Modelling Non-Linear Spacetime Deformations via Ray Traced Null Geodesics

UNIVERSITY OF
LINCOLN

Ashley Knowles

School of Computer Science

College of Science

University of Lincoln

Submitted in partial satisfaction of the requirements for the
Degree of Master of Science
in Computer Science

Supervisor Mr. Philip Carlisle

July 2019

Abstract

In this paper, several methods and optimisations for visualising non-linear light traversal via ray tracing are proposed. By representing the geodesic path of light traversal as discrete delineations, the expensive collision computations involved in ray tracing can be significantly optimised to allow for real time globally illuminated rendering. Additionally, an adaptive ray integration technique is employed to dynamically adjust the length of the rays as more or less detail is required. Combined with modern optimisation techniques such as the use of SIMD intrinsics and multithreading, a convincing and realistic simulation of a singularity is created. Finally, the system is evaluated and compared against state of the art works as well as a discussion about its potential applications in both video game development and scientific relativistic simulation.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Rationale

With the advent of modern graphical processing hardware Nvidia, 2020a, ray tracing is rapidly becoming the de-facto standard for light transport and global illumination.

By physically simulating the traversal of light, ray tracing is able to produce high fidelity realistic renderings of real world counterpart scenes.

Indeed, as raw compute power is becoming cheaper and much more easily accessible, as well as with the recent introduction of acceleration methods such as machine learning, widespread adoption of ray tracing is becoming increasingly likely.

The technique is currently being explored in numerous applications, ranging from scientific research to visual effects in video games and entertainment Nvidia, 2020a.

Conventional ray tracing algorithms assume that the path through which light rays travel is strictly linear from source to destination, with no curvature or deviation from the original path. Non-linear ray tracing however, allows for the paths to become curved or deformed under the influence of an external force.

Although prior research has been conducted into the visualisation of non-linear space via ray tracing, many implementations significantly simplify the problem, through means such as dimensionality reduction or by lowering the accuracy of the mathematical approximations of light.

1.2 Aims, Objectives and Hypothesis

The aims of this thesis are several fold. The first, is to better understand the limitations inherent to nonlinear ray tracing. As each form of non-linearity is implemented, a significant impact on the performance of the application is expected, thus having a deeper understanding of the ways to optimise and improve these algorithms will be of vital significance.

The second aim is to propose and implement several new, useful and exciting visual effects, as a means to demonstrate the possibilities of what can be achieved with such a system.

Finally, after the implementation, an evaluation of each method will be conducted to measure accuracy, performance, image quality and practicality to determine how well each algorithm performs.

1.3 Report Structure

- Chapter 1 discusses the rationale for the undertaking of the paper, as well as the overall aims and objectives and finally the contributions this paper will make.
- Chapter 2 covers related work in the subject field, highlighting existing work to examine and build upon.
- Chapter 3 outlines the methodology used to design the proposed techniques and evaluate the artifacts that were produced.
- Chapter 4 lays out the requirements necessary to complete the aims, as well as the initial design and final implementation of the system.
- Chapter 5 evaluates the final system against the initial aims, to determine if they were completed successfully.
- Chapter 6 finalises and draws conclusion about the results of the implementation.

1.4 Contributions

The contributions made in this thesis are in the areas of physics simulation and computer generated graphics. State of the art ray tracing techniques are combined with null geodesic equations that describe the distortion of spacetime around objects exhibiting a large gravitational attraction.

The ideas and constructs put forth in this paper make contributions not only to video game development but also to academia in the areas of general relativity and black hole research.

A linear ray tracer is first constructed, after which the non-linearity is introduced to showcase various different effects that become possible after small tweaking to the equations in the implementation.

The implementation's potential in different fields is discussed, and how it can be used within video games, as well as its usefulness to ongoing scientific research efforts.

Chapter 2

Related Work

2.1 Linear Ray Tracing

Although many linear ray tracing techniques are well documented in computer graphics literature, evaluating the different methods can provide a useful insight into the various characteristics, such as performance and realism, that each method exhibits in order to build a theoretical foundation for non-linear ray tracing.

To begin, it is important to first make the distinction between ray casting and ray tracing, as these are unrelated concepts for the purpose of this literature review. Ray casting, as seen in games such as Wolfenstein 3D iD, 1992, is defined by Vandevenne, 2020 as a rendering technique to create a 3D perspective from a 2D map. Inversely, ray tracing allows for Globally Illuminated rendering by recursively bouncing light rays around a scene to support reflections and shadows.

Ray tracing in its most basic form is not a new concept, as can be seen in work by Appel, 1968, however only recently has computational power advanced in such a way to allow these techniques to be used in real-time critical applications.

In most literature, two ray tracing techniques are commonly referenced. The first, as put forth by Glassner, 1989, is forward tracing. Often referred to as photon tracing, forward tracing is a method of casting rays (or the equivalents of photons) from a light source to an observer, or in this instance a camera.

The source sends out a theoretically infinite number of rays in every direction, bouncing on objects within the scene until they eventually hit the observer, where each

ray is collected and the final image frame is formed over several samples using a Monte Carlo average.

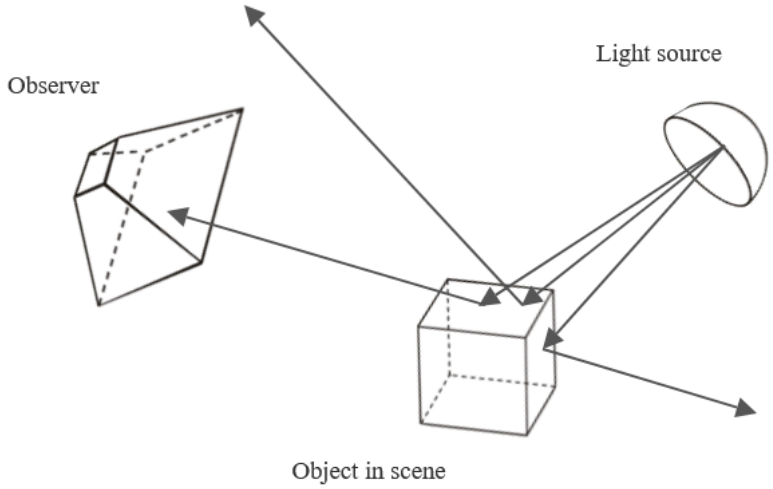


Figure 2.1: Forward ray tracing with a single light source and object in a scene

The primary advantage of this technique is the ability to render caustics. Caustics are intricate light patterns formed as light travels through a specular material, into a diffuse material, before finally entering the eye or camera. Caustics are regarded as being notoriously difficult to render,

Although this forward ray tracing method produces the most realistic and accurate image, figure 1 demonstrates that a large majority of photons sent by the light source do not ever reach the camera view frustum, wasting valuable computational resources.

Techniques exist for reducing the cost of this method, for instance Glassner, 1984 proposes the use of octree-based space subdivisions, associating a given bounding box with only those objects whose surfaces pass through the volume. Such octree based approach may provide significant performance benefits to a ray tracing algorithm, however a disadvantage with this method is that some bounding volumes may be

of an inappropriate shape for the object contained within it, creating a large "void" area in which only a small portion of the object exists, shown in Fig 2.

