the input in order to generate an image output [1].

Rendering is an important topic in the field of computer graphics, due to its wide application. One of the uses that often comes in mind, are video games. The graphics of such interactive medium are achieved by generating the frames rapid enough that the viewer does not see the individual images, but rather becomes immersed [2]. Next up is, the movie industry that takes advantage of the rendering technology to achieve high graphics fidelity. These are typically the result of offline renderer that make use of state-of-the-art techniques. Rendering applications are not limited to the entertainment industry but are also used in fields such as scientific visualization, which the rendering technology proves to be an integral component. The technology enables visualization of complex scientific data on which scientists can get more insights from. On the other hand, computer-aided-design facilitates engineers to design and manufacture their product by generating detailed plans and drawings. Other fields that also make use of the rendering technology are for example in the medical sector, architectural visualization, product visualization and many more. [3] All in all, there are a lot of areas that makes use of the rendering technologies. With the increase of technology in our everyday life, the expanded uses of rendering are to be expected.

|  |  |  |
| --- | --- | --- |
| https://upload.wikimedia.org/wikipedia/commons/4/4a/FAE_visualization.jpg  *Figure. Visualization of how a car deforms in an asymmetrical crash.* [4] | *https://www.nibib.nih.gov/sites/default/files/inline-images/spect.jpg*  *Figure. SPECT computer generated imaging of radioactive tracer molecules* [5] | https://cdn.collider.com/wp-content/uploads/avatar-movie-image1.jpg  *Figure. Image from CGI movie Avatar* |

Rendering is often classified into two distinctive categories, real-time and offline. The former is mostly concerned with rapidly generating the image on the computer, it is therefore more performance-oriented. A requirement of real-time rendering application is a fast-successive cycle of rendering and user interaction. If this process runs at a fast-enough rate that the user does not see individual images, then the viewer becomes immersed in the application. The physical quantity at which these images are renderer is measured in frames per second (FPS) or Hertz (Hz). [2]

Rasterization is a technique that is generally chosen for real-time constraints. This is because rasterization is extremely fast compared to other rendering techniques such as ray tracing. Though, in the recent years, the trend is more shifting towards a hybrid approach combining rasterization and raytracing. As a result, this hybrid solution can take advantage of both techniques.

On the other hand, we have offline rendering, where the rendering process is usually much less performance critical. This opens the opportunity to use time expensive techniques which would not be suitable for real-time requirements [6]. Offline rendering is mainly used in the film industry to create high fidelity graphics. For this category, ray tracing is commonly the preferred rendering techniques due to its ability to produce output that is nearly indistinguishable from reality.

In this paper, the last category is the focus. The light transport algorithms that will be studied are more appropriate in the context of an offline environment. Although, such comparisons are possible in real time constraints, it requires a large amount of well-done optimizations and/or external libraries, APIs (Nvidia OptiX, DXR, Radeon Rays ...) and/or dedicated hardware (Nvidia GeForce RTX series GPU). In this thesis, the primary focus lies on the algorithms itself. Thus, we chose for an offline environment over real time constraints.

This thesis starts by providing the necessary background information related to the research. A condensed introduction of the important radiometric quantities will be provided. Then, using the radiometry knowledge, the rendering equation will be explained and its importance in rendering in general will be illustrated. With the previous information, the two light transport algorithms are defined: Whitted-Turner and Path tracing. Afterwards, we will detailly formulate our research question and follow up by a hypothesis. In the section ‘Case Study’, specifics concerning the implementations will be pointed. We then continue by conducting several experiments. This includes performance-oriented and accuracy experiments. Then an analysis of the results of the experiments shall be carried. The conclusions drawn from the different experiments will answer the sub questions of the main research question. By answering these sub questions, the main research question got answered indirectly, as these sub questions and their accompanied answers form a direct answer to our initial research question. Afterwards, we’ll formulate our findings and answer the research question in its fullest. Finally, we’ll discuss some possible further work.

## Chapter II. Research

A real digital camera is nothing less than a measurement apparatus than measures the amount of light energy that strikes every pixel of its sensor. This process results in a two-dimensional data records, that is more commonly known as a digital image or digital photograph. Thus, in order to synthesize such an image on a computer, we often use a relatively similar approach. As a result, rendering is measurement problem that requires a well-defined quantitative representation of the light energy. [7]

### Radiometry

The whole field of rendering is about simulating the behavior of light. Thus, we need a way to precisely describe how light is represented [8]. This is where radiometry and photometry come in, as they are part of physics optics. Photometry primarily focuses on determining optical quantities that are related to some degree to the sensitivity of the human eye [9]. While radiometry deals with the description of electromagnetic propagation, thus including visible light. The study of radiometry is based on the principle that light propagates at the geometric/ray optics level, thus certain properties and phenomena of wave optics are abstracted and ignored [8]. An important result of modeling light in geometric/ray optics, is that light propagates along a ray.

There are large variety of radiometric quantities, but as an introduction to radiometry, only the most important ones for rendering will be handled. The ones that are to be discussed are radiant energy, radiant flux, irradiance, solid angle and last but not least radiance.

