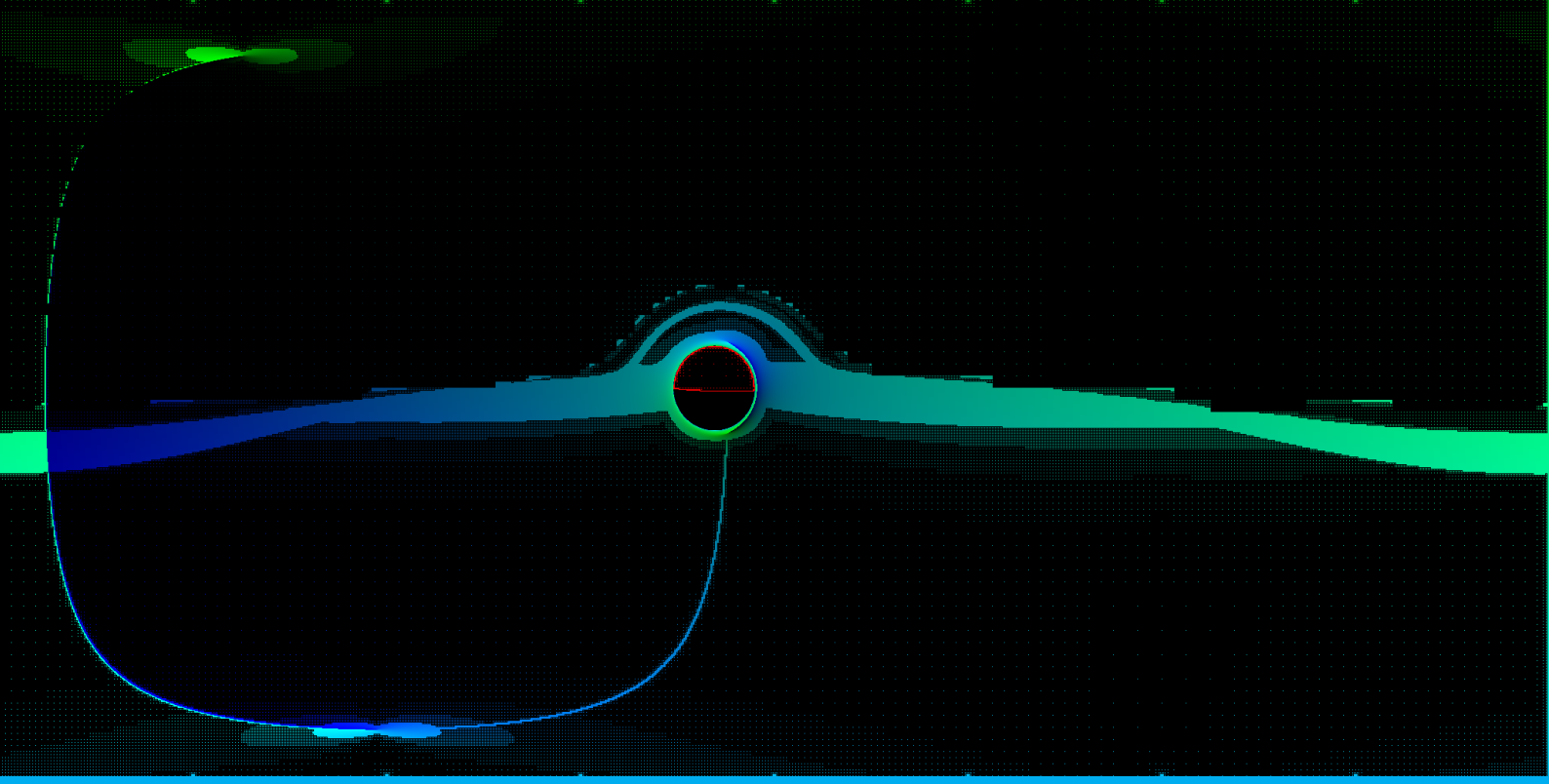


Interactive Accretion Disk in Kerr metric

Mika Zeilstra

Technische Universiteit Delft

TU Delft

Interactive Accretion Disk in Kerr metric

by

Mika Zeilstra

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended on Friday, October 6, 2023, at 10:00 AM.

Student number:	4960904	
Project duration:	November, 11, 2022 – October 6, 2023	
Thesis committee:	Prof. dr. E. Eisemann,	TU Delft, supervisor
	Prof. dr. ir. C. Vuik,	TU Delft
	Dr. R. Marroquim,	TU Delft

TU Delft

Abstract

Computer-generated images of black holes are invaluable tools for science, teaching, and entertainment. Nevertheless, up until recently, real-time imagery of situations with extreme space-time curvature were not possible. The work of Verbraeck and Eisemann [1] changed this. This work follows their line of research, building upon an adaptive grid-based rendering approach. Additionally, it includes the support of an accretion disk with realistic coloration and grid-interpolation, enables full free movement around the black hole and uses stereoscopic imagery to simulate depth. By carefully arranging the computations, a speed-up of up to 8 times over previous work was achieved for the grid generation, which enables a full real-time exploration of the black-hole space-time distortion.

1

Introduction

Computer-generated images of black holes are invaluable for science, education, and entertainment. Physics students use black holes to explore the extremities of general relativity [2]. Visualizations might make this exploration less abstract by summarizing results in a simulation. While this also applies to domain experts, there is an additional use : an accurate renderer may in the near future be used to corroborate models with what is observed in reality, in fact progress in imaging real black holes has recent breakthroughs [3].

Besides academic purposes, black holes also serve as an interesting story telling element. In the movie Interstellar, the relativistic effects of the black hole are one of the main concerns for the protagonists. What is more interesting to us, however, is that director Christopher Nolan wanted the black hole to look as realistic as possible to help convey his story. This desire lead to research into visualizing the black hole [4]. Creating these realistic images takes a long time, If this were faster it might be more commonplace, enticing youth with to follow a career in science.

This work will focus on extending previous work done by Verbraeck and Eisemann [1]. Their work introduced a method for speeding up the rendering of a Kerr black hole. This speedup is achieved based on a modification of the method of James et al. [4]. which was used to create the special effects for the movie interstellar.