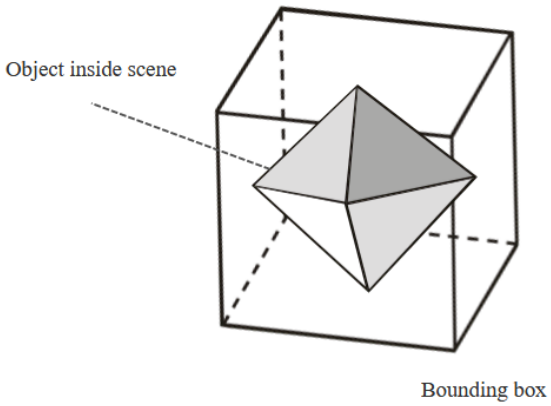


Figure 2.2: A bounding volume of inappropriate shape for the object contained within

The second method of ray tracing which aims to solve the aforementioned problems is backwards tracing. In backwards tracing, the rays are created at the image plane and cast into the scene, instead of being emitted by a light source, as demonstrated in the figure below.

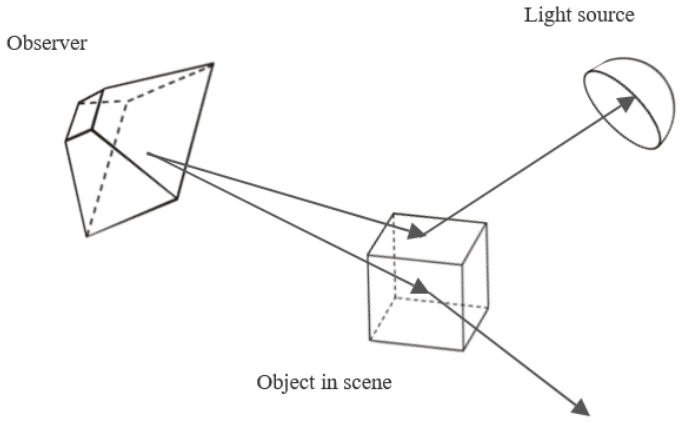


Figure 2.3: Backward ray tracing with a single light source and object in a scene

Although this method is indeed faster, in some cases the assumption that the only light to contribute to the final scene is that which leaves the view plane may be incorrect. As noted by Roberts, 1997, if for example a lens is held at a distance from a surface and is illuminated by a light source directly above, a focal point will exist beneath the lens with a large concentration of light (caustics). A backwards ray tracer intrinsically is unable to produce such an effect, as multiple rays are required to enter the image plane in an overlapping fashion, increasing the overall intensity of the final output colour.

Ray tracers can be categorized into those that are stochastic and those which are deterministic. Deterministic, or recursive ray tracers, are the simplest form of ray tracing, where a single ray is fired from the view plane per pixel, and bounces to all light sources in the scene to determine which are visible. A large limiting factor for this method is that the resultant output does not consider additional light bounces from other objects in the scene, and thus global illumination is not possible.

Cook, Porter and Carpenter, 1984 introduces distributed ray tracing as a stochastic method in order to achieve anti-aliasing through spacial oversampling. Spatial oversampling fires multiple rays from each pixel in the view plane, averaging the results from multiple samples into a single rendered frame. This technique acts as a simplistic form of denoising and produces a relatively clean image.

In order to simulate global illumination, that is - effects such as light bleeding onto other surfaces in the scene as well as reflections, refraction and other optical effects, the Monte Carlo integration method by Metropolis and Ulam, 1949 can be used to solve the rendering equation introduced by Kajiya and J.T, 1986. The Monte Carlo method is a generalised mathematical operation to compute the integral of a given function by selecting random samples in a domain and averaging the value of the function at said samples. In the context of ray tracing, it is possible to calculate the amount of light entering the view plane per pixel given multiple rays. Although the distributed tracing technique above is similar to the Monte Carlo method, it is a more simplistic approach and is unable to solve the rendering equation, which offers

a large amount of realism for the scene through radiance and energy conservation approximations.

A significant drawback to the Monte Carlo method however is the production of visual artefacts known as fireflies. Fireflies are bright anomalies that appear out of place next to adjacent pixels, as a result of a low probability event that a ray may be unable to hit a light source in the scene, leaving a dark or discoloured pixel. Fireflies are often resolved naturally given more samples, however it could be the case that some may occur in areas which never resolve due to numerical inaccuracies, such as those inherent to floating point numbers.

Numerous methods exist for remedying the firefly problem as well as general visual noise reduction, for example the first-order regression technique Bitterli et al., 2016 or NVidia's proprietary RTX based approach. As shown in the comparison below, denoising has a large impact on the overall visual clarity of the scene.

Figure 2.4: A ray-traced image with fireflies on the left, and denoised on the right

While the aforementioned techniques are effective, they are also generally computationally expensive. The RTX technique, for example, relies on specialised and proprietary GPU architecture to enable fast computation of their neural network

solution, as well as requiring significant changes to the target application to support the framework.

2.2 Non-Linear Ray Tracing

Having covered various linear ray tracing literature and establishing a solid theoretical framework, it is now possible to review techniques for transforming the framework in such a way to allow for non-linearity.

Einstein's theory of general relativity describes how, given source of matter exhibiting a strong enough gravitational attraction, spacetime is able to warp around it. Light rays travelling from a source, usually a black hole, to the observer become curved as they pass through the distorted spacetime, creating an effect known as a gravitational lens. The phenomenon, often referred to as an Einstein ring, allows the observer to view objects that would under normal circumstances be occluded behind the gravity source, producing a ring of light originating from beyond the Schwarzschild radius - that is, the point at which no light can escape from a black hole.

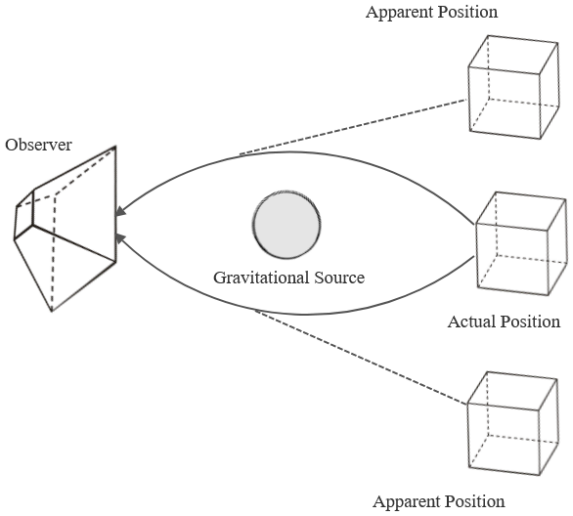


Figure 2.5: Gravitational lensing effect on an object occluded by a gravity source

To produce effects such as this, Gröller, 1995 introduces a generic approach to non-linear ray tracing as a visualization technique, using multiple sources of non-linearity such as gravity centers, gravity lines, chaotic dynamical systems and parametric curved rays.

Although all aforementioned sources are viable methods of producing gravitational distortion effects, the use of curved rays is of particular significance. When travelling through distorted spacetime, the path that a photon takes may be expressed as a simple curve, defined by the mass and gravitational field strength for each object influencing it.

As the entire path of a curve is known in advance, efficient ray-object intersection testing can be performed. As well as this, the method is both the simplest to use with an existing linear ray tracer, and arguably produces the most accurate results.

