Realtime Approximation of Light in Curved Spacetimes

Wyatt Marvil

4/24/2023

Abstract

In this paper, we present an approach to approximating light in curved spacetimes in real-time with GPU raytracing. Our method allows for the simulation of complex gravitational lensing effects and other phenomena in curved spacetimes in real-time, making it ideal for use in interactive applications such as video games and virtual reality. We achieve this by using both explicit and implicit numerical integration techniques to efficiently trace rays through the curved spacetime, and by leveraging the parallel processing capabilities of modern GPUs to accelerate the computation. Our approach produces high-quality, but physically inaccurate approximations of light curving effects, while also being scalable and flexible enough to handle a wide range of scenarios and configurations. We demonstrate the effectiveness of our method through several examples, and show that it creates sufficiently convincing relativistic behavior. Our work has the potential to enable new forms of interactive and immersive experiences that leverage the rich and fascinating geometry of curved spacetimes.

1 Introduction

The behavior of light in the presence of gravity has long been a subject of fascination and study in the field of theoretical physics. In the theory of general relativity, the curvature of spacetime affects the path of light rays, leading to phenomena such as gravitational lensing and black holes. Raytracing is a powerful technique used in computer graphics to render realistic images of three-dimensional objects. In the context of general relativity, raytracing can be used to simulate the behavior of light in the presence of massive objects such as black holes. In this paper, we will discuss the implementation of a Vulkan compute raytracer that is capable of rendering approximations of relativistic effects.

We will start by exploring the literature around rendering relativistic phenomena in Section 2. In Section 3, we will introduce the architecutre of our GPU raytracer, and how it allows us to implement arbitrary metrics. Section 4 will explain the implementation of a Newtonian approximation for general relativity, while section 5 will cover accurate relativistic models such as the Schwarzschild and Kerr metrics. Sec-

tion 6 will be a brief aside on rendering celestial accretion disks and radiative heat transfer. We will close the paper with our results, conclusions, and avenues for future work.

2 Related Work

The simulation of light in curved spacetimes has been an alluring topic across several fields, including theoretical physics, cinema, and interactive media. Several methods have been proposed to model the behavior of light in general relativity, ranging from analytical solutions to numerical techniques.

One commonly used approach is based on the use of geodesics, which are the curves followed by particles (including light) in the presence of gravity. The geodesic equation can be solved analytically for certain spacetimes, such as the Schwarzschild metric, which describes the spacetime around a non-rotating black hole. However, for more complex spacetimes, analytical solutions are often not available, and numerical methods must be used.

Numerical methods for simulating the behavior of light in curved spacetimes include modified ray-tracing algorithms, which simulate the path of light rays through a curved spacetime. One example is the work of [9], using ray bundles to approximate the Kerr metric. These methods are computationally intensive and are typically used for offline simulations. Realtime simulations usually require faster methods, and several approaches have been proposed to address this issue.

One such approach is based on the use of precomputed lookup tables, which store the paths of light rays for a given set of initial conditions. These lookup tables can be used to approximate the path of light rays in real-time, but they are limited to a specific set of initial conditions and cannot account for changes in the spacetime geometry.

Another approach is based on the use of numerical integration techniques, such as the Runge-Kutta method, to solve the geodesic equation in real-time. However, these methods can still be computationally expensive, especially for spacetimes with high curvature.

Recently, machine learning techniques have been applied to the problem of simulating light in curved spacetimes. One approach is based on the use of neural networks to approximate the path of light rays, which can be trained on a set of precomputed paths. Another approach is based on the use of reinforcement learning, where an agent learns to navigate through a curved spacetime by maximizing a reward function.

Other work has also been done to move offline methods to realtime with GPU acceleration [3]. There is also the work of [1] to create massive, life-scale interactive simulations for consumer exploration of cosmology.