In this work, we add an accretion disk to the method described by Verbraeck and Eisemann[1]. During geodesic integration, we can check for intersection with an infinitesimally thin disk and use this information to render the disk.

We also introduce the grid-summary, a diverse tool used in grid-interpolation and stereoscopic images. This summary speeds-up frame generation by not having to generate a grid for every frame. It also gives a warpable description of the accretion disk to allow for screen space stereo images.

The last addition this work will introduce is a black hole viewer. An interactive application in which the user can freely move a camera around the black hole.

Thus, the main goal of this thesis is to add a realistic accretion disk, while maintaining the speed achieved in previous work. Additionally, the room for optimzation will be used to allow for free movement, and stereoscopic images.

These topics will be treated in the following order: First, we will discuss some essential background concepts essential to understanding the remainder of the paper in chapter 2. The next chapter, chapter 3, will discuss some related work. After the related work, the method for adding the accretion disk is discussed in chapter 4 and the disk summary and interpolation are discussed in chapter 5. When the disk summary is discussed, we will leverage the speedup it provides to make a viewer capable of freely moving around the black hole in chapter 6, and warping the summary to simulate depth in chapter 7. Lastly, some miscellaneous optimizations and the actual performance of the grid summary will be discussed in chapter 8. The last chapter, chapter 9, contains the discussion of the results, future work, and conclusion.

2

Background

This chapter will explain some concepts of general relativity and some terms used integral to understanding the remainder of this work. The first part is about the metric a description of how space-time is warped around an black hole, the second part is about geodesics the path light follows through warped space-time, and the last part is about the grid, and optimization technique introduced by Verbraeck [5].

2.1. Metric

In contrast to our everyday experience on earth, light does not travel in straight lines near supermassive objects. The mass of the object curves space and time around it. This space-time curvature means that the shortest path between two points, referred to as a geodesic, might appear curved to an outside observer.

This means that in certain circumstances, we might be able to look "around" opaque objects. When a photon follows a curved trajectory, we can even see behind an object. As a simile, a photon behaves in this case like a ball kicked with effect, curving itself around the defensive players. Very close to the black hole, this effect becomes larger, where light rays can travel multiple times around the black hole. By traveling around the black hole, we approach a singularity in the sense that there are an infinite amount of light paths from any object in the sky around the blackhole to the observer. Since photons of the same object can appear to be coming from different angles, it means the object will appear multiple times in different "images". All these peculiarities mean that standard methods that have been employed in standard computer graphics, such as rasterization and ray-tracing, do not work correctly to generate an image out of the box, since they assume that light follows straight lines.

The curvature of space-time and thus the shape of the geodesics is described by a metric, a mathematical description of how two events are separated independent of observer. This description is usually given in the form of a line segment, the separation of the events as a function of the difference in the 4 coordinates defining an event, 3 space coordinates and one extra coordinate for time. For Minkowski-space or flat space-time, the line-element has the form as given in Equation 2.1.

$$ds^2 = dx^2 + dy^2 + dz^2 - (c^2 dt^2) \tag{2.1}$$

In Equation 2.1 we see that the line segment created by two events is the same independent of location, as no coordinates of the observer are required. Thus, if we take two events and translate them, the line segment stays the same.

In the Kerr metric, which describes a rotating black hole, it is not the case that a line segment is necessarily the same after a translation. the metric is shown in Equation 2.2 and Equation 2.3.

$$ds^s = -\alpha^2 dt^2 + (\rho^2 \Delta) dr^2 + \rho^2 d\theta^2 + \bar{\omega}^2 (d\phi - \omega dt)^2 \tag{2.2}$$

$$\begin{gathered} \rho = \sqrt{r^2 + a^2 \cos^2\theta}, \qquad \Delta = r^2 - 2r + a^2, \qquad \Sigma = \sqrt{(r^2 + a^2)^2 - a^2 \Delta \sin^2\theta} \\ \alpha = \frac{\rho\sqrt{\Delta}}{\Sigma}, \omega = \frac{2ar}{\Sigma^2}, \qquad \bar{\omega} = \frac{\Sigma \sin\theta}{\rho} \end{gathered} \tag{2.3}$$

Equation 2.2 describes a rotating black hole at the origin of the coordinate system. In these equations the most important constant is a, its rotational energy over the mass, in essence how fast the black hole is rotating. The coordinates are in Boyer–Lindquist coordinates, a special coordinate system based on an oblate spheroid coordinate system. The system also reduces to an oblate spheroid coordinate system, Schwarzschild coordinates, as a approaches zero, as expected, since we would have a non-rotating black hole in that case.

2.2. Geodesics

To use a metric we need to find the null-geodesics, the path a massless particle, such as a photon will take though space. The process of acquiring these geodesics is out of the scope of this thesis and will not be discussed. The geodesics themselves are however very important, and the equations for a photon traveling through space taken from James et al. [4] is given in Equation 2.4. Here p_r and p_ϕ are the momenta of the photon in those directions and R, θ and b are a fairly simple function of the initial value of these momenta. ζ is the integration parameter and can be seen as a sort of proper time, we simply step through the values of ζ to get the locations along the geodesic.

$$\begin{aligned} \frac{dr}{d\zeta} &= \frac{\Delta}{\rho^2} p_r, \frac{d\theta}{d\zeta} = \frac{1}{\rho^2} p_\theta, \frac{d\phi}{d\zeta} = -\frac{\partial}{\partial b}\left(\frac{R+\Delta\Theta}{2\Delta\rho^2}\right) \\ \frac{dp_r}{d\zeta} &= \frac{\partial}{\partial r}\left[-\frac{\Delta}{2\rho^2} p_r^2 - \frac{1}{2\rho^2} p_\theta^2 + \left(\frac{R+\Delta\Theta}{2\Delta\rho^2}\right)\right] \\ \frac{dp_\theta}{d\zeta} &= \frac{\partial}{\partial \theta}\left[-\frac{\Delta}{2\rho^2} p_r^2 - \frac{1}{2\rho^2} p_\theta^2 + \left(\frac{R+\Delta\Theta}{2\Delta\rho^2}\right)\right] \end{aligned} \tag{2.4}$$

This stepping is done using an adaptive explicit Runge-Kutta Cash-Karp integration. This method uses a linear combination of up to 5th order derivatives to find the direction and length of the next step. The size of the step is adaptive, meaning it can take advantage of smoother parts of the geodesics to step further, while staying under an estimated error. Stepping through these equations does not have a determined endpoint, however a logical endpoint is after we have traveled a “large” distance. Stopping after a large distance works since the effects of the black hole quickly diminish with distance and far away points can thus be considered to be in flat space-time, in which photons travel in straight lines.