Efficiency is of great importance for ray tracing, and Gröller, 1995 notes that most of the computational cost of ray tracing comes from the ray intersection tests, which can be particularly expensive in the case of curved rays. To significantly reduce the performance impact, a technique known as linearization can be used.

The linearization technique uses Euler integration to approximate linear paths along the curve, transforming one complicated intersection test with several linear tests. The curved path that a ray of light takes around a source of gravity is defined as:

$$\vec{X}'' - \frac{g \cdot M_g \cdot (\vec{X}_n - \vec{X}_g)}{||\vec{X}_n - \vec{X}_g||}$$

The equation can then integrated to produce discrete linear vector segments with origins and directions approximating the original path:

$$\vec{X}_{n+1} = \vec{X}_n + h \cdot \vec{V}_n$$

$$\vec{V}_{n+1} = \vec{V}_n - \frac{g \cdot M_g \cdot (\vec{X}_n - \vec{X}_g)}{||\vec{X}_n - \vec{X}_g||} \cdot d(||\vec{X}_n - \vec{X}_g||)$$

Each new linear ray begins at the end of the previous, with position vector $\vec{X}_{n+1}$, direction vector $\vec{V}_{n+1}$ and length h defining the step size of the Euler integration function.

The effect of this can be seen clearly in the figure below, with multiple linear rays defining the curve of light as it travels close to the gravity source.

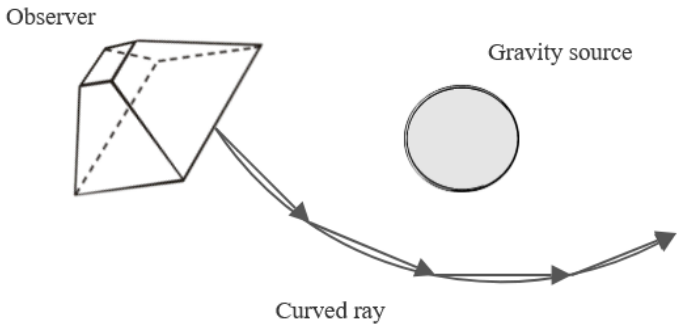


Figure 2.6: Linearization of a curve around a gravity source

By approximating a curve to multiple linear rays, the same linear algorithms that have received extensive scrutiny in literature can be applied, significantly increasing overall performance.

While the process of linearisation which relies on discrete integration steps provides a significantly large performance boost to the simulation, it also results in unavoidable inaccuracies tied to the interval of the discrete steps. If the step size is too large, the distance between each ray increases, causing collision intersections to miss. These inaccuracies can lead to visual artefacting and misaligned ray hits.

Weiskopf, 2000 presents a visualization technique to observe objects under the influence of a gravitational field, using a null geodesic based approach. Geodesics, or

null geodesics in reference to light, are defined as the straightest path a photon can take through distorted or curved spacetime. These geodesics are of interest as the mathematical formulae used to describe them can also be applied in a path tracing application, as demonstrated in the aforementioned paper. The work applies this method to visualising rigidly rotating rings of dust as well as warp drive bubbles. Although competent visualisations were produced, a somewhat naive approach was taken, in that the author extended a preexisting ray tracing solution, RayViS, which contains no implementation of global illumination effects such as reflection, refraction or realistic lighting conditions that would have significantly contributed to producing a more physically accurate simulation.

Additionally, Weiskopf, 2000 notes that the rendering time for standard linear three-dimensional ray tracing is determined by the computation of intersections between rays and objects, however this is not the case for non-linear relativistic ray tracing as generating curved light rays by solving geodesics plays an even more dominant role. In order to be able to solve geodesics in real time, extensive optimisations would be required, and thus it is likely more efficient to use a simpler, less accurate approximation such as those mentioned above.

2.3 Optimisations

Regardless of the chosen methods to compute non-linear ray tracing, optimisations will be required to allow the simulation to run in real time. Object space subdivision (Fujimoto et al. 1986, Glassner 1984) and bounding volume hierarchies (Glassner 1989; Kay and Kajiya 1986) can be used to divide the scene into discrete subsections. By doing this, the renderer is able to disregard large portions of the scene until a ray hits a bounding box, at which point the triangle-ray intersection detection can occur.

Weiskopf, Schafhitzel and Ertl, 2004 propose acceleration techniques such as early ray termination and adaptive ray integration. Early ray termination involves pruning and selectively discarding rays that are no longer needed to contribute to the final scene, reducing overall rendering times. Rays that hit an opaque surface, for example,

no longer require additional rays to be fired to determine the lighting at that position. As mentioned above, the h value when integrating a curve determines the length of each new additional ray segment. Adaptive ray integration aims to vary the h value when more accuracy is needed, such as when travelling around the gravity source.

Chapter 3

Methodology

As with many graphical research works, the experimental and visual nature of the subject requires a design-driven methodology. As such, this research will follow the Design Science Research methodology created by Ruben, with slight modification based on Hevner, 2004's guidelines.

Hevner R. et al suggest breaking down the core methodology into different guideline components that result in the creation of a meaningful artifact:

- Design as an Artifact

The primary output of design research is to create an artifact in the form of construct or technique. In the case of this paper, a short technology demonstration of the rendering technique is created.

- The Relevance of Problem

By creating an artifact to solve the design problem, it becomes relevant and applicable to the field, in this case computer graphics and physics simulation. Having an artifact to evaluate and demonstrate as a proof of concept becomes valuable for future works to begin from.

- The Design Evaluation

At each step in the creation of the artifact, the implementation's efficacy is rigorously tested through an evaluation method. Screenshots and performance measurements are taken periodically and assessed against the aims of the paper.

- Research Contribution

The contribution being made to the subject must be made clear. By creating an artifact, the contribution is easily accessible to those in the contributing field, and easily comparable to existing works.

- Design as a Search Process

Perhaps the most important guideline is to utilise different search processes to find a workable solution to the research problem. In this case, by experimenting in different avenues, unique and interesting visual effects can be created.

In the specific case of this research, a workflow similar to the flowchart below is used:

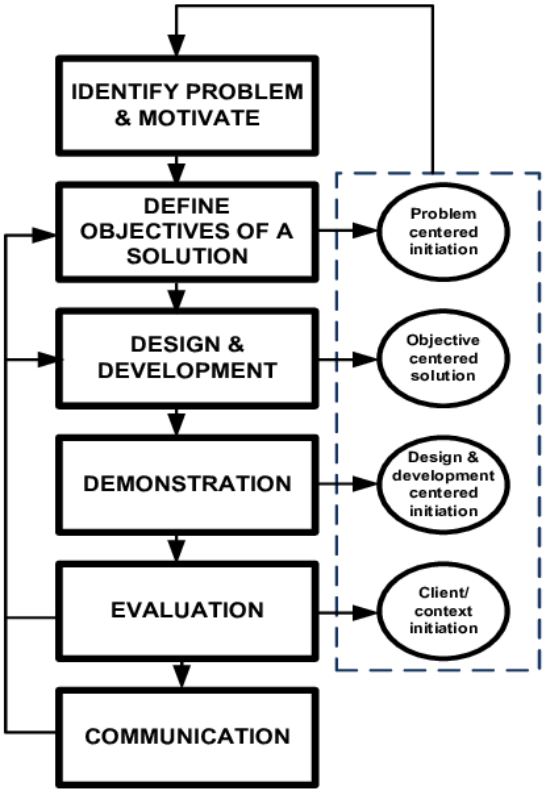


Figure 3.1: A design driven research workflow

By iterating over several designs throughout the entire development process, it be-

comes easy to make both large and small changes and adjustments to the artifact, bringing it more in line with the initial aims of the project.