We based our method on numerically solving discretized geodesics with a metric-agnostic integration model. We can toggle between explicit euler integration and an implicit fourth order Runge-Kutta method, and we can also utilize any spacetime metric that operates in three spatial dimensions (even non-physical solutions, such as the white hole in [10])

3 Raytracer Architecture

To build our GPU raytracer, we began with the compute raytracing sample from Sascha Willems' Vulkan examples repository [12]. On top of this sample, we added several features to support our project, including a free-flight camera, a dynamically updated Shader Storage Buffer Object (SSBO) for blackhole objects, and an additional cubemap texture sampler for integrating skyboxes and backgrounds. This baseline raytracer performs implicit intersection tests against spheres and planes, and includes hard shadows and multibounce reflections.

3.1 Ray Marching

The first step in supporting relativistic behavior in our renderer is to move to a raymarch-

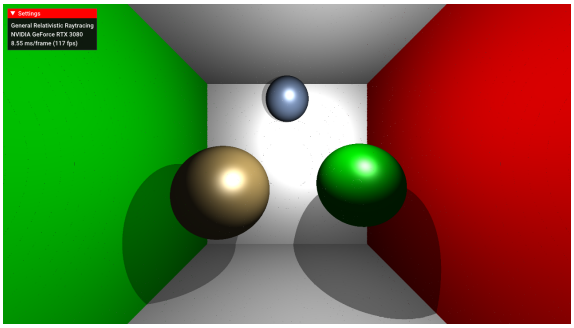

Figure 1: Raymarched scene with trivial BendPath, 117 FPS

ing scheme instead of the standard intersection traces.

Algorithm 1 Ray Marching Intersections

```
1: procedure INTERSECT(Ray r)
2:     id ← −1
3:     for i < iterations do
4:         id, t ← Intersect(r)
5:         if t < eps then
6:             break
7:         r ← BendPath(r, dt)
8:     return id
```

On each iteration, we test the current ray for intersections. t is the distance to the nearest intersection in world units, and once it falls below a certain value, we assume a final intersection. Next, we calculate the ray for the next iteration by calling a BendPath function, which will return a position and direction exactly one timestep away, accounting for the curvature of spacetime as dictated by the current spacetime metric.

The trivial BendPath function simply returns the ray, yielding a conventional raytracer.

Algorithm 2 Bend Path

```
1: procedure BENDPATH(Ray r, float dt)
2:     return r
```

Rays that miss the scene entirely can render a static background color or sample from an HDR cubemap. If the ray falls within a Schwarzshild radius, it is physically unable to escape the gravity of a nearby blackhole, and we can terminate the ray marching loop early and return an ID denoting black.

4 Newtonian Approximation

The Newtonian approximation for general relativity is an unphysical model that assumes the photon has a mass of 1kg. However, it is much simpler to implement than the more complicated relativistic models we will discuss later, and to the human eye they are mostly indistinguishable.

Algorithm 3 Newtonian Bend Path

```
1: procedure BENDPATHNEWT(Ray r, float dt)
2:     newPos ← vec3(0.0)
3:     for b in blackholes do
4:         r ← b.pos - r.pos
5:         r2 ← dot(r, r)
6:         rSW ← 2 ∗ b.mass
7:         if r2 < rSW ∗ rSW then
8:             break
9:         g ← b.mass/r2
10:        newPos ← newPos + g ∗ norm(r) ∗ dt
11:    r.dir ← norm(pos − r.pos)
12:    r.pos = pos
13:    return r
```

We can compute the Schwarzschild radius from the mass of the blackhole. This can be used to terminate early, since any position that falls within this distance of a massive object will be unable to escape.

$$rSW = 2 * G * M/c^2$$

For numerical precision, we do not use the real world values for Newton's gravitational constant ($6.674310^{-11} m^3 kg^{-1} s^{-2}$) or the speed of light ($299792458 m/s$). Instead, we set assume both of them to be 1.0.

Next, we modify the ray position, and accumulate gravitational force across the scene. We use Newton's law of gravitation, taking m_1 to be 1kg.

$$F_g = G * \frac{m_1 m_2}{r^2}$$

After applying the accumulated force to the new position, scaled by the simulation timestep, we can compute the new direction by normalizing the difference between the new and old positions.