To get a better intuition on how the geodesics are actually curved through space, a tool was written, which shows the geodesics as they were traced by the program through the metric. In this program, the user can move around and look at the geodesics from different angles. The results are shown in Figure 2.1. It can be seen that some geodesics seem to stop midair around the equator, this is due to them hitting the accretion disk, a light emitting cloud of particles orbiting around the black hole. chapter 4 goes into more detail about this disk and shows examples.

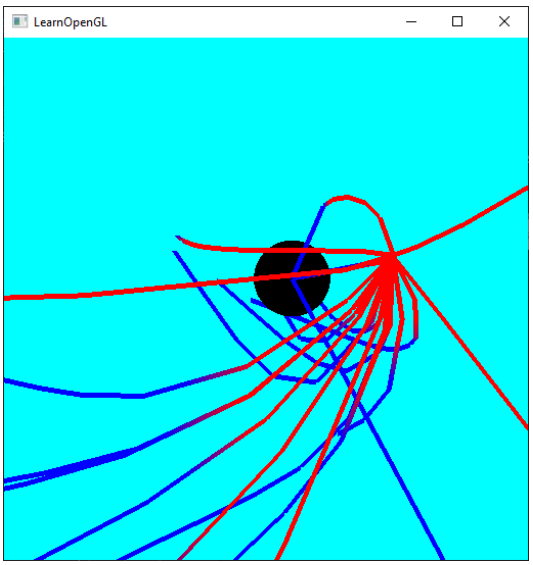


Figure 2.1: The output of the geodesic visualizer, showing a subset of the traced geodesics. They are colored based on the amount of steps taken going from red to blue. Each geodesic is sampled every tenth step.

Following these geodesics, we can map a photon going in a specific direction to a point on the celestial sky. The celestial sky in this case refers to what would be seen in flat spacetime and other possible objects, in this case the background images, stars and, the accretion disk. The result is thus a mapping from the camera sky, All the rays coming into our virtual camera, to the celestial sky, a direction into what is considered flat time-space or an intersection point with the accretion disk.

2.3. Grid

If it was required to integrate a geodesic for every pixel, reaching interactive speeds would simply be impossible. Therefore, the last important idea builds upon the use of an adaptive grid to take advantage of the relative predictability further away from the black hole, instead of following a geodesic for every pixel separately as described by Verbraeck and Eisemann [1].

For an accelerated computation, a grid with equally spaced points is created, which are much sparser than the pixels. Each point forms a square “block” together with three other points. If the celestial sky image is known for each of these points, a decision is made whether interpolation will produce good enough results, or it needs to be refined. This decision is made based on a measure of how deformed the block is. If the deformation inside the square is small the interpolation will not make noticeable mistakes, if it is deformed a lot we need to divide the block into four smaller blocks, which are recursively processed in the same way reducing the error down to the required accuracy. At the finest level, each grid-point represents the corner of a pixel.

3

Related work

Work on relativistic renderers has been on a slow but steady pace since 1979, when Luminet [6] first produced an image of a black hole with accretion disk. Most of these renderers are specialized for non-rotating Schwarzschild black holes [6, 7, 8, 9, 10, 11, 12], as opposed to the rotating Kerr black holes this work will focus on [5, 4, 13, 1, 14]. Research into the simulation of the accretion disk has been there from the start [15, 16, 17], but in the computer science realm it has recently subsided.

3.1. Schwarzschild metric renderers

Schwarzschild black hole renderers can further be subdivided into two categories, analytic solvers [7, 10, 6] and numerical integrators [8, 9, 11].

Solving the null geodesics analytically, while possible for a Schwarzschild black hole, according to Yamashita [8], does not provide performance gains, but yet such techniques usually do increase computational complexity. Moreover, analytic methods are currently not practicable for Kerr black holes, which are defined by a more complicated metric. For these reasons, we will not further discuss these solutions.

Regular flat space-time ray-tracing/marching and numerically integrating null-geodesics have a lot in common. In both cases we follow the path of a light ray backwards in time back to the source. Mandal et al.[9] use these similarities to its fullest potential and enable features standard in modern ray tracing, such as medium changes and indirect lighting, and add these to a relativistic ray tracer. This slows the image generation down significantly, but can produce stunning images of black holes in everyday scenes.

Stars in our context can be considered point light sources and are therefore in general not hit by the 1-dimensional geodesics if simply back-traced through time. Riazuelo [11] proposes a solution for this. It is known that the Schwarzschild metric is fully symmetric in both the theta and phi axis. This symmetry means that all null-geodesics are planar, they do not exit the plane formed by the starting point, end point and black hole. Using this planarness, we can efficiently search for the geodesic between two points, since the emitting/receiving angle is the only free variable. Therefore, stars are not integrated backwards in time, but forwards. Riazuelo [11] searches for the geodesic connecting the star and camera and uses the incident angle at the camera to paint the star into the final picture. This can be repeated as many times as required, since multiple geodesics can fulfill these criteria.

Another feature Riazuelo [11] introduces is the ability to go inside the event horizon. The standard Schwarzschild coordinates break down in this region, therefore they used a system that dynamically switches between Schwarzschild and Kruskal coordinates.

Yamashita and Müller et al. [8, 10] both separately introduce an interesting optimization utilizing lookup tables for the geodesics. They exploit the same symmetry Riazuelo [11] uses, but instead of simply finding the geodesics connecting stars and the camera, they save the geodesics. This lookup table can then be used to project geometry onto the image similarly to the rasterization pipeline. Thus, in practice for each sufficiently small part of the 3d model, A geodesic can be found connecting it to the camera, which is saved and used to color a pixel in the image.