In this instance, as there is no defined outcome for how the artifact should appear, specific landmark visuals are used as a guideline, e.g. a singularity and the associated gravitational lensing phenomenon.

Chapter 4

Design

4.1 Requirements

Given that the research being carried out in this paper is largely experimental in nature, the specific design requirements stem from the exploration of the ray tracing technology, probing its potential applications and practicality to different fields and disciplines.

In order to construct an implementation which could be evaluated against the initial aims of this paper, it was important to define the key characteristics from a selection of natural phenomena so that they could be reproduced.

One particular area of interest is the gravitational lensing phenomenon found when observing a large cluster of mass exerting a strong gravitational field.

Einstein's general theory of relativity, Einstein, 1905 describes how large concentrations of matter are able to distort the space-time around them. As light travels through the warped space-time, it's able to curve around the object, distorting, magnifying and often producing multiple mirage images of the original light emitter.

4.2 Design

4.2.1 Linear Ray Tracing

The fundamentals of linear ray tracing have been extensively covered in recent computer graphics literature, thus requiring no additional introduction. It may be per-

tinent however to present a discussion on how the various designs may affect the artifact performance when non-linearity is eventually introduced.

4.2.2 Introducing Non-Linearity

As outlined in the requirements, the gravitational lensing effect produced by a black hole is selected as one of the main visual phenomena to reproduce. Although many reproduction methods have been proposed in literature, perhaps the most interesting and versatile is the curved ray approach.

Gröller, 1995 introduced an integration based approach to non-linearity, in which light paths are represented as discreet, linear segments of a parametric curve. By dividing the ray into individual segments, each can be treated as if they are regular linear rays, vastly simplifying the required collision detection calculations.

The below equations describe the position and direction vectors for each sequential ray starting at the user view frustum.

$$\vec{X}_{n+1} = \vec{X}_n + h \cdot \frac{\vec{V}_n}{||\vec{V}_n||}$$

Figure 4.1: Where $\vec{X}_{n+1}$ represents the new segment position

$$\vec{V}_{n+1} = f(\vec{X}_n, \vec{V}_n)$$

Figure 4.2: Where $\vec{V}_{n+1}$ represents the new segment direction

The vectors are derived via Euler integration, where each new segment is iteratively expressed as position $\vec{X}_n$ and direction $\vec{V}_n$. The value h determines the length of each new line segment, and adjusting this will serve to increase or decrease the granularity of the simulation. As h decreases, the distance between each segment becomes shorter, allowing for the ray to more accurately travel around objects before colliding.

The exact value of h will be adjusted dynamically, rather than remaining a static value. This is because as with most high fidelity renderings, increased accuracy often comes at the cost of speed - this is to say that there is a positive correlation between

the amount of detail and accuracy in a scene against time taken to process each frame. With this in mind, and given that the integration is a simplified approximation of its real world counterpart, h will be allowed to adjust dynamically, decreasing in situations where more accuracy is needed such as around an object.

To calculate the new direction of the ray, one of several different functions will be used as f, depending on the desired output, covered later. Gröller, 1995 notes various different systems for producing non-linearity, of which the equations for some can be found below.

$$\vec{V}_{n+1} = \vec{V}_n - \frac{g \cdot M_g \cdot (\vec{X}_n - \vec{X}_g)}{||\vec{X}_n - \vec{X}_g||} \cdot d(||\vec{X}_n - \vec{X}_g||)$$

Figure 4.3: Gravity center / gravity line

$$\vec{V}_{n+1} = \begin{pmatrix} \sigma \cdot (y_n - x_n) \\ r \cdot x_n - y_n - x_n \cdot z_n \\ x_n \cdot y_n - b \cdot z_n \end{pmatrix}$$
$$\vec{X}_n = (x_n, y_n, z_n)$$

Figure 4.4: Lorenz system

$$\vec{V}_{n+1} = \begin{pmatrix} -y_n - z_n \\ x_n + a \cdot y_n \\ b + z_n \cdot (x_n - c) \end{pmatrix}$$
$$\vec{X}_n = (x_n, y_n, z_n)$$

Figure 4.5: Rossler system

Although any one of the above functions are suitable nonlinear representations, the gravity center / gravity line equation derived from Newton's law of universal gravitation most accurately fits the requirements and will be used.

The equation assumes a singular point object in space, with g gravitational force, M_g mass and X_g position. As demonstrated in reality, the gravitational force exerted by an object falls off exponentially with distance. Furthermore, in this equation, the fall off function d is represented by the equation below:

$$d(r) = (2\frac{r^3}{R^3} - 3\frac{r^2}{R^2} + 1)$$

Figure 4.6: Gravity falloff where R represents the radius of influence around the mass.

Having determined the mathematical representation, it is then a requirement to select a real-world candidate black hole to attempt to reproduce. Selecting a real-world black hole is a relatively trivial task, as any of the defining properties such as mass and radius of influence will result in a similar output.

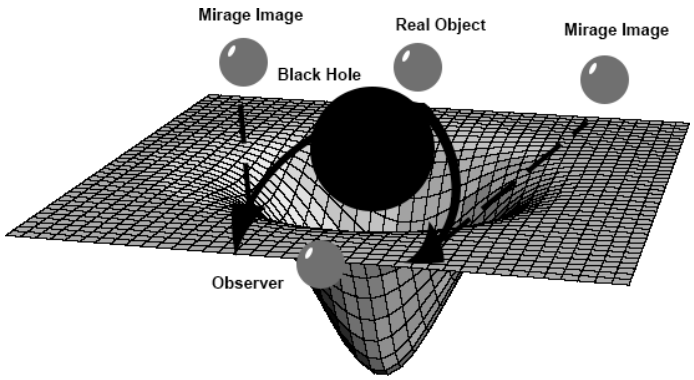


Figure 4.7: Embedding diagram illustrating gravitational lensing in 3 dimensions

For these reasons, an appropriate singularity candidate is Messier 87 with a mass equivalent to that of 20 times the Sun. The mass value is scaled down by several magnitudes in order to fit the scale of the rest of the rendering.

4.2.3 Optimisations

Ray tracing is regarded as one of the most computationally expensive rendering techniques currently available. To understand the reason for this, and why significant optimisation strategies are required to allow for the introduction of non-linearity, the problem must first be broken down into sub-components.

At any given display resolution, a ray should be fired from each individual pixel in order to create a fully rendered frame. A common resolution for modern displays is 1920 pixels in width by 1080 pixels in height, resulting in a total of 2,073,600 rays having to be cast every frame, each performing expensive collision checks and potentially generating subsequent bounce rays.

As such, the introduction of non-linearity is therefore expected to significantly lower the performance by several orders of magnitude, as each ray diverges from a single ray per pixel, into multiple new linear rays along the curve.