Radiant Energy Q:

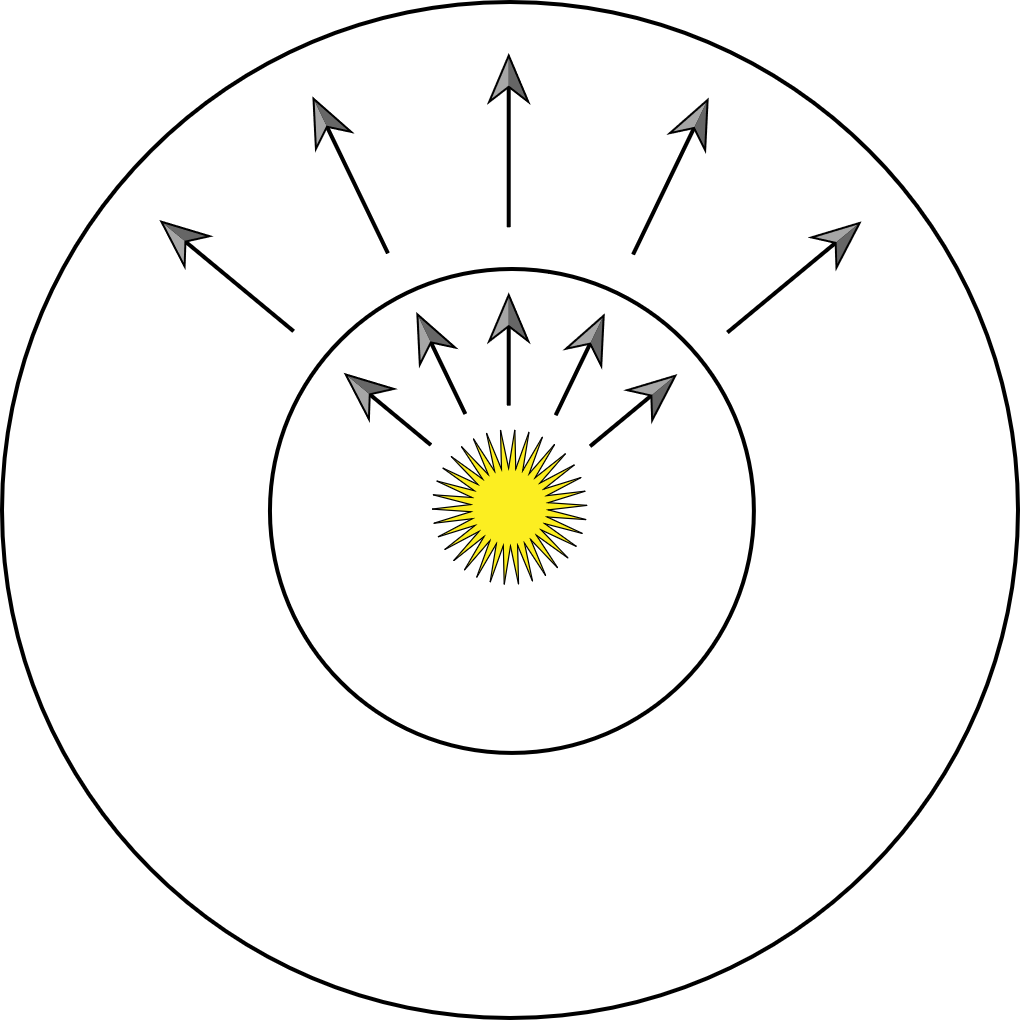
Radiant energy is a quantity that measures the energy of electromagnetic radiation, which includes light, measured in joules with symbol J. This can be thought as the energy carried by all the photons in a light bundle [10].

Radiant Flux Φ:

Radiant flux, also known as power, is the amount of radiant energy that pass per unit time. Its SI units are watts (W), which is equivalent to joules per second (J/s). This can mathematically be expressed as [8]:

Irradiance E:

Irradiance, E, is the amount of radiant flux arriving or leaving at a surface. Thus, irradiance is the flux density at a given surface area and is measured in W/m².



*Figure. Radiant flux and irradiance, the concentric circles receive the same flux but different irradiance* [8]*.*

Solid Angle ω:

Solid angles are in some way an extension of angles in two dimensions to three dimensions. Planar angle is defined as the ratio of arc length to the radius or equivalently the length of the arc on a unit circle and is measured in radians. While solid angle, is the projection of an object on a unit sphere and is measured in steradians [8].

|  |  |
| --- | --- |
| *Figure. Planar angle (2D)* [8] | *Figure. Solid angle (3D)* [8] |

Radiance L:

The last and most important, radiometric quantity is radiance. Radiance is the radiant flux per solid angle and per projected area with units expressed as []. Thus, radiance measures the flux from a point in space, coming from a certain direction on a surface. Another way to illustrate it, is to think as the amount photons travelling in a direction [10]. Definition:

It is important to note that radiance in our case is a function in terms of a point in space and direction, denoted as .