A last interesting paper to discuss about schwarschild metric renderers is the work by Viergutz [12]. They do not go into detail about the actual render process, in which they show an observer descending into a

schwarschild black hole. However, they do go into a detailed discussion about how stereoscopic vision does not work correctly to visualize depth in warped space-time.

3.2. Kerr metric renderers

Further work on visualizing Kerr black holes is mostly focused on speeding up frame generation. Chan et al.[13] do this with their GRay renderer by applying the GPU to integrate geodesics, realizing significant speed-ups.

Projecting stars on the image poses an interesting problem. The previously discussed method of Riazuelo [11], does not work efficiently enough in the Kerr metric. In the Kerr metric, geodesics are generally not planar, and thus an efficient search is not possible. The rotation that defines the Kerr black hole essentially drags the geodesics out of the plane by applying some of its angular momentum to the photon, hereby reducing the spherical symmetry of a Schwarzschild black hole to only axial symmetry on the axis of rotation of the black hole.

In 2014 the movie Interstellar was released, A significant part of the plot of this movie is focused around a Kerr black hole and an Ellis wormhole. The Double Negative special effects studio which realized the special effects for the movie wrote two papers about this process [4, 18]. In [18] James et al. describe how to visualize the wormhole, while in [4] they describe the process of creating IMAX quality images of a black hole.

The approach of James et al. [4] to visualize a black hole and its effects on the celestial sky is to map elliptical ray-bundles, from the camera to the celestial sky by following the geodesics backwards, creating a mapping from the spherical uv coordinates of the sky to the spherical uv coordinates of the camera. This works by tracing the ellipse along the geodesic, and filling the found patch of celestial sky to the rendered image. This technique also trivializes the adding of stars, since they can simply be added together with the patch of the celestial sky.

Verbraeck and Eisemann [1] significantly speed up the frame generation process. They introduce a method to only densely sample where it is actually required. This is done by introducing an adaptive grid. On each vertex of this grid, representing a corner of a block, a geodesic is integrated to get its celestial sky coordinate, creating a mapping from the camera's local sky to the celestial sky as explained in section 2.2 and section 2.3. If a part of the grid is far away from the black hole the distortion is minimal, and we can approximate the change by using spline interpolation. In contrast, if it is closer, the grid gets denser until each pixel has its own set of vertices close to the black hole. An example of this grid can be seen in Figure 3.1.

Figure 3.1: Grid generated as by the work of Verbraeck and Eisemann [1]. The grid as seen from $r = 10$ and $\theta = 0.5\pi$. If the pixel is black, the grid coordinate remains unsampled, if it is red the ray hit the black hole, otherwise the blue channel is the theta coordinate and the green channel the phi coordinate of the celestial sky.

A second improvement is the interpolation between grids. When several successive frames are required, they will be very similar. The similarity can be exploited by not generating a grid for every frame, but inter-

polating two grids over several frames. Generating fewer grids means trading a bit of accuracy for a speedup. The rest of this work will be based on their work.

3.3. Accretion disk

Research on the visualization of the disk has been done since the start of research into black hole visualization. Before the first actual visualization work, Page and Thorne in 1974 [16, 17] describe how an accretion disk can be modelled. They describe how the temperature behaves and how the disk is shaped.

The initial paper by Luminet [6], which first describes a black hole visualization, already included a thin accretion disk. They show the isoradial curves of an accretion disk around a Schwarzschild black hole and use these curves in combination with the work of Page and Thorne [16] to also created an image of bolometric flux, the light intensity combined over all frequencies.

Fukue and Yokoyama [15] later extended the approach of Luminet, with the application of redshift (gravitational and Doppler) and, maybe just as important a clear explanation of how to take the photograph. They end their work with a brief discussion about eclipses of the accretion disk.

Viergutz [14] describes an accretion disk around a Kerr black hole. They thoroughly discuss how to analytically find geodesics connecting the observer and all visible geometry in the Kerr metric, and use this information to show both an infinitesimally thin disk and the much more impressive thick disk. Besides the disk visualization, they also explain how to calculate the gravitational redshift.

Further research has mostly been focused on simply showing that a disk can be added in a reasonable amount of time. Müller and Frauendiener [7] show for the first time a real-time image generation algorithm for the Schwarschild case with an accretion disk. While James et al. [4] show several accretion disk effects. They do not go into any detail on how they are achieved. One of the most recent works to include an accretion disk is the work by Verbraeck and Eisemann [1] shows that a disk can be added, but did not investigate corresponding rendering or acceleration solutions.

4

Our Method - Accretion disk

One important element of our method is the calculation of an accretion disk, which forms around a black hole when it is "consuming" mass. In this chapter, we describe the derived model and discuss the resulting implementation. The next chapter will focus on the possibility to navigate freely around the blackhole, before we illustrate a solution to generate stereoscopic image pairs for a 3D visualization.

Dust and gasses end up in orbits and fall into the black hole. Conservation of momentum restricts particles to a disk like shape around the equator, which is heated up by viscous stresses. The heated particles emit a large amount of light in a wide variety of wavelengths [16]. To visualize this disk, some approximations need to be made. In this work, we will assume that the disk is opaque and infinitesimally thin, located at $\theta = \pi$. The start and end radius of the disk can be determined by the user. This setup is shown in Figure 4.1.

The naive solution to add the disk would be to check during geodesic integration steps if a step crosses the equator within the radius of the disk. Given the assumption that the disk is opaque, the stepping procedure is stopped. Some additions are necessary, though, as this method leads to artifacts, and does not provide a definition for the actual color of the disk.