To reduce the impact of this performance decrease, several techniques are employed. The first is known as adaptive ray integration. As proposed by Weiskopf, Schafhitzel and Ertl, 2004, the length of each ray can be dynamically adjusted as and when needed to provide increased performance and quality. The figure below illustrates the data flow in which an adaptive process allows rays to be terminated early:

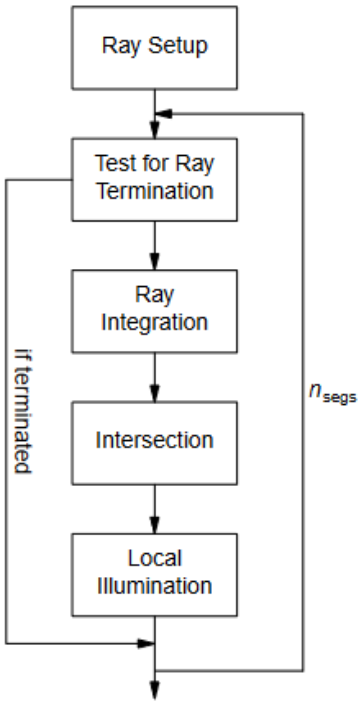


Figure 4.8: Early ray termination data flow

The total number of rays can be pruned significantly using the two methods described in the aforementioned work. The first method utilizes the fact that when a ray hits an opaque object, only a single ray is required and no subsequent bounce rays are necessary.

The second culling method removes any ray casts that pass outside of a designated area. After leaving the area in which all of the objects in the scene reside, the ray is no longer able intersect with them, and thus should be removed. Similarly, an octree-based subdivision data structure can be used in combination to further increase performance.

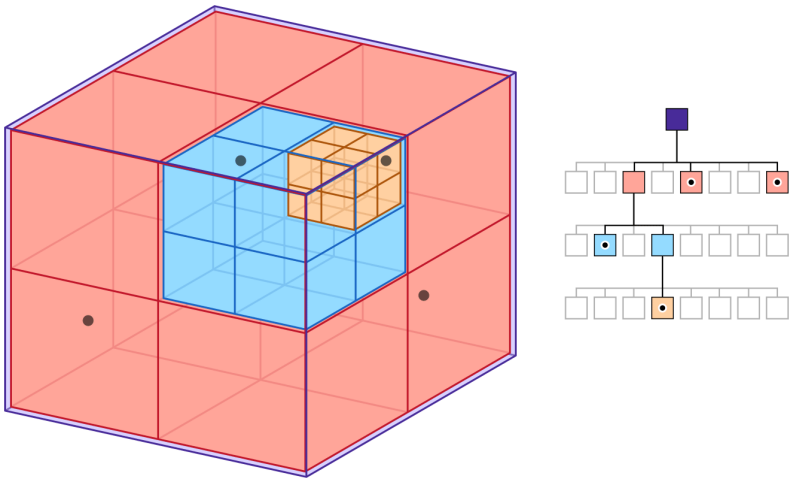


Figure 4.9: Octree visual representation

Shown above, an octree divides the available space into multiple subdivisions. If an objects in the scene were to reside within the orange portion, it would be a waste of computational power to continue to check for collisions if the ray entered any of the red portions.

This technique is especially prominent when using 3D models, instead of primitive shapes such as spheres. Generally, 3D models are comprised of several thousand triangles, of which each require expensive collision checks. By reducing these checks to limited regions of space, a significant performance increase is gained.

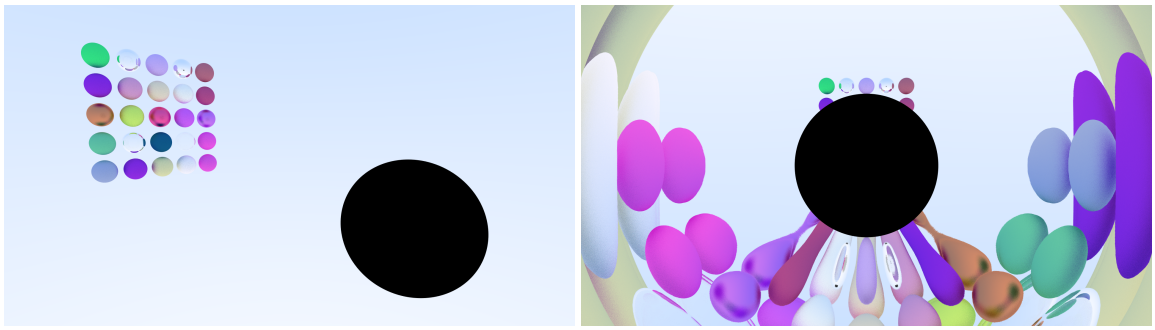
4.3 Implementation

The artifact implementation is programmed in C++, using the SDL library as a means to create a window with an OpenGL context. While no OpenGL functions are used, the ability to blit (draw) the ray traced pixel data to the display is still necessary.

To demonstrate the abilities of a fully globally illuminated scene, various materials have been implemented according to the BRDF standard:

- Metallic - Surface microfaceting to scatter incoming light and create a semi-reflective surface
- Dielectric - Glass-like material where light is able to pass through
- Emissive - Light emitting material

Shown below are various screen captures of the implementation, demonstrating several different linear and non-linear counterpart scenes.

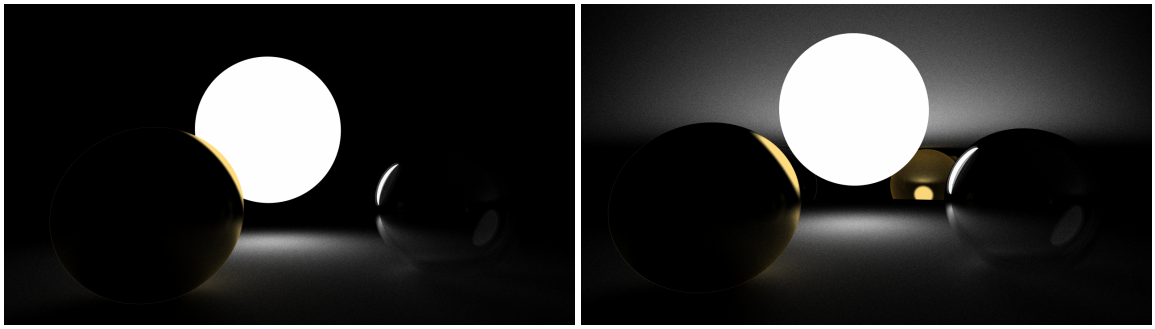


(a) Linear (b) Non-Linear

Figure 4.10: A comparison of a scene featuring multiple different materials

Shown on the left above is an array of spheres with randomized materials, properties and non-linearity disabled. On the right, the singularity is enabled and the gravitational lensing effect can be observed. Around the circumference of the singularity, the objects behind it appear stretched and distorted, as originally predicted.

It should also be noted that while this distortion occurs, reflections and other global illumination effects are still present in the rendering.



(a) Linear (b) Non-Linear

Figure 4.11: A dimly lit scene comparison

Similarly to the comparison above, two scenes are shown, both linear and non-linear. The scene is comprised of a floor, a metallic sphere, a dielectric sphere, and a large emissive sphere illuminating the scene.

As the linear scene is poorly lit, with the only source of light being the emissive material, the reflections on the floor and spheres appear relatively dim. Enabling non-linearity however creates a unique and interesting effect. A mirage image of the ground below the sphere is created, providing more opportunity for the rays to bounce around the scene and provide more light to the other spheres within it.