One of the reasons that makes radiance such a crucial component, is because it remains unchanged along rays through vacuum. Consequently, it is a natural quantity to choose whenever using ray tracing [8]. We’ll assume that light travelling through media like air also remains constant. In reality, there is a certain amount of absorption in the medium. This is generally nearly negligible and thus we’ll make this simplification.

Outgoing and incoming radiance:

The conservation of radiance along a ray implies for two points in space and , with no object in between, that the radiance leaving towards must all reach . This means that we only have to consider the radiance at intersection with a surface and not anything in between. We can state this behaviour formally as [7]:

Where

denotes as a small constant close to zero.



*Figure. Incoming radiance and outgoing radiance.* [8]

In order to simplify the formulations, we will use the following symbols and to indicate radiance leaving a surface and radiance arriving at a surface respectively [7].

If is a point on the film of a camera and is a direction towards our scene, then computing the radiance is exactly what we need to render the pixel of an image.

Bidirectional Reflectance Distribution Function (BRDF):

Whenever a light ray hits the surface, the surface can absorb and/or scatter the light, thus reflecting it back to the environment. For the case of the skin of a lemon, light striking at the surface of the lemon will mostly absorb the blue frequencies but will reflect red and green wavelengths of incoming light. Thus, the yellow appearance color of the lemon under white light condition. Its color remains approximately the same on all observation angles. While the image that is seen on a mirror surface, is entirely dependent on the viewing angle. As soon as the viewer moves, the reflected image changes immediately. We will ignore complex materials such as translucent surfaces, where light that enters the surface at some point might exit the surface at another point. In order to describe how light gets reflected on a surface, we will use the bidirectional reflectance distribution function (BRDF). The BRDF, , formally states [8]:

That means that the BRDF is the ratio between a differential outgoing radiance for a direction to a differential incoming irradiance at a direction . Simply stated, the BRDF indicates how much incoming light gets reflected towards the outgoing direction.

Lambertian BRDF

One of the simplest BRDF is the Lambertian BRDF which is a perfect diffuse BRDF. All incoming are reflected equally in all directions over the hemisphere. However, no real surface behaves exactly in this way. Depending on the material it can be good enough approximation. The full derivation of a perfect diffuse BRDF can be found in the appendix. Below, we are stating the result where is the reflectance.

Reflective BRDF:

Given a pair of and , a reflective surface will only reflect towards , if is no different than the reflected direction. This means that the light is only reflected in a single direction. An incoming light ray has no contribution to directions other than the reflected direction. Thus, this BRDF is a good candidate to represent it as a delta function. A direction, can be represented in spherical coordinates as . We can show than the reflected direction from an incoming direction can be represented as

. Such a BRDF can be derived to result in the following expression [10]:

With as the reflection coefficient, as the reflection color and as a delta function.

Light tracing

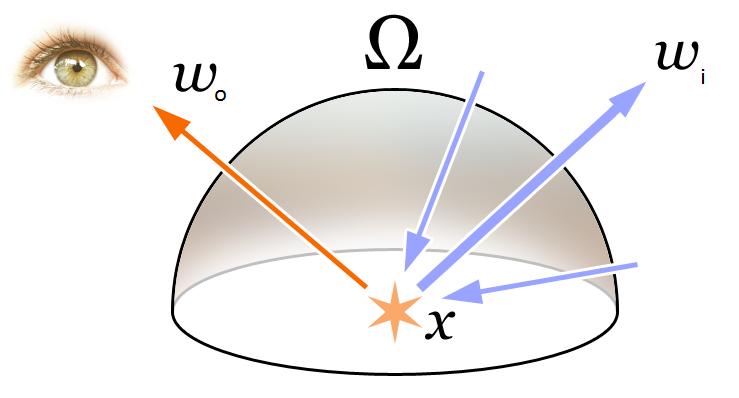
A light source emits a large number of light rays that travels in a straight path. These starts from the source until it hits the surface of an object. Part of the rays gets absorbed, other gets reflected according the BRDF. This continue, until a ray reaches our eye. The resulting ray allows us to ‘see’ what the ray has hit. In order to simulate this process, we’ll need to generate a very large number of rays from the light source. Eventually, only a very small portion of these will reach our eye [11]. The rays that don’t arrive at our eye has no contribution at all. This works in nature but is an impractical approach to use on a computing device, as it has a ceiling on its processing power.

Ray tracing

The idea of ray tracing, instead of wasting computational resources on generating rays that potentially have no contribution, is to generate rays backwards from our eye. The rays starting from our eye, also called primary rays, will then possibly be reflected on a surface intersection into another direction. Some of which will arrive at the light source and those will contribute to what is visible. In short, ray tracing in the reverse procedure of light tracing and vice versa. But this approach proved to be much more practical than the reverse.