5 General Relativity

5.1 Null Geodesics

Unfortunately, the Newtonian method is only an approximation. In order for our model to be physically accurate, our photons need to travel along null geodesics. A null geodesic is a path in spacetime that is followed by a massless particle, such as a photon. In other words, it is the trajectory of light or other electromagnetic radiation in a gravitational field.

Mathematically, a null geodesic is defined as a curve in spacetime that is parameterized by an affine parameter and satisfies the geodesic equation, which describes the path of a free particle moving under the influence of gravity. The affine parameter is a quantity that varies linearly along the curve and is used to specify the position of the particle along the curve.

One important property of null geodesics is that they always travel at the speed of light, since they are followed by massless particles. This means that their worldlines are always tangent to the light cone at each point along the curve.

In general, the approach to finding a null geodesic involves solving the geodesic equation, which is a set of second-order differential equations that describe the motion of a particle in spacetime, and it can be written as follows:

$$\frac{d^2 x^\mu}{d\lambda^2} + \Gamma^\mu_{\alpha\beta} \frac{dx^\alpha}{d\lambda} \frac{dx^\beta}{d\lambda} = 0 \qquad (1)$$

where x^μ is a set of four coordinates that describe the position of the particle in spacetime, λ is an affine parameter that parameterizes the curve, and $\Gamma^\mu_{\alpha\beta}$ are the Christoffel symbols, which encode the curvature of the spacetime.

To solve the geodesic equation, one typically starts with the metric tensor of the spacetime, which specifies the geometry of the spacetime. The metric tensor can be used to calculate the Christoffel symbols, which in turn can be used to write down the geodesic equation.

In general, the geodesic equation is a difficult set of equations to solve analytically, and solutions may only be found for special cases of spacetimes with particular symmetries, such as the Schwarzschild metric for a non-rotating, spherically symmetric black hole.

In this paper, we solve the geodesic equation numerically for arbitrary 3+1 spacetimes using the fourth order Runge-Kutta method.

5.2 Schwarzschild Metric

The Schwarzschild Metric is typically written in terms of (t, r, θ, ϕ).

$$ds^2 = -\left(1 - \frac{2GM}{c^2 r}\right) c^2 dt^2 + \frac{dr^2}{\left(1 - \frac{2GM}{c^2 r}\right)} + r^2 sin^2\theta\phi^2$$

The metric tensor $g_{\mu\nu}$ is written as

$$\begin{bmatrix} -\left(1 - \frac{2GM}{c^2 r}\right) c^2 & 0 & 0 & 0 \\ 0 & \left(1 - \frac{2GM}{c^2 r}\right)^{-1} & 0 & 0 \\ 0 & 0 & r^2 & 0 \\ 0 & 0 & 0 & r^2 \sin^2\theta \end{bmatrix}$$

In order to begin solving the geodesic equation with this metric, we need to calculate the non-zero Christoffel symbols.

$$\Gamma^t_{tr} = \Gamma^t_{rt} = \frac{GM}{c^2 r^2}\left(1 - \frac{2GM}{c^2 r}\right)$$

$$\Gamma^r_{tt} = \frac{GM}{c^2 r^2}\left(1 - \frac{2GM}{c^2 r}\right)$$

$$\Gamma^r_{rr} = -\frac{GM}{c^2 r^2}\left(1 - \frac{2GM}{c^2 r}\right)^{-1}$$

$$\Gamma^\theta_{\theta r} = \Gamma^\theta_{r\theta} = \frac{1}{r}$$

$$\Gamma^\phi_{\phi r} = \Gamma^\phi_{r\phi} = \frac{1}{r}\sin^2\theta$$

$$\Gamma^\phi_{\theta\phi} = \Gamma^\phi_{\phi\theta} = \cot\theta$$

We can plug these into the differential system of equations given by the geodesic (1). Now, taking our ray as the initial value, we can solve the geodesic equation as an initial-value problem differential equation with a method of your choice. For simplicity, we chose the widely used Runge-Kutta method.