Regarding optimisations, a custom Vector3 class is implemented to facilitate large computational speed-ups by making use of SIMD (single instruction multiple data) intrinsics. SIMD intrinsics allow for the compiler to more efficiently compute float operations by treating individual values as vector calculations. Given AVX (advanced vector extensions) are supported by the processor, up to 8 operations can be completed per instruction.

The code listing below is an example of a Vector3 cross product function implemented via SIMD intrinsics.

```
Vector3 Vector3::Cross(const Vector3& v) const
{
  __m128 tmp0 = _mm_shuffle_ps(v.data, v.data, _MM_SHUFFLE(3, 0, 2,
   1));
  __m128 tmp1 = _mm_shuffle_ps(data, data, _MM_SHUFFLE(3, 0, 2, 1));
  tmp0 = _mm_mul_ps(tmp0, data);
  tmp1 = _mm_mul_ps(tmp1, v.data);
  __m128 tmp2 = _mm_sub_ps(tmp0, tmp1);

  return Vector3(_mm_shuffle_ps(tmp2, tmp2, _MM_SHUFFLE(3, 0, 2, 1))
   );
}
```

Listing 4.1: Vector3 cross product function implemented via SIMD intrinsics

The images below are screenshots taken throughout the development process of both the linear and non-linear ray tracing artifact:

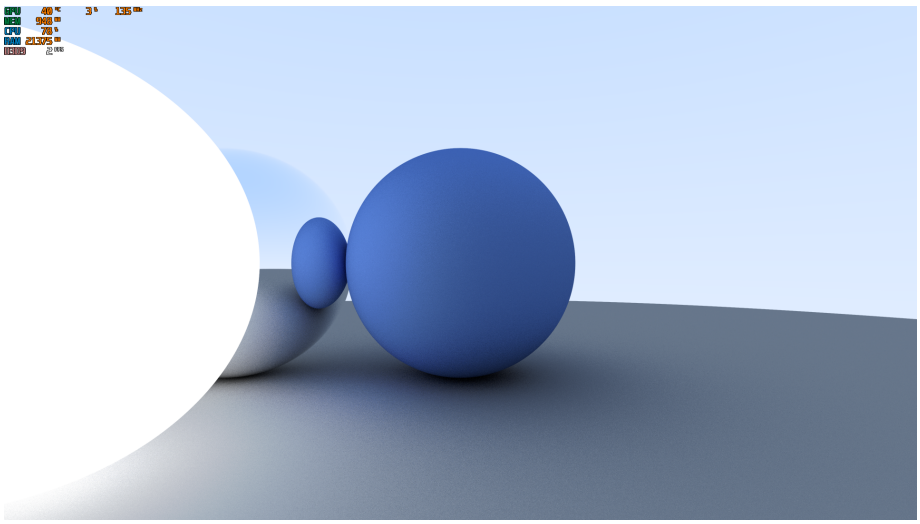


Figure 4.12: Noiseless image

The above image was created with the linear ray tracer, and demonstrates how smooth the final render can eventually become, given enough time for samples to collect and average. The scene consists of a large sphere for the floor, a diffuse blue sphere, a dielectic reflective sphere, and a white emissive sphere.

The blue sphere can be clearly seen reflected in the dielectic sphere, demonstrating how each ray is able to propagate and bounce with each material.

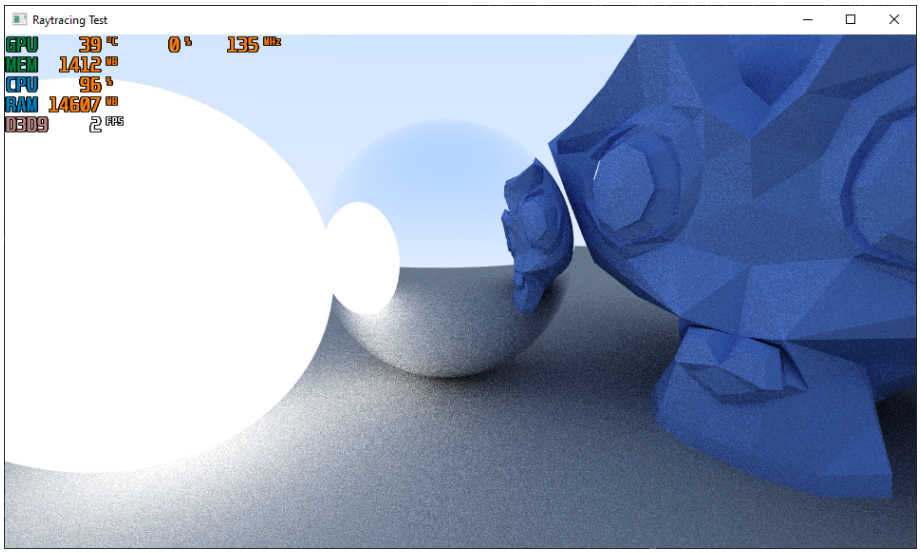


Figure 4.13: Suzanne reflected

Above, the image has a similar scene setup to the previous, however this image replaces the diffuse blue sphere with the popular 3D test model "suzanne". The model is also seen reflected in the dielectric sphere.

This image is of particular importance, as without the implementation of octrees and the other various optimisation techniques, it would not have been possible to render a model with such high triangle count at a reasonable framerate for viewing.

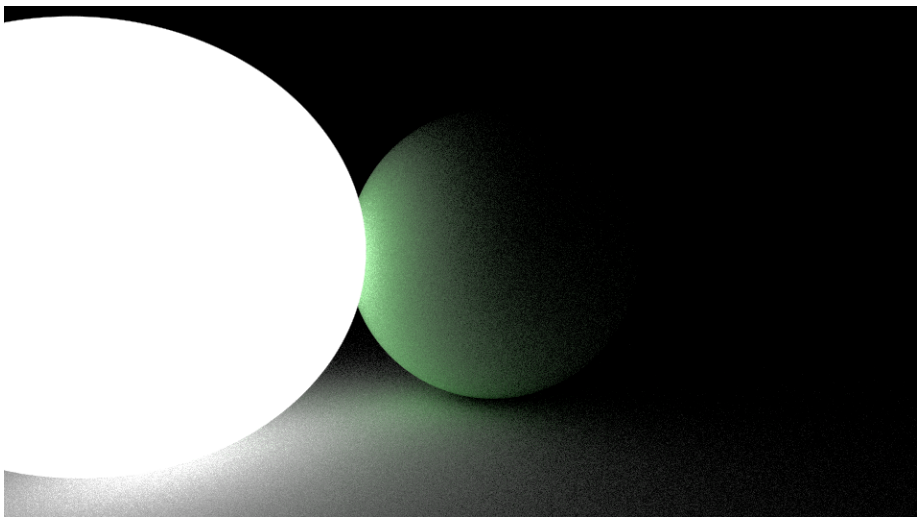


Figure 4.14: Dimly lit scene

Although a more simple scene setup, consisting of only an emissive light and a diffuse green sphere, the ambient lighting has been completely disabled, leaving only the scene lights to illuminate the world. By doing so, the reflected light "bleeding" from the green sphere to the ground becomes much more apparent.