### Rendering Equation

The rendering equation is probably the single most important equation in image synthesis. A large amount of light transport algorithms are attempts to solve or approximate this equation. It is an integral equation which states that the outgoing radiance is equal to an emitted radiance and the sum of radiance from all incoming directions:



*Figure. The rendering equation describes the total amount of light emitted from a point x along a viewing direction* [12]*.*

We are for example interested in the outgoing radiance from a point in the scene towards our eye. This equation describes how one might compute the radiance of interest. However, this equation cannot be computed in this form directly. As the unknown radiance, is recursively dependent on other unknowns, . It can be easily showed that this is a problem of infinite dimensions. Therefore, we need either an iterative solution or an approximated trade-off. This is where the two light transport algorithms come in, Whitted Turner and Path tracing.

### Whitted Turner Algorithm

Whitted Turner is one of the pioneers in ray tracing algorithms, he described a way to compute specular indirect illumination, perfect reflection and refraction. He started from the ray casting algorithm, proposed by Arthur Appel, which is to trace rays from the camera point (also known as primary rays), one per pixel and to find the closest intersection and then shade accordingly. Whitted extended by introducing three types of ray, shadow, reflection and refraction rays. Depending on the intersected material, one of these rays are generated. On a diffuse material, direct illumination is computed by sending a shadow ray towards each light source. If no geometry is found in the ray’s path, then the pixel is simply shaded according the diffuse surface. On a mirror-like surface, we’ll shoot reflective ray that goes through the same algorithm again. Transparent types will also generate a reflective ray, but on top of it a refractive ray. Shadow rays are terminal rays, as they don’t generate any rays afterwards. While reflection and refraction rays will possibly generate many more recursively. This approach will thus indirectly traverse a tree of rays.

The pseudo code below illustrates the common structure between both algorithms, which is to generate primary rays per pixel and then average in order to avoid aliasing.

Pseudocode of the common ray tracer:



Pseudocode of Whitted Turner algorithm:



### Path tracing Algorithm

Path tracing is a simple algorithm that follows the rendering equation closely. It is a Monte Carlo solution to the rendering equation. The idea is to send rays from the camera into the scene, as with Whitted Turner algorithm, but on surface contact a new ray is spawned. This new ray might hit a surface which causes another spawn ray. This process continues until a light source is hit. How the rays are generated are based on the BRDF of the material that is intersected. In the case for a diffuse BRDF, the generated ray would be a random ray over the hemisphere of the intersection. The pseudocode of path tracing is simple as can be seen below.

Pseudocode of Path tracing algorithm:



### Research Question

In the following sections, we will attempt to answer our research question which is:

*What advantages does Whitted Turner algorithm offer over Path tracing, and what advantages does Path tracing has over Whitted Turner algorithm?*

In this paper, we are going to perform a comparative study on Whitted Turner algorithm and Path tracing algorithm in the context CPU ray tracing. The study will be performed as a mean to answer this research question. In the first part we perform a performance-oriented comparison with varying parameters. In the second part of our ‘Case study’, we’ll compare the visual accuracy of outputs generated by the algorithms with a reference image. Based on the data and our interpretation of the data, we’ll formulate an answer on our research question.

### Hypothesis

Only based on the information described above, we can formulate a hypothetic answer to our research question. Path tracing requires hitting the light source in order to return a non-zero valued radiance. Thus, we can already say that Path tracing will probably produce a lot of noise and converge relatively slow. On the other hand, Whitted Turner algorithm account directly for the light source, so we think that it will result in less noise and faster convergence. In terms of accuracy of the output, Path tracing will probably generate a closer to reality output because it accounts more different light paths than Whitted Turner algorithm.

## Chapter III. Case Study

In the first part of this section, we’ll explain details about our implementation. Then, we will answer our research question, which was described at the end of the previous section. To answer the question, we will perform a comparative study on Whitted Turner and Path tracing algorithm. First, we’ll analyze their running time on increasing samples per pixel, resolution and multiple scenes. Afterwards, we compare the output of both algorithm with a reference image.

### Implementation

The implementation was fully written in C++14 and utilizes OpenGL in the backend to display the output on a window. The renderer is multi-threaded on 2 threads via OpenMP. All the experiments were executed on an Intel i7-4710HQ CPU @2.5GHz in 64bit release mode.

Our implementation contains a few changes compared to the original algorithms described at the end of the previous section. Whitted Turner algorithm doesn’t support area light, while Path tracing does support area light but not point and directional light. Path tracing cannot out of box support point light because the probability for a light ray to hit a point is zero, thus the output would be fully black. So, in order to compare them, we decided to modify the base Whitted Turner to support area lights. As a result, all our scenes contain only area light. It is worth mentioning that all real lights have a certain area in contrast to point and directional light.

We found that brute-force Path tracing has an extremely slow convergence. It produces a very large amount of noise even at relative high number of samples. In order to produce results in a reasonable time, we used a technique called ‘Next Event Estimation’, which is basically sampling the light directly as one would do with Whitted Turner. We experimented with the Next Event Estimation on first, last and every bounce and decided to choose for the last one, as it seems to converge rapidly. However, we are not entirely sure if this result in a biased solution.