Algorithm 4 Fourth-order Runge-Kutta method

procedure RUNGEKUTTA(Initial values y_0 and t_0, step size h, number of steps N)
  $t = t_0$
  $y = y_0$
  for $n = 1$ ¡ N **do**
    $k_1 = f(t, y)$
    $k_2 = f(t + \frac{h}{2}, y + \frac{h}{2}k_1)$
    $k_3 = f(t + \frac{h}{2}, y + \frac{h}{2}k_2)$
    $k_4 = f(t + h, y + hk_3)$
    $y = y + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$
    $t = t + h$

5.3 Kerr Metric

The process is similar (but more difficult) for the Kerr metric. It is left as an exercise to the reader.

6 Accretion Disk

Accretion disks, though not relativistic objects in and of themselves, possess an aesthetically pleasing quality due to their proximity to massive celestial bodies. These disks are essentially rotating disks of matter subject to the gravitational force exerted by the corresponding massive object.

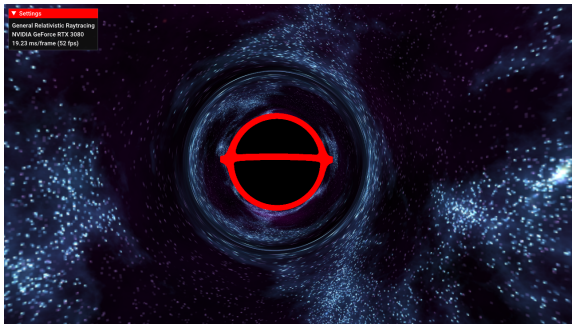

Figure 2: A single blackhole with a thin accretion disk (1.25x SW Radius), 52 FPS

Rendering engines employ several techniques to produce accurate representations of accretion disks. One method is to precompute the geometry of a disk/washer and instance that geometry throughout a scene as desired, by sending triangle buffers to the GPU. Another approach is similar to how we render our spheres and planes, where we can send an SSBO with an array of data detailing individual disk positions and radii. We can then test intersections against an implicit disk equation to get high precision mathematical intersections. The third, and simplest approach is to create a signed distance field (SDF) for the disk, and test against that during our ray marching.

We have chosen the third approach, utilizing an SDF generated at the center of each black hole in the scene. The SDF is an intersection of a sphere and two inward-facing planes, with the disk's radius computed based on the black hole's mass, scaled by a factor of the Schwarzchild radius. It is important to note, however, that the lower precision of this technique results in visible aliasing along the edges of the disk, while spatially relating one portion of the disk to another is challenging, limiting its artistic potential.

6.1 Radiative Transfer

The most accurate method of rendering accretion disks is known as radiative transfer. Researchers use radiative transfer calculations to model how the light emitted by the disk is absorbed, scattered, and emitted as it travels through the disk. This involves solving a set of equations that describe the transfer of radiation through the medium of the disk, taking into account the properties of the matter and the geometry of the disk.

One approach to modeling accretion disks is to use Monte Carlo methods, in which individual photons are traced through the disk, with their properties (such as wavelength and polarization) changing as they interact with the matter in the disk. Another approach is to use numerical methods to solve the radiative transfer equations directly. Either of these techniques could be applied to our baseline generalistic renderer, and would be a good topic for later efforts.

7 Results

All of our renders currently use the Newtonian Approximation scheme. We have implemented a Schwarzschild metric with implicit 4th order Runge-Kutta integration, but it currently does not give accurate or satisfying results. We have several examples, including a micro blackhole indoors, an Einstein Ring in deep space, multiple interaction between massive bodies, and an intersting case of gravitational lensing.

8 Conclusion

In this paper, we have presented a method for approximating the behavior of light in curved spacetimes using compute raytracing. Our method allows for realtime rendering of complex scenes with curved spacetimes, which was previously a computationally expensive task. By utilizing compute raytracing, we were able to achieve significant speedup compared to previous methods that relied on traditional CPU-based raytracing.