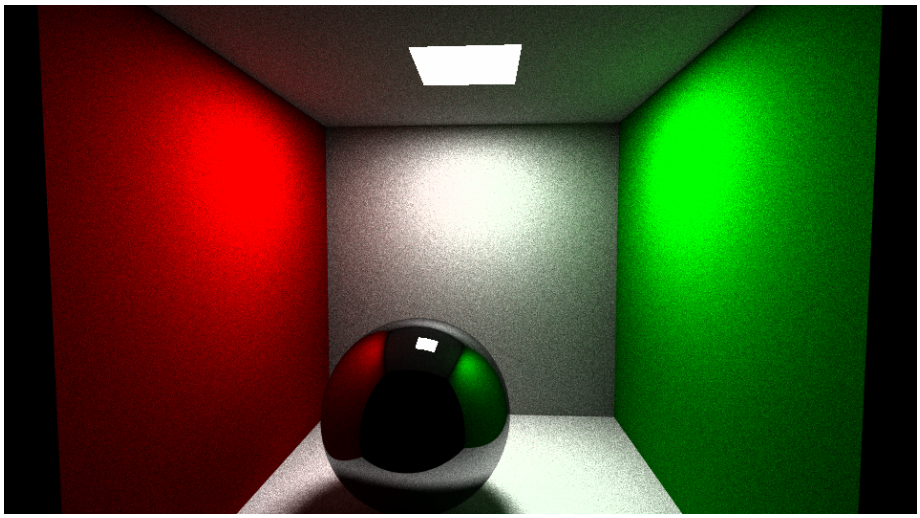


Figure 4.15: A standard Cornell box

Finally, the standard Cornell box is shown, demonstrating all of the materials and how the light reflects around the room.

Chapter 5

Evaluation

The research questions this paper sets out to answer, as defined in § 1.2, are primarily in relation to understanding the limitations of non-linear ray tracing. The knowledge gained can then be used create interesting and novel visual effects for use in both entertainment and scientific visualisation applications.

5.1 Evaluation Methodologies

Evaluations in literature, especially instances in which the work carried out is largely visual, are often performed as a comparative study. That is - several qualitative and quantitative metrics are obtained to measure the performance of the implementation, after which they can be compared to existing solutions.

As the work presented in this paper fills a niche category of computer graphics, very few comparative works are available to evaluate against. For this reason, both direct comparisons as well as empirical evidence gained via extrapolation are used in this evaluation.

5.1.1 Performance

Performance is measured using a number of different metrics. One of the first important performance metrics of any rendering system is usually the amount of frames pushed to the screen per second (FPS). In this case however, perhaps a more useful measurement to record is the time taken to create a single sample. With each

increasing sample taken, the visual fidelity of the final frame increases, and thus a higher number of samples per frame results in a higher quality rendering.

A number of factors may have an effect on the sample rate, such as the introduction of non-linearity, h-value, render resolution and CPU core count. Furthermore, while the performance is largely dependant on the composition and different materials that make up the scene, a static benchmark is used to keep the conditions largely the same throughout the tests for more accurate measurements.

The benchmark consists of various primitive objects such as spheres and cubes, as well as more complex 3D models including the popular "Suzanne" monkey, with a variety of materials applied including metallic, dielectric and emissive.

Below are the results of the benchmark, with gradually decreasing resolution for both linear and non-linear systems.

Table 5.1: Linear Benchmark

Resolution	Sample Time (ms)	CPU
1080p	135	i7-9700K @ 5GHz
720p	60	i7-9700K @ 5GHz
480	27	i7-9700K @ 5GHz
240	15	i7-9700K @ 5GHz

Table 5.2: Non-linear Benchmark

Resolution	Sample Time (ms)	CPU
1080p	992	i7-9700K @ 5GHz
720p	444	i7-9700K @ 5GHz
480	197	i7-9700K @ 5GHz
240	41	i7-9700K @ 5GHz

As mentioned previously, the h value of the curve defines the distance of each linear segment in the chain. In essence, for each ray in a linear scene, between 5 and 10

are required for a non-linear scene, thus the required processing power is expected to rise exponentially.

The results above, however, tell a different story. Although an exponential performance decrease was expected in the non-linear benchmark, a linear increase similar to that of the linear benchmark was instead observed.

The reason for this is likely due to the metric being recorded. Although the sample time is much higher for the non-linear benchmark, changing the resolution reduces the number of pixels by the same ratio for both benchmarks, and thus a linear change is observed for both. To gain a more accurate benchmark, a change in the complexity of the scene could have been created.

5.1.2 Accuracy

Measuring how accurate each simulation is with regards to the real world becomes an increasingly difficult task, as each non-linear visualisation deviates further from the realm of possibility in which we are able to observe and measure such phenomena.

Instead, the individual components that compose the system are broken down and analysed with regards to how they are constructed for accuracy. For example, while the h value mentioned above does impact sample time, it also primarily affects the accuracy of the simulation.

If the h value remained constant for each ray as they traverse scene, detail would be lost if the linear ray was too large in length to turn around corners. Instead, a technique known as Adaptive Ray Integration is used to dynamically adjust the h value according to what is in the scene, helping to maintain a balance of performance and accuracy.

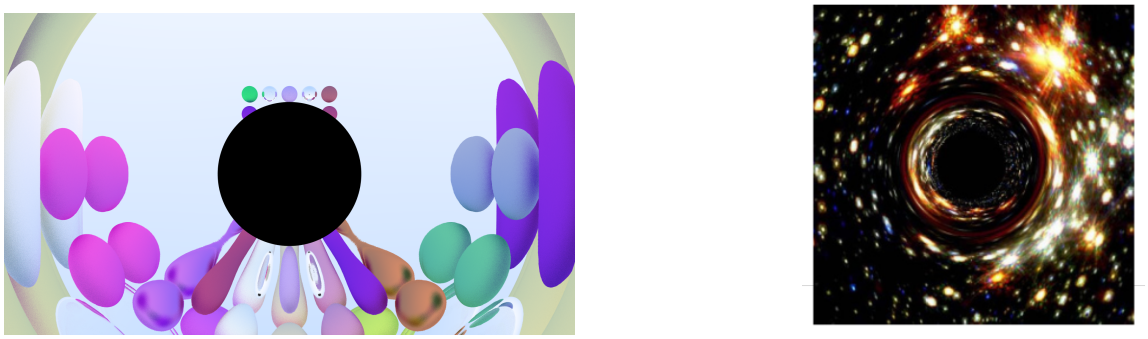
Another large factor in ensuring the simulation is accurate is the precision of the data structures that are used to hold the numerical values. Given that the scale of real world measurements such as black hole masses are orders of magnitude greater than a 64 bit floating point value can hold, they must be scaled down to fit. If scaled

down too much however, accuracy is lost and the reproduction is no longer accurate to the real world. For this reason, double precision data types are used.

5.1.3 Image Quality

The visual clarity of the scene is ultimately dependent on several factors. For instance, as the number of accumulated samples increases, the amount of noise in the image will gradually decrease.