For Path tracing, in order to be mathematically correct, the number of bounces should go to infinity. But as this is an impractical approach to compute, one could define a hard limit of numbers of bounces to terminate the computation. If the chosen number is too low, the image will be visibly biased. Choosing a too high limit will result in wasting computational resources. By implementing ‘Russian roulette’, this problem can be alleviated. Each ray has a probability to be terminated based on how much the ray contributes to the output. Non-terminated ray on the other hand get boosted by the probability to survive. One can show that this technique produces unbiased result.

Lastly, our software architecture is heavily inspired by the design of Kevin Suffern, which he described in his book, Ray Tracing from the Ground Up. All boiler plate codes that are necessary to build the application, are based on a previous fork of the Radiance graphics engine [13].

### Performance Comparison

In this section, comparisons were performed between both algorithms. First off, the analysis of samples per pixel – running time will be showed, follow up, by image resolution - running time experiment and lastly will be an observation of how the algorithms run on different scenes.

#### Samples Per Pixel

The data were recorded using the scene data of the Cornell box, but slightly adapted by Kari Kivisalo. All the outputs were rendered as 300x300 resolution image. For each amount of sample per pixel, the experiment was executed 5 times. Then the average of 5 iterations were used to plot the graphs below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Samples per Pixel | Running time 1 (s) | Running time 2 (s) | Running time 3 (s) | Running time 4 (s) | Running time 5 (s) | Average time (s) |
| 1 | 8,19E-02 | 8,25E-02 | 8,25E-02 | 8,21E-02 | 8,22E-02 | 8,22E-02 |
| 4 | 1,92E-01 | 1,94E-01 | 1,71E-01 | 1,97E-01 | 1,97E-01 | 1,90E-01 |
| 16 | 6,34E-01 | 6,20E-01 | 6,05E-01 | 6,25E-01 | 6,10E-01 | 6,19E-01 |
| 64 | 2,30E+00 | 2,44E+00 | 2,43E+00 | 2,39E+00 | 2,13E+00 | 2,34E+00 |
| 256 | 1,03E+01 | 1,18E+01 | 1,19E+01 | 1,31E+01 | 9,52E+00 | 1,13E+01 |
| 1024 | 3,88E+01 | 3,47E+01 | 3,59E+01 | 3,41E+01 | 3,71E+01 | 3,61E+01 |

Table. Running time of Whitted Turner algorithm per samples per pixel

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Samples per Pixel | Running time 1 (s) | Running time 2 (s) | Running time 3 (s) | Running time 4 (s) | Running time 5 (s) | Average time (s) |
| 1 | 2,36E-01 | 2,45E-01 | 2,45E-01 | 2,45E-01 | 2,35E-01 | 2,41E-01 |
| 4 | 8,44E-01 | 8,41E-01 | 8,41E-01 | 8,40E-01 | 8,31E-01 | 8,40E-01 |
| 16 | 3,18E+00 | 3,18E+00 | 3,17E+00 | 3,17E+00 | 3,18E+00 | 3,18E+00 |
| 64 | 1,27E+01 | 1,24E+01 | 1,24E+01 | 1,25E+01 | 1,25E+01 | 1,25E+01 |
| 256 | 5,10E+01 | 5,16E+01 | 5,33E+01 | 5,21E+01 | 5,11E+01 | 5,18E+01 |
| 1024 | 2,06E+02 | 2,08E+02 | 2,02E+02 | 2,04E+02 | 2,04E+02 | 2,05E+02 |

Table. Running time of Path tracing algorithm per samples per pixel

Figure. Linear-Log plot of running time and samples per pixel.

The result of the linear-log plot is not clear to see, especially for low samples per pixel. This is due to the x-axis, number of samples per pixel, being plotted logarithmically. Below a log-log plot is provided, which is much clearer to see.

*Figure. Log-Log plot of running time and samples per pixel.*

From the tables and graphs above, it is clear that Path tracing is slower than Whitted Turner, especially for higher samples per pixel. This is expected, as the Cornell Box is only composed of diffuse objects, which would terminate Whitted Turner approach on ray-intersection. While Path tracing doesn’t stop there.

Based on the linear results in the log-log plot, both functions in a linear-linear plot would be of the form where and are constants with respect to x.

In the log-log plot, there is a linear relationship that can be expressed as:

With m the slope of the log-log plot.

Filling the points in the equation above gives us:

Hence, both methods running time are linear with respect to the samples per pixel.

This result is not a surprise, as increasing the number of primary rays by a factor of K, increases the computational work by the same factor K. This is because the work of each light paths is independent from each other.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Figure. Whitted Turner SPP experiment, from top left to bottom right 1SPP, 4SPP, 16SPP, 64SPP, 256SPP and 1024SPP

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Figure. Path tracing SPP experiment, from top left to bottom right 1SPP, 4SPP, 16SPP, 64SPP, 256SPP and 1024SPP

Although from data timings, Path tracing seemed to be only slower by a factor 4, it is much slower when comparing the image output. Whitted turner converges really fast, around 16 samples per pixel produces an acceptable result. While Path tracing results for the same samples per pixel in a really noisy output and requires over 1024 samples per pixel to get an acceptable result.