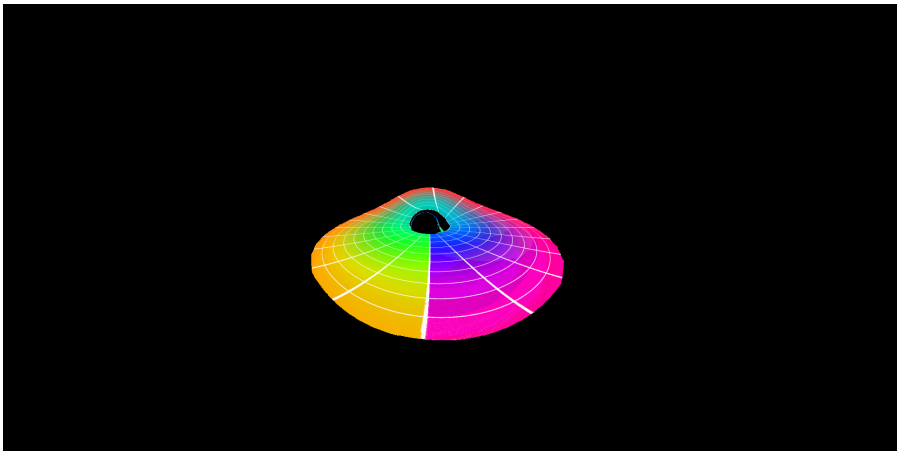


Figure 4.1: A 360 degree image of an accretion disk with a grid texture mapped showing equal phi and R values. The disk is in the equatorial plane of 0.5π and ranges from the most inner stable orbit at $r = 6$ to $r = 40$ and the camera is located at $[50, 0.4\pi, 0]$.

In the rest of this chapter, first, there will be a discussion on how to properly model the predicted color of the accretion disk section 4.1. Starting with the proper temperature of the disk, redshift effects will be added to obtain the observed temperature and lastly the final color will be generated from this temperature. After the discussion about the modelling in section 4.2, we will go into more detail about the encountered challenges and the solutions during the implementation of this model.

4.1. Modelling the disk

4.1.1. Proper temperature

The light emitted by an accretion disk can be modelled as black body radiation. In this model, the disk consists of small dust particles, each being a black-body radiator. But to model black body radiation we need the temperature at a given point of the disk, Fukue and Yokoyama present an equation for exactly this relation [15], where the bolometric flux, the total luminosity at all wavelengths, is related to the particle's R coordinate.

$$F = \frac{3GM\dot{M}}{8\pi r_g^3}\frac{1}{(R_s-3/2)R_s^{5/2}}[R_s^{1/2}-3^{1/2}+\frac{(3/2)^{1/2}}{2}\ln\frac{R_s^{1/2}+(3/2)^{1/2}}{R_s^{1/2}-(3/2)^{1/2}}\frac{3^{1/2}-(3/2)^{1/2}}{3^{1/2}+(3/2)^{1/2}}] \tag{4.1}$$

$$T = (F/\sigma)^{1/4} \tag{4.2}$$

Equation 4.1 and Equation 4.2 describes this relation between the r coordinate, bolometric flux F and temperature. In these equations we have constants G, σ representing the gravitational constant and the Stefan-Boltzmann constant respectively and $\dot{M}, M$ and r_g are the accretion rate, Mass and Schwarzschild radius of the black hole respectively. Lastly, R_s is the radius of the point in the accretion disk in units of Schwarzschild radius. The fact that we need to use schwarschild radii instead of the simple R coordinate is annoying, since it does not directly correspond to the rest of the application. If we substitute the equation for the Schwarzschild radius, and simplify the constants, we will get a much better picture of the entire equation.

$$r_g = \frac{2GM}{c^2} \tag{4.3}$$

$$T = 4.2939\cdot 10^9\left(\frac{\dot{M}\left(4.0R^{0.5}+2.4495\log\left(\frac{(-\sqrt{6}+2\sqrt{3})(2\sqrt{R}+\sqrt{6})}{(\sqrt{6}+2\sqrt{3})(2\sqrt{R}-\sqrt{6})}\right)-6.9282\right)}{M^2R^{2.5}\cdot(2.0R-3.0)}\right)^{0.25} \tag{4.4}$$

Equation 4.3 shows the equation for the schwarschild radius and Equation 4.4 has this equation substituted into Equation 4.1, and all constants numerically evaluated. This equation now only depends on the radius R of the particle's orbit, the mass of the black hole and accretion rate of the black hole. The black hole properties are defined at the start, so that a lookup table can be generated to reduce the amount of computation required for each accretion disk pixel.

The table size is determined by the user and starts at $R = 3$ as at lower R the temperature function is undefined. On lookup, the entries in this table are linearly interpolated to generate a value for any requested R.

4.1.2. Observed temperature

Due to the particles moving at relativistic speeds and the light needing to travel away from the black hole, red/blue shifting occurs and distorts the proper temperature such that it is observed differently. According to Müller [7], this redshift can be split up in the gravitational part, z_{grav}, caused by light traveling outwards to flatter space-time and the Doppler part, z_D, caused by the radial direction with respect to the emitted light rays. These two parts can be combined to the total redshift, z according to Equation 4.5. where the observed temperature is equal to Equation 4.6.

$$1+z = (1+z_{grav})(1+z_D) \tag{4.5}$$

$$T_{obs} = \frac{1}{1+z} * T_{proper} \tag{4.6}$$

Gravitational redshift

The gravitational redshift solely depends on the space-time metric and the positions of the emitter and observer. This problem is usually solved for an observer at infinity to confirm real-world observations. It is out of the scope of this project to properly modify this formula to depend on the radius difference between observer and emitter, therefore the gravitational redshift will be approximated with the observer being at infinity. This means the redshift is in general overestimated, with it being more accurate the further away the observer is. The gravitational redshift in this situation according to Dubey and Sen [19] is then given as in Equation 4.7.

$$\frac{1}{1+z_{grav}} = \sqrt{-g_{tt} - 2g_{t\phi}\Omega - g_{\phi\phi}\Omega^2} \tag{4.7}$$

Here $g_{tt}, g_{t\phi}$ and $g_{\phi\phi}$ are metric tensor entries and Ω is the angular velocity of the emitting particle. We need to be careful here as Dubey and Sen [19] use the (+,-,-,-) coordinate convention, and we use (-,+,+,+). Coordinate conventions are a handy tool for physicist to keep the inherently different acting time dimension separate from the spatial dimensions. Luckily, switching between conventions is as easy as flipping all the signs. Equation 4.7 has been converted to the convention used in this thesis and other previous work.