Our approach is based on approximating the geodesics of light rays in curved spacetimes using a series of discrete steps, and integrating the cumulative displacement due to spacetime curvature. To demonstrate the effectiveness of our method, we presented several examples of rendering scenes with curved spacetimes, including several blackholes and gravitational lensing. In each case, our approach allowed for fast and convincing rendering of these complex scenes in realtime.

In the future, we would like to continue pursuing more physically accurate spacetime metrics for simulating and rendering maximally realistic gravitational phenomena, ideally indistinguishable from real world examples. There are also some artifacts where geometry close to massive objects can sometimes have bands of missing color. Issues like this are extremely difficult to debug on the GPU, so in the future we would also like to keep a CPU implementation up to date, and use it as a testing ground to get the complex math correct in a more developer-friendly environement.

There is also opportunity for an adaptive sampling method, which would allows us to focus computational resources where they are needed most, resulting in improved accuracy while maintaining high performance. There is opportunity for this in the raymarching scheme, where we can fallback to conventional traced intersections over large swaths of space that are not occupied by massive objects. Perhaps we could utilize an SDF of the massive objects with a cheaply computed inverse square fall off to create small regions where rays must be marched in discrete chunks, while elsewhere they can be used for continuous implicit intersections.

9 Acknowledgements

We would like to give special thanks to Barbara Cutler for a semester of support and encouragement, and being a beacon of computer graphics

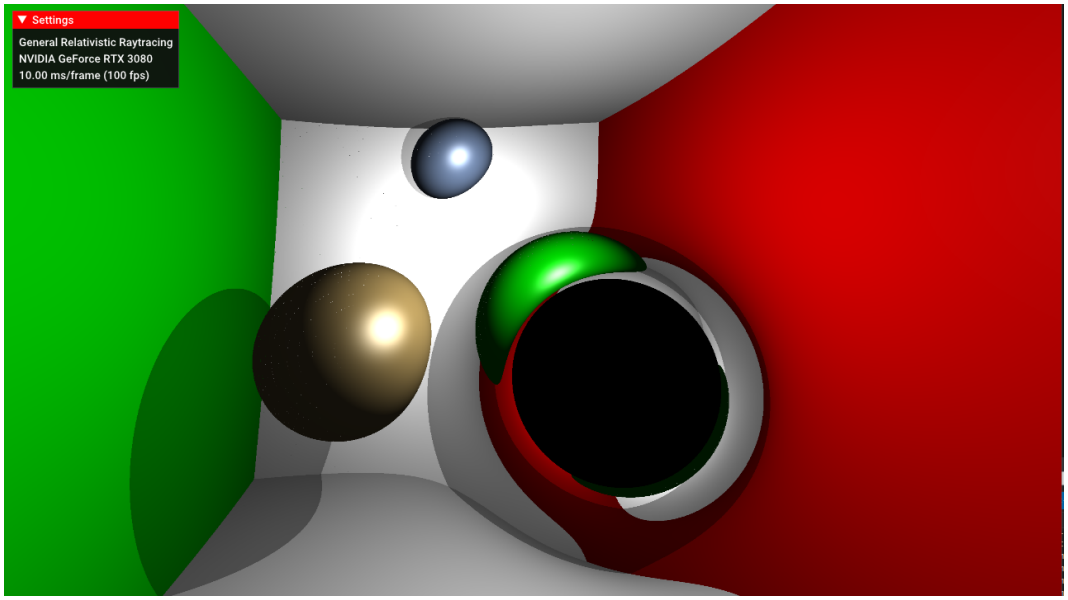

Figure 3: A blackhole threatening our beloved Cornell Box. (100 FPS)