#### Resolution

In this part, all renders are generated with the Cornell Box and keeping the samples per pixels constant at 4. Analogous to the previous experiment, the test per resolution is executed 5 times. The averages of these results are then used as for the graphs.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Resolution | Running time 1 (s) | Running time 2 (s) | Running time 3 (s) | Running time 4 (s) | Running time 5 (s) | Average time (s) |
| 32 | 2,32E-03 | 2,33E-03 | 2,32E-03 | 2,38E-03 | 2,38E-03 | 2,35E-03 |
| 64 | 9,12E-03 | 8,75E-03 | 9,04E-03 | 9,71E-03 | 9,70E-03 | 9,26E-03 |
| 128 | 3,71E-02 | 3,68E-02 | 3,58E-02 | 3,63E-02 | 3,55E-02 | 3,63E-02 |
| 256 | 1,41E-01 | 1,38E-01 | 1,36E-01 | 1,55E-01 | 1,46E-01 | 1,43E-01 |
| 512 | 5,53E-01 | 5,68E-01 | 5,59E-01 | 5,61E-01 | 5,52E-01 | 5,58E-01 |
| 1024 | 2,30E+00 | 2,19E+00 | 2,27E+00 | 2,23E+00 | 2,12E+00 | 2,22E+00 |

Table. Running time of Whitted Turner algorithm per image resolution

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Resolution | Running time 1 (s) | Running time 2 (s) | Running time 3 (s) | Running time 4 (s) | Running time 5 (s) | Average time (s) |
| 32 | 9,93E-03 | 9,59E-03 | 9,56E-03 | 9,67E-03 | 9,57E-03 | 9,66E-03 |
| 64 | 4,72E-02 | 3,85E-02 | 3,91E-02 | 3,84E-02 | 4,83E-02 | 4,23E-02 |
| 128 | 1,67E-01 | 1,57E-01 | 1,53E-01 | 1,57E-01 | 1,66E-01 | 1,60E-01 |
| 256 | 6,11E-01 | 5,95E-01 | 6,01E-01 | 5,99E-01 | 6,11E-01 | 6,03E-01 |
| 512 | 2,39E+00 | 2,40E+00 | 2,40E+00 | 2,38E+00 | 2,41E+00 | 2,40E+00 |
| 1024 | 9,39E+00 | 9,56E+00 | 9,96E+00 | 9,67E+00 | 9,39E+00 | 9,60E+00 |

Table. Running time of Path tracing algorithm per image resolution

Figure. Lin-Log plot of running time and image resolution.

Figure. Log-Log plot of running time and image resolution.

On all the dataset, Path tracing performs worse than Whitted Turner version. These results are relatively analogous to the previous experiment, this is because an increase in resolution, also increase the number of primary rays over the whole image. By multiplying the resolution by a factor , in both width and height, their will be more of numbers of light paths.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Figure. Whitted Turner resolution experiment, from top left to bottom right 32x32, 64x64, 128x128, 256x256, 512x512 and 1024x1024

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Figure. Path tracing resolution experiment, from top left to bottom right 32x32, 64x64, 128x128, 256x256, 512x512 and 1024x1024

One of the visual differences between increasing the resolution and increasing the samples per pixel, is that an increase in resolution keeps it aliasing artefacts on close up. One way to handle this would be to down sample the higher resolution image or increasing the number of samples.

#### Scenes

Before proceeding to the next section, an experiment shall be performed under multiple scenes. The scenes that will be considered are the ‘Cornell Box’ scene, a ‘Caustics’ scene, a ‘Reflection’ scene and a ‘Refraction’ scene. The Cornell Box is a nearly closed environment that is enclosed on each side except from the camera direction. The three other scenes are fully open and contains materials indicated by the name’s scene. The ‘Reflection’ scene contains an environment light which is not the case for the others. In this test all were rendered with 32 samples per pixel and an output resolution of 720 by 720.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Scenes | Running time 1 (s) | Running time 2 (s) | Running time 3 (s) | Running time 4 (s) | Running time 5 (s) | Average time (s) |
| Cornell Box | 7,24E+00 | 7,27E+00 | 7,17E+00 | 7,11E+00 | 7,10E+00 | 7,18E+00 |
| Caustics | 3,64E+00 | 3,62E+00 | 3,59E+00 | 3,68E+00 | 3,87E+00 | 3,68E+00 |
| Reflection | 4,84E+00 | 4,55E+00 | 4,69E+00 | 4,56E+00 | 4,63E+00 | 4,65E+00 |
| Refraction | 1,65E+01 | 1,69E+01 | 1,65E+01 | 1,65E+01 | 1,64E+01 | 1,66E+01 |

Table. Running time of Whitted Turner algorithm per scene.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Scenes | Running time 1 (s) | Running time 2 (s) | Running time 3 (s) | Running time 4 (s) | Running time 5 (s) | Average time (s) |
| Cornell Box | 3,41E+01 | 3,60E+01 | 3,46E+01 | 3,41E+01 | 3,41E+01 | 3,46E+01 |
| Caustics | 7,85E+00 | 7,38E+00 | 7,91E-01 | 7,86E+00 | 7,74E+00 | 6,32E+00 |
| Reflection | 7,54E+00 | 7,53E+00 | 7,66E+00 | 7,46E+00 | 7,69E+00 | 7,58E+00 |
| Refraction | 2,11E+01 | 1,94E+01 | 2,21E+01 | 2,30E+01 | 2,16E+01 | 2,15E+01 |

Table. Running time of Path tracing algorithm per scene.