Doppler redshift
The Doppler redshift is due to the particles in the accretion disk moving at relativistic speeds with respect to the observer. Since this effect does not depend on the metric directly, the method proposed by Müller [7] for schwarschild black holes could work, however we need the four vector of the incoming light direction. While the 3-direction is easily determined using the provided derivatives, the time component is hard to determine. However, an approximation can be made as the emitting particle is presumed to be in a stable orbit around the black hole which is a textbook example of an inertial reference frame, and we can apply special relativity instead. In this case, we only need the incoming ray directions' space components and the direction of the particles' space components. Since the particles orbit the black hole in the same direction as the black hole rotates, the normalized velocity vector is simply 0,0,1. The direction of the ray can be calculated using the derivatives of the intersecting null geodesic, which are already required for the ray integration. Equation 4.8 shows the formula for the special relativistic case used as an approximation. Here, V and R are the 3-velocity vector and the ray 3-direction respectively and β is the magnitude of the observed speed of the emitting part of the disk.

$$\frac{1}{1+z_D} = \frac{1 + (||V|| \cdot ||R_d|| * \beta)}{\sqrt{1-\beta^2}} \tag{4.8}$$

4.1.3. Observed chromaticity

With the temperature of the disk and its relativistic corrections, we can now calculate what an observer would see when looking at the disk. The first part of this process is calculating the spectrum emitted, this can be done using Planck's law as shown in Equation 4.9, here λ is the wavelength of the emitted light in meters, P is the hemispherical flux density in watts per square centimeter, and k_b is the Boltzmann constant [20]. This spectrum now needs to be multiplied with the CIE matching functions [21] and then integrated over the spectrum to get the color and intensity in the XYZ color space. We normalize this to xyz chromaticity coordinates, since the intensity of the emitted light is very large, it is left out for now and will be discussed in the next section. Now the chromaticity coordinates are known, we can apply the transformation matrix from xyz to linear sRGB [22] and gamma correct [22] the color to get the final chromaticity in sRGB.

$$P = \frac{2\pi hc^2}{\lambda^5} \frac{1}{e^{\frac{hc}{\lambda k_B T}}} \tag{4.9}$$

The RGB color at a given temperature thus only depends on the temperature of the emitter, this combined with the fact that the matching functions are already discretized means that this is an excellent candidate to make a lookup table from temperature to sRGB values. The result of this can be seen in Figure 4.2. We can now use this to lookup the color values of a realistic black hole. If we take the values for M87[23] the visualization can be seen in Figure 4.3.

Figure 4.2: Image of sRGB values in lookup table (replicated 40 times in height for clarity). The temperature ranges from 0 to 10000 kelvin and is linearly interpolated between lookup table entries at every 100 degrees Kelvin.

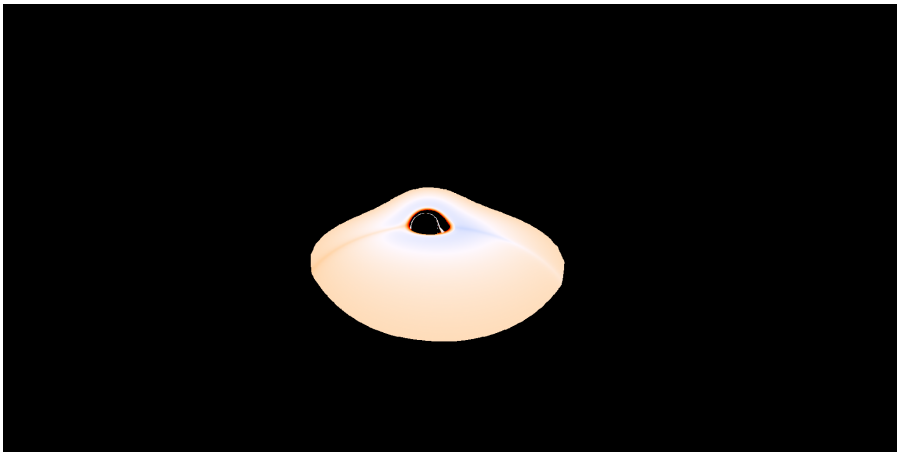


Figure 4.3: RGB values of a black body radiator mapped on the calculated temperature of an accretion disk. $\dot{M}, M$ are 0.1 solar masses/ year and 3.5×10^9 solar masses, respectively. The camera and disk are the same as Figure 4.1

4.1.4. Handling Intensity

To account for the wildly varying intensity of the emitted light, it was decided to attenuate the intensity based on the maximum proper temperature at 4.8 Schwarzschild radii, where the intensity is the highest. The intensity varies wildly over several orders of magnitude, to normalize this we first take the logarithm of the light intensity. This logarithmic light intensity can then be compared much easier. Using some experimentation, it was found that comparing the intensities by division gives good enough results, mitigating color clipping while retaining the color information.

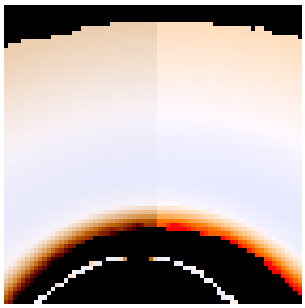


Figure 4.4: Magnified example of the accretion disk near the top part of the black hole. On the left we have the intensity corrected values and on the right uncorrected.

$$I_p = I_{sRGB} * \log(Y) / \log(Y_{T_{max}}) \tag{4.10}$$

Equation 4.10 Shows the exact formula used for attenuation. I_p is the intensity the final pixel color will get, while I_{sRGB} is the color calculated previously without using any intensity. Y is the color intensity in the XYZ color space and $Y_{T_{max}}$ is the intensity at the maximum temperature.

4.1.5. Gravitational lensing

Gravitational lensing occurs due to the warped space-time; beams of light get focused together into a tighter beam or spread out. An approximation for the strength of this lensing is the ratio between the flat space-time solid-angle of a patch and the actually observed solid-angle in curved space-time. For the celestial sky, this is fairly trivial and has been done before in previous work [5]. The flat space-time solid angle is divided by the observed solid angle and the bigger this ratio, the brighter the pixel will appear. However, the method used to calculate the solid angle does not directly work for the accretion disk. This is because the light rays might also curve and thus show a part of the disk which is tangent to the initial ray direction. Hitting a piece of disk tangent to the initial ray direction means a standard solid angle calculation does not work, as it would be zero, which contradicts the fact that it is visible.