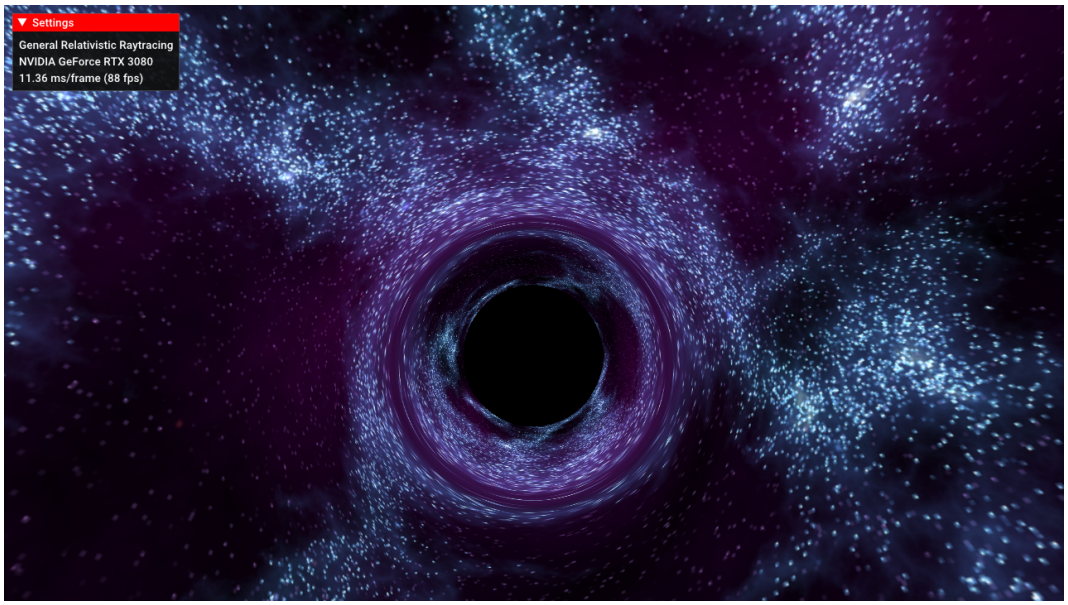

Figure 4: A blackhole against a nebulous backdrop. Demonstrating the "Einstein Ring" effect. (88 FPS)