Figure. Column graph running time per scene.

Overall Whitted Turner implementation is clearly the winner in term of running time. This is notably true for the Cornell Box scene due to being a somewhat closed environment with diffuse objects which cause Path tracing to keep tracing rays. The other scenes are open and as result, Path traced rays are terminated much faster than in a closed environment.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Figure. Whitted Turner scene experiment, from top left to bottom right Caustics scene, Cornell Box scene, Reflection scene and Refraction scene.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Figure. Path tracing scene experiment, from top left to bottom right Caustics scene, Cornell Box scene, Reflection scene and Refraction scene.

One interesting result is the fast convergence of Path tracing under an environment light. This largely boost the probability to hit the light source in contrast to the small area lights in the other scenes.

### Visual Accuracy Comparison

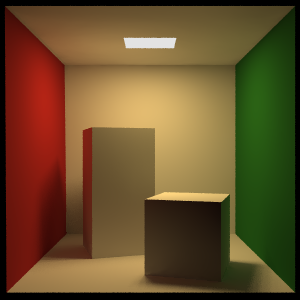
In this last comparison, a visual accuracy test will be made using the scene description of the Cornell Box. This Cornell Box version rendered with POV-Ray is slightly modified by Kari Kivisalo, as it fits our renderer more than the original from Cornell University. The original version contains surface reflectance data per wavelengths, which is not supported by our renderer. Note that although, we’re using the modified version, not all parameters were translated exactly to our system. As result, this will induce some errors regardless the algorithms. These renders are all 300 by 300 pixels. Note that the complete set of renders for this test can be viewed in the appendix, below are only a small part of the image set.

|  |  |  |
| --- | --- | --- |
| Samples per pixel | MSE Whitted turner | MSE Path tracing |
| 1 | 2,87E+03 | 2,04E+03 |
| 4 | 2,56E+03 | 1,44E+03 |
| 16 | 2,51E+03 | 1,36E+03 |
| 64 | 2,51E+03 | 9,09E+02 |
| 256 | 2,51E+03 | 5,03E+02 |
| 1024 | 2,51E+03 | 3,53E+02 |
| 4096 | 2,51E+03 | 3,18E+02 |
| 16384 | 2,51E+03 | 3,05E+02 |

Table. Mean squared error with respect to samples per pixel

Figure. Mean squared error with respect to samples per pixels.

From the graph and the table above, Path tracing has much less error than Whitted Turner algorithm. Increasing the amount of samples per pixel doesn’t help to reduce the error for the recursive ray tracing, while the error of Path tracing steadily reduces as the samples increases.

.  
Figure. Reference image, Cornell Box rendered with POV-Ray

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Figure. Reference image on the left, Whitted Turner output in the middle and on the right is the difference image. First row rendered with 1SPP and last row with 16384SPP.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Figure. Reference image on the left, Path tracing output in the middle and on the right is the difference image. First row rendered with 1SPP and last row with 16384SPP.

Without relying on the mean squared error data, one can wrongly deduce that Whitted Turner generates a more accurate result on lower number of samples, based on the render output. But the overall error of Path tracing is much lower at any samples. The difference images confirm that the error for recursive ray tracing doesn’t improve much. Whilst the error image of Path tracing converges to a complete black image. The difference image of Whitted Turner has three noticeable areas. The bright red and green spot on the sides of the smaller boxes are two of those. This is because Whitted Turner algorithm is not able to render diffuse-diffuse reflection. The ceiling is completely black for the recursive ray traced images, which is incorrect. Again, this is because indirect diffuse lights are not considered in the first algorithm. In the previous section, in the scene test, it is apparent that Whitted Turner algorithm could not generate any form of caustics. Lastly, although not visible, the original Whitted Turner is not capable of rendering soft shadow, while Path tracing doesn’t support punctual light and directional light as mentioned at the start of the ‘Case Study’

To better understand what light paths is supported or not by the algorithms, we can make use of Heckbert notation. Heckbert notation is a way to classify light paths. stands for light, , are labels for diffuse and specular surface respectively and corresponds to the eye or camera. One can proof that Whitted Turner algorithm can generate light paths of the form where “\*” stands for zero or more and “()” indicates an optional argument. Path tracing has the form of where “|” stands for “or”. Caustics are formed by light that is reflected or transmitted by a number of specular surfaces before interacting with a diffuse surface. [14] Cautics, formulated with Heckbert notation would be of the form where “+” stands for one or more. is a valid expression for but not , hence why it is not possible to render caustics using Whitted's solution. The color bleeding on the side of the boxes is the result of paths of the form of where “2+” denotes for two or more which are also valid light paths for . Lastly, the expression of Path tracing encompasses . Thus, all light paths generated by Whitted Turner method can also be produced using Path tracing.