Instead, to calculate the solid angle, we first calculate the actual surface area covered on the disk. To get a solid angle out of this surface, we need to know how much of a unit sphere centered around the camera this surface would block. This is done by assuming the rays traveled in a straight line, giving us the solid angle of light actually concentrated on the pixel. Comparing this to the value expected when space-time is flat gives an approximation of the magnitude of the lensing which can be used in the same manner as the solid angles of the celestial sky in the work of Verbraeck and Eisemann [1]. With this last addition, the calculation of the disk is complete, giving us the final image in Figure 4.5.

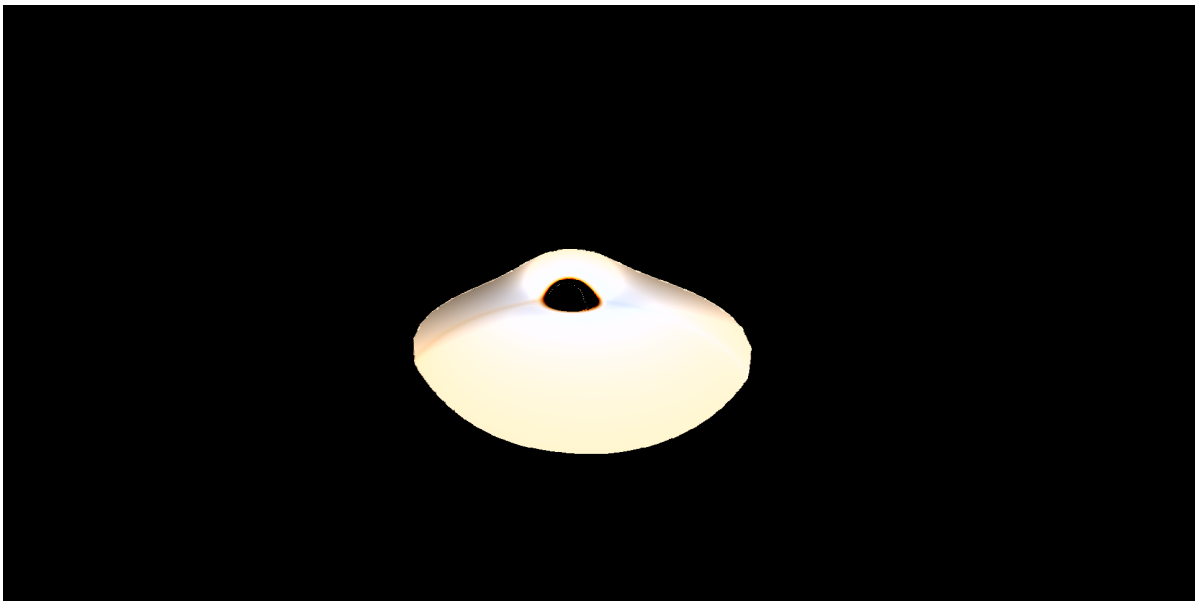


Figure 4.5: RGB values of a black body radiator mapped on the calculated temperature of an accretion disk corrected for gravitational and Doppler redshift, relative intensities and gravitational lensing. Camera and black hole setup is the same as Figure 4.3

4.2. Implementation details

Now that we have the model describing the accretion disk and its observed colors, several implementation challenges still remain. These challenges go from aliasing issues with the disk to the actual color computation.

4.2.1. Extra computation data

To actually color the disk, we need to store some extra values for use in the color computation, in contrast to the naive approach introduced at the start of the chapter. To use the above model, we need the R coordinate of the geodesic disk intersection to calculate the proper temperature. We also need the direction the geodesic is going and the length we traveled up until now. Additionally, we store the ϕ value of the intersection as well to allow for texture mapping. All the extra values are stored in the same type of grid structure as the celestial sky.

Storing the extra values in the same grid structure enables the use of existing interpolation functions [1]. However, these functions are specialized to work for the celestial-sky. The celestial-sky grid consists of 2-vectors, which are element wisely interpolated, and automatically wrapped to be between 0 and 2π. The only required modification are some generics to support different data types. This allows us to also interpolate the geodesic direction, which consists of a 3-vector, in contrast to the 2-vector used for the celestial sky. Also, a switch is added to make sure the θ/ϕ wrapping for interpolation can be switched off, as doing this automatically would be wrong for the geodesic direction. Both issues are easily solved using the c++ templating feature. Incidentally, this also allows the functions to be used for any other exotic grid values that might need to be interpolated in the future.

Lastly, the function which checks if the grid needs to be refined at a given grid block is still unaware of the disk. Two simple rules will fix this. The first one says that if at least one but not all corners hit the disk, we need to refine. This rule ensures the border of the disk is refined as much as possible for a smooth effect. The second rule is if all four points are on the disk, we need to bound the error just as the celestial sky. Therefore, we use as the size difference of the diagonals as a bound, following previous work [1].

4.2.2. Reducing disk jagging

The first issue arising with this naive approach is that the disk looks jagged, as shown in Figure 4.7a. Looking at the edge, the issue arises from how the edge is determined while integrating the geodesics. To determine where the disk is, at every geodesic integration step the θ coordinate is compared to the previous position, just as in the naive method discussed at the start of this chapter. If the coordinate passes $\pi/2$ in any direction, it has passed the equator, and we can check the R coordinate to see if the geodesic intersects the accretion disk and store the start of the step.

However, the actual $\pi/2$ crossing is in general not exactly at an integration step, but somewhere in between the step. As the rays from adjacent pixels will hit the disk in similar points, most of the time this will be at the same step, giving a smooth transition between pixels such as between the red and blue geodesics in Figure 4.6, but as each ray hits the disk at a different point, it is inevitable that the step of the intersection changes, causing a jump in the intersection point. This is exemplified between the blue and green geodesics in Figure 4.6.