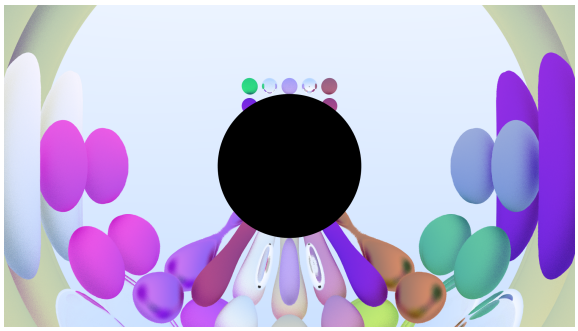
It is therefore useful to perform a side-by-side visual comparison using the work produced in this paper against existing methods, to determine if the effects are consistent with what should be expected:



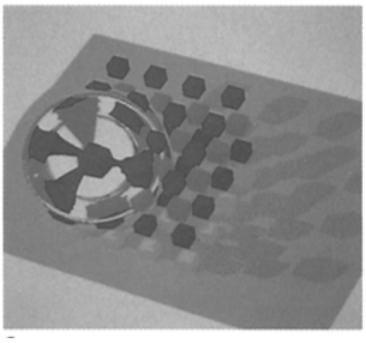
(a) This paper (b) Satoh's implementation

Figure 5.1: A comparison of this paper's work and Satoh, n.d.

The above comparison shows a visualisation created in this paper, alongside the work of Satoh, n.d. Although the starfield in the background of Satoh's work is not present in this work due to a lack of texture mapping, the gravitational distortion effect can be clearly seen in the ring around the outer edges of both images. Additionally, Satoh's work does not run in real time, and requires a significant amount of time to render.



(a) This paper



(b) Groller's implementation

Figure 5.2: A comparison of this paper's work and Gröller, 1995

While Gröller, 1995 provides an accurate representation of the gravitational distortion effect, the rendering techniques are relatively outdated. The final render is low resolution, and lacks any materials or global illumination. Additionally, the rendering is slightly noisy, leading to the conclusion that no de-noising algorithms were applied.

5.1.4 Practicality

Having discussed the various aspects of the rendering system and performed a comparative study, it then becomes possible to discuss its practicality and usefulness in different fields.

In relation to video game development, a rendering such as this may prove useful in certain specialized circumstances, however the substantial performance impact may serve dissuade any potential widespread adoption.

Inversely, as a large number of scientific research fields that deal with simulations of this nature often don't require a large amount of speed, a rendering such as this is far more beneficial.

Chapter 6

Conclusions

Having evaluated a few of the effects produced by the outlined method in this paper, conclusions can be drawn about whether the initial research aims were met, as well as its usefulness and applicability to different fields.

The first aim was to better understand the limitations of a non-linear system, and gain a deeper understanding of how to better design a system around them. With this in mind, existing literature and implementations were reviewed, noting the advantages and disadvantages that each system exhibits. The review found that while the Monte Carlo method produces much higher quality results, a de-noising algorithm is a necessary requirement to create a cohesive final render.

Using the literature as reference, an implementation was created to satisfy the second aim of demonstrating possible visual effects that can be achieved. An evaluation of these effects found that when compared to older as well state of the art techniques, the visualisation held up relatively well, creating a convincing non-linear scene.

Finally, there was a discussion about the applicability of such a system in both a video game development and scientific research context.

6.0.1 Further Work

Although numerous optimisations were employed to allow the implementation to run smoothly, little work was done in the area of denoising. Various techniques exist for this, ranging from DLSS 2.0, Nvidia, 2020b to Spatiotemporal Variance-Guided Filtering, Schied et al., 2017 and more recently RTX, Nvidia, 2020a. While varying in

implementation, these techniques operate under a similar premise. A small number of samples are taken each frame, creating a very broken and noisy image. After this, the image is passed through various reconstruction filters to make the image whole, saving valuable computational resources.

Furthermore, as the entire scene is fully ray traced and globally illuminated, perhaps an interesting addition would be to convert the implementation to use wavelengths to represent colour, as an alternative to simple RGB values. Such a conversion would allow for interesting visual effects, including shifting the visible parts of the electromagnetic spectrum.

Finally, the addition of a visible accretion disc, that is - the hot stream of matter which flows around a black hole, would have helped to put the singularity into better context.

Acknowledgements

I would like to thank my supervisor Mr. Phil Carlisle for the support and supervision of this dissertation. Additionally, many thanks to Michal Manda for his continued assistance and contributions to the numerous difficult programming challenges.

Thank you to Dr Phil Sutton for helping to explain some of the more complicated physics concepts surrounding black holes.

References

Appel, A (1968). 'Some techniques for shading machine renderings of solids.' In: *spring joint computer conference* (cit. on p. 7).

Bitterli, B et al. (2016). 'Nonlinearly weighted first-order regression for denoising Monte Carlo renderings'. In: *In Computer Graphics Forum (Vol. 35, No. 4, pp. 107-117).* (cit. on p. 11).

Cook, R.L., T Porter and L Carpenter (1984). 'Distributed ray tracing.' In: *In Proceedings of the 11th annual conference on Computer graphics and interactive techniques (pp. 137-145)* (cit. on p. 10).

Einstein, A (1905). 'Special Theory of Relativity'. In: (cit. on p. 20).

Glassner (1984). 'Space subdivision for fast ray tracing.' In: *IEEE Computer Graphics and applications* (cit. on p. 8).

– (1989). 'An introduction to ray tracing.' In: *Elsevier* (cit. on p. 7).

Gröller, E (1995). 'Nonlinear ray tracing: Visualizing strange worlds'. In: *The Visual Computer*, 11(5), pp.263–274. (Cit. on pp. 13, 21, 22, 35).

Hevner, A (2004). 'Design Science in Information Systems Research'. In: (cit. on p. 17).

iD (1992). 'Wolfenstein 3D'. In: (cit. on p. 7).

Kajiya and J.T (1986). 'The rendering equation'. In: *In Proceedings of the 13th annual conference on Computer graphics and interactive techniques (pp. 143-150)* (cit. on p. 10).

Metropolis, N and S Ulam (1949). 'The monte carlo method'. In: *Journal of the American statistical association, 44(247), pp.335-341* (cit. on p. 10).

Nvidia (2020a). *GeForce RTX 3090*. URL: `https://www.nvidia.com/en-gb/geforce/graphics-cards/30-series/rtx-3090/` (cit. on pp. 4, 36).

– (2020b). 'NVIDIA DLSS 2.0: A Big Leap In AI Rendering'. In: (cit. on p. 36).

Roberts, E (1997). *cs.stanford.edu* (cit. on p. 10).

Satoh, T (n.d.). 'Symplectic Ray Tracing: A new frontier in non-linear ray tracing'. In: () (cit. on p. 34).

Schied, C et al. (2017). 'Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination'. In: (cit. on p. 36).

Vandevenne (2020). 'Raycasting'. In: (cit. on p. 7).

Weiskopf, D (2000). 'Four-dimensional non-linear ray tracing as a visualization tool for gravitational physics.' In: *In Proceedings Visualization 2000. VIS 2000 (Cat. No. 00CH37145) (pp. 445-448). IEEE.* (cit. on pp. 14, 15).

Weiskopf, D, T Schafhitzel and T Ertl (2004). 'GPU-based nonlinear ray tracing.' In: *In Computer graphics forum (Vol. 23, No. 3, pp. 625-633). Oxford, UK and Boston, USA: Blackwell Publishing, Inc.* (cit. on pp. 15, 24).