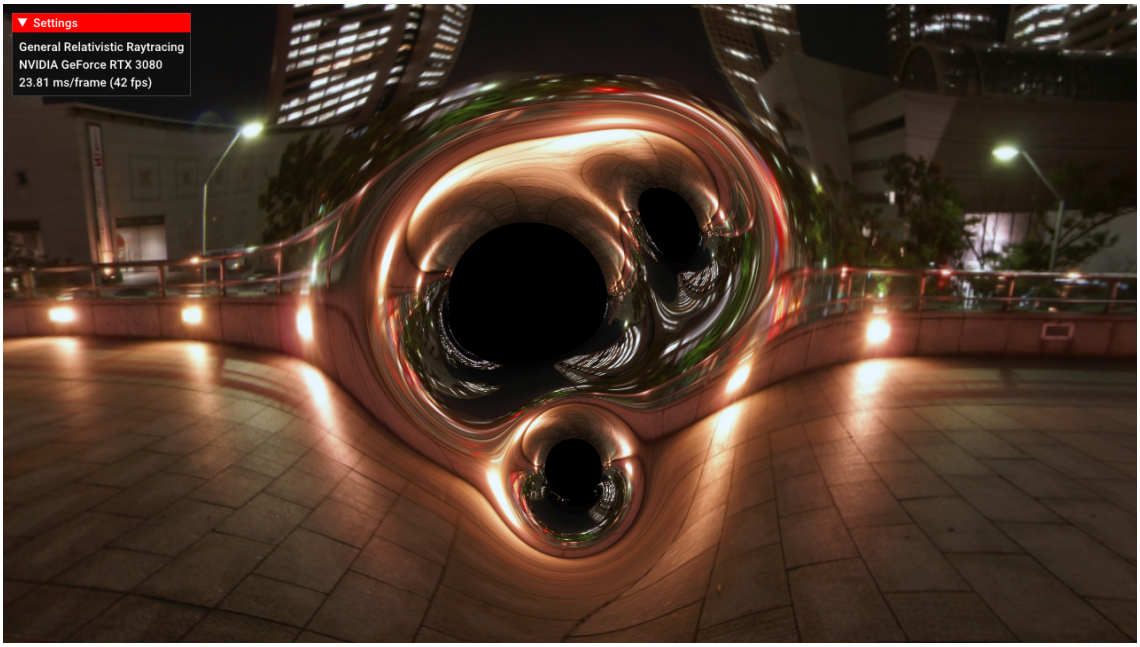

Figure 5: Multiple interaction. 3 blackholes in close proximity, bending an HDR cubemap. (42 FPS)

knowledge in the RPI community, and also to Ronan Tegerdine for being a living and breathing rubber duck.

References

[1] General relativity 1: Kerr black holes. 2022.

[2] Kumar Ayush Avirup Mandal and Parag Chaudhuri. Non-linear monte carlo ray tracing for visualizing warped spacetime.

[3] Chi-kwan Chan. Gpu-accelerated general relativistic ray tracing for simulating black hole images. YouTube video, 2020.

[4] Jordy Davelaar and Zoltán Haiman. Self-lensing flares from black hole binaries: General-relativistic ray tracing of black hole binaries. *Physical Review*, 2022.

[5] A. Y. Chen F. Bacchini B. Ripperda and L. Sironi. Generalized, energy-conserving numerical simulations of particles in general relativity. *The Astrophysical Journal*, 2018.

[6] E Gourgoulhon F H Vincent, T Paumard and G Perrin. Gyoto: a new general relativistic ray-tracing code. 2011.

[7] Federico Carrasco Joaquín Pelle, Oscar Reula and Carlos Bederian. Skylight: a new code for general-relativistic ray tracing and radiative transfer in arbitrary spacetimes. 2022.

[8] Geraint F. Lewis Chris J. Fluke Madhura Killedar, Paul D. Lasky. Gravitational lensing with three-dimensional ray tracing. 2011.

[9] Paul Franklin Kip S. Thorne Oliver James, Eugenie von Tunzelmann. Gravitational lensing by spinning black holes in astrophysics, and in the movie interstellar. *Classical and Quantum Gravity*, 2015.