## Chapter IV. Conclusion

The research question is reformulated below and answered based on the experiments performed in the previous chapter.

*What advantages does Whitted Turner algorithm offer over Path tracing, and what advantages does Path tracing has over Whitted Turner algorithm?*

Whitted Turner approach converges fast and executes overall much faster. Conversely, Path tracing outputs more truthful renders that approaches reality. This includes the ability to account for global illumination: diffuse-diffuse light transport and caustics. But this comes at the cost of large amount of noise and/or long rendering time.

#### Further Work

This paper handles these techniques in a CPU only environment. A possible next step would be exploring these on the GPU side with or without real-time constraints. Utilizing the GPU has several implications that are worth to delve into, for example to what extends are the methods parallelizable, dealing with shared memory...

From the studies in the previous sections, one can note that each approach has its pros and cons. Therefore, to what extend can the approaches combined into a new solution that minimizes their drawbacks and amplifies both their strengths. By detecting the area’s where Whitted solution are good enough, we can avoid using the expensive Path tracing and only use it where it really matters. For example, in a real-time scenario, geometry in the far distance will generally not benefit from an accurate result as this is generally not the focus for an end-user. While certain areas are more crucial to produce a correct result. We think that working towards a new combined solution is a path worth exploring.

## References

|  |  |
| --- | --- |
| [1] | "Rendering (computer graphics)," [Online]. Available: www.en.wikipedia.org/wiki/Rendering\_(computer\_graphics). [Accessed 30 October 2019]. |
| [2] | E. H. a. N. H. Tomas Möller, Real-Time Rendering, CRC Press, 2018. |
| [3] | "Application of Computer Graphics," 30 October 2019. [Online]. Available: www.linkedin.com/pulse/application-computer-graphics-niropam-das/. |
| [4] | "Nonlinear Finite Elements," 4 November 2019. [Online]. Available: ww.en.wikiversity.org/wiki/Nonlinear\_finite\_elements. |
| [5] | "Nuclear Medicine," 2 November 2019. [Online]. Available: www.nibib.nih.gov/science-education/science-topics/nuclear-medicine. |
| [6] | "What Is the Difference between ‘Offline’ and ‘Real Time’ Rendering?," 30 October 2019. [Online]. Available: www.gamedev.stackexchange.com/questions/70244/what-is-the-difference-between-offline-and-real-time-rendering. |
| [7] | M. McGuire, The Graphics Codex, Casual Effects Publishing, 2018. |
| [8] | M. a. G. H. Pharr, Physically Based Rendering: From Theory to Implementation, Amsterdam: Elsevier Science, 2010. |
| [9] | "Basic Radiometric Quantities," 1 November 2019. [Online]. Available: www.gigahertz-optik.de/en-us/basics-light-measurement/light-color/quantities-radiometric/. |
| [10] | K. Suffern, Ray Tracing from the Ground Up, A K Peters, 2007. |
| [11] | "Light Transport," 1 November 2019. [Online]. Available: www.scratchapixel.com/lessons/3d-basic-rendering/rendering-3d-scene-overview/introduction-light-transport. |
| [12] | "Rendering Equation," 1 November 2019. [Online]. Available: www.en.wikipedia.org/wiki/Rendering\_equation. |
| [13] | Nickelium, "Radiance Engine," [Online]. Available: https://github.com/Nickelium/Radiance. |
| [14] | H. W. Jensen, "Images with Caustics," 3 November 2019. [Online]. Available: www.graphics.stanford.edu/~henrik/images/caustics.html. |
| [15] | T. Whitted, An improved illumination model for shaded display, Communications of the ACM, 1980. |
| [16] | L. J. G. Eric Veach, Robust monte carlo methods for light transport simulation, Stanford University, Stanford, CA, 1998. |

## Appendix

#### Lambertian BRDF derivation:

In order to derive the Lambertian BRDF, we need to first define the reflectance quantity, (hemispherical-directional reflectance). Reflectance is the defined as proportionality between the outgoing flux and incoming flux:

We want to express the BRDF in terms of the reflectance, thus we’ll first find a formulation for and . Starting from the definition of radiance:

Moving back to the definition of the BRDF:

Because the Lambertian BRDF is constant in terms of and , we can move it out of the integral:

As the is independent of and , we can move it outside the integral in the flux formula:

From the previous derived formula, we find this compact representation for a Lambertian BRDF [10]:

#### Complete render of the visual accuracy test

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Figure. Reference image on the left, Whitted Turner output in the middle and on the right is the difference image for 1, 4, 16, 64, 256, 1024, 4096 and 16384SPP.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Figure. Reference image on the left, Path tracing output in the middle and on the right is the difference image for 1, 4, 16, 64, 256, 1024, 4096 and 16384SPP.