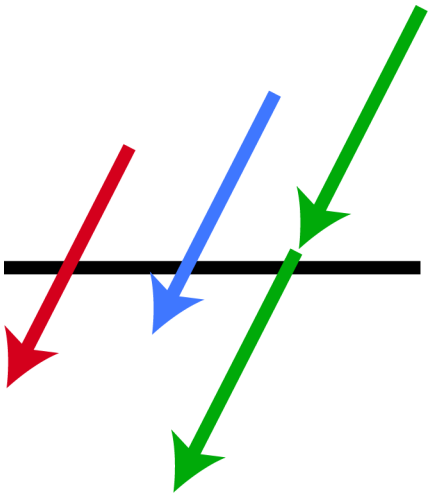


Figure 4.6: We can see geodesics originating from 3 adjacent pixels intersecting the disk. In the naive method, the starting point for each integration step is used as the intersection point. While this would not cause so much disparity between the red and blue geodesics, the intersection point jumps between the blue and green geodesics, causing a visible jump in color in our grid image and final image.

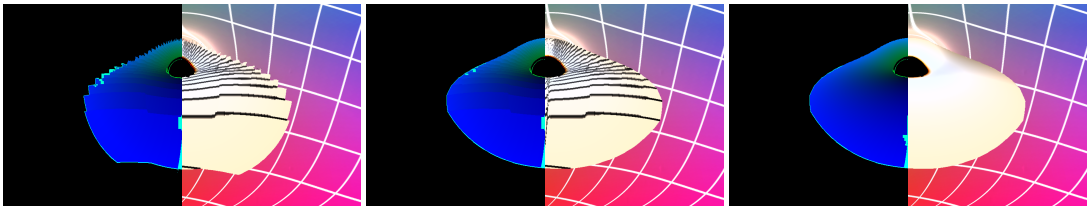
A first attempt to fix this disk jagging problem is to first find out how far we are overstepping by carefully comparing the θ coordinate between steps and interpolating both the r and ϕ coordinate to the intersection point using the θ comparison. As seen in Figure 4.7b, this interpolation fixes the outside edge as it is smooth now, however the coordinates still show a sawtooth pattern and thus more needs to be done to fix the problem.

The next attempt at minimizing these discontinuities is to first find the step which intersects the disk, return to the point before that step and take smaller steps. This is slower than interpolation since it requires much more calculation, but will provide with as much accuracy as required.

To determine the size of the extra refinement steps, the dynamic step size provided by Runge-Kutta is overwritten. Instead of the provided step size, we use a heuristic to determine the step size required to get to the disk. This heuristic step size, h_{heur}, is based on the current θ coordinate, the Runge-Kutta step size, h, and the absolute minimum step size $h_{\min}$. The relation is shown Equation 4.11.

$$h_{\text{heur}} = \max(|\theta - 0.5\pi| * h, h_{\min}) \tag{4.11}$$

By dynamically decreasing the step size proposed by Runge-Kutta we get a much more accurate estimate of the geodesic-disk intersection point. As seen in Figure 4.7c this refinement combined with the interpolation completely fixes the jagging issue.



(a) A composite image with the interpolated grid on the left and the result on the right, displaying the unfixed jagged edges.

(b) A composite image with the interpolated grid on the left and the result on the right, displaying the interpolation fix.

(c) A composite image with the interpolated grid on the left and the result on the right, displaying the interpolation fix combined with extra refinement.

Figure 4.7: Comparing the interpolated grid before and after fixing the jagged edges. The image is generated with the same parameters as Figure 4.1. The 360 degree image is cropped to mostly show the disk only.

4.2.3. Disk edge antialiasing

With the extra refinement to correctly find the disk, it now looks great at steeper angles, however when looking at angles closer to the equator the disk edges become flatter than it is supposed to be. Here, we will propose a solution to address this issue. First, we introduce grid images as a debugging tool, after which we use these to diagnose the exact issue.

Grid images

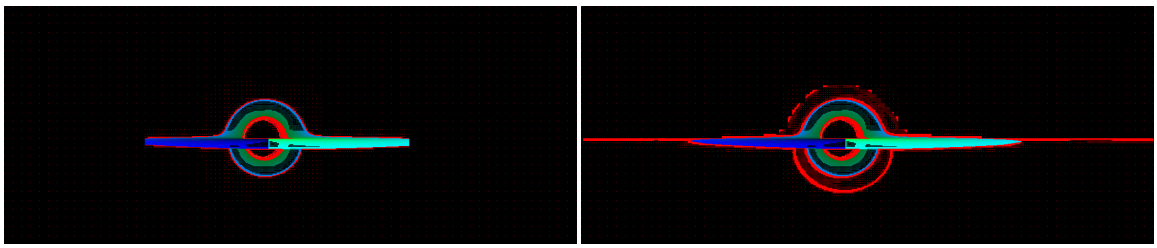
The idea of these grid images is to not only show how the celestial sky was sampled, but also what values were found at these points. Showing the grid values allows us to determine if the integration, refinement rules or a later step is at fault for artifacts in the image.

By using grid images as a debugging tool, we can check if the refinement rules are wrong if blocks we expect to be further refined are not or the opposite they are refined too far. This is visible in the form of grid points missing or showing where they should not be.

Checking the integration can be done by checking the colors. If the colors are different from expected, the values in that area are wrong and should be further investigated. On the other hand if both look fine the problem is most likely in a latter part of the algorithm. An example of a grid image is on the cover of the page and Figure 3.1.

Fixing the edges

An image of the grid and coordinates reveals the source of the aliasing. A grid block at the edge of the disk can have all four corners outside the disk, while there is still a part of the disk overlapping, as shown in Figure 4.8a. The simple rule, detecting whether all corners of the considered grid block are on the disk or all corners are off the disk, proves insufficient and causes aliasing. A fairly straightforward fix is to check whether the disk could be in the given grid block. This can be done by checking if the r coordinate in any corner is within one grid block of the disk, if so we refine. The results are shown in Figure 4.8b. Adding this fix costs a little extra compute time as the amount of rays traced increases by around 18% but solves the problem.



(a) A grid image showing the grid aliasing near the most left part of the disk.