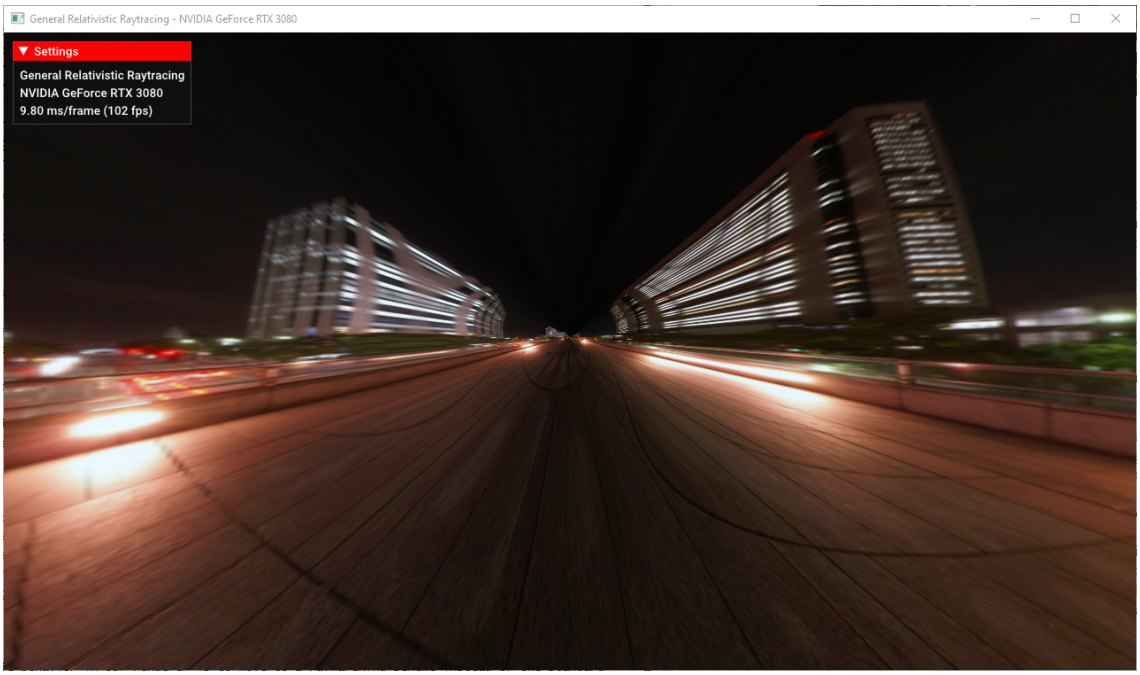

Figure 6: The whitehole acts inversely to a blackhole, pushing space away at the speed of light. (102 FPS)

Figure 7: Gravitational lensing. The red sphere is fully occluded by the green sphere, but it appears as though the red sphere is larger because the green sphere bends light around itself. (36 FPS)

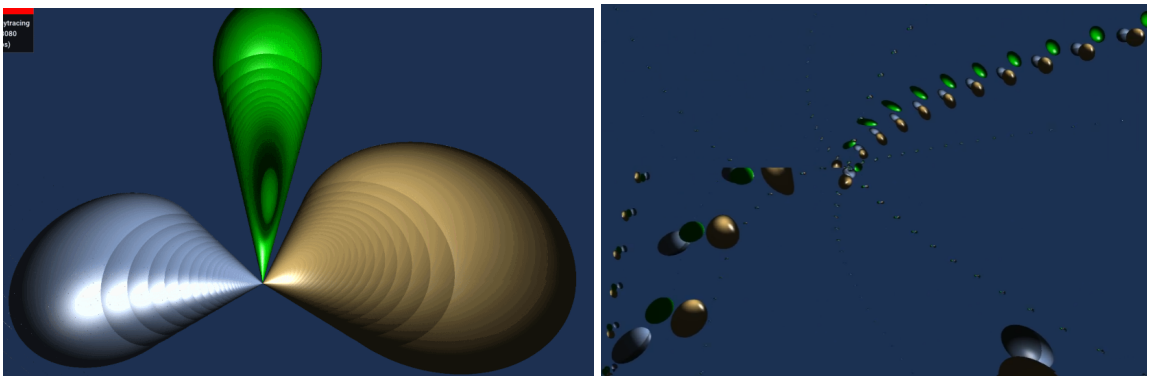

Figure 8: Current state of the Schwarzschild metric implementation

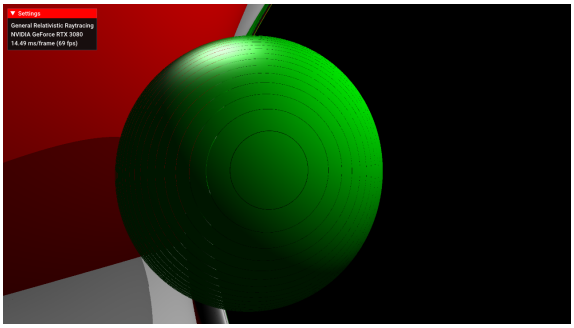

Figure 9: Visible banding on the surface of some geometry in close proximity to a blackhole

[10] Mack Qian. Approximate black hole renderings with ray tracing. *RPI Advanced Computer Graphics*, 2021.

[11] Alain Riazuelo. Ray tracing in a schwarzschild metric to explore the maximal analytic extension of the metric and making a proper rendering of the stars. 2018.

[12] SaschaWillems. Vulkan. https://github.com/SaschaWillems/Vulkan, 2023.

[13] Jeremy Schnittman. Nasa visualization probes the doubly warped world of binary black holes. 2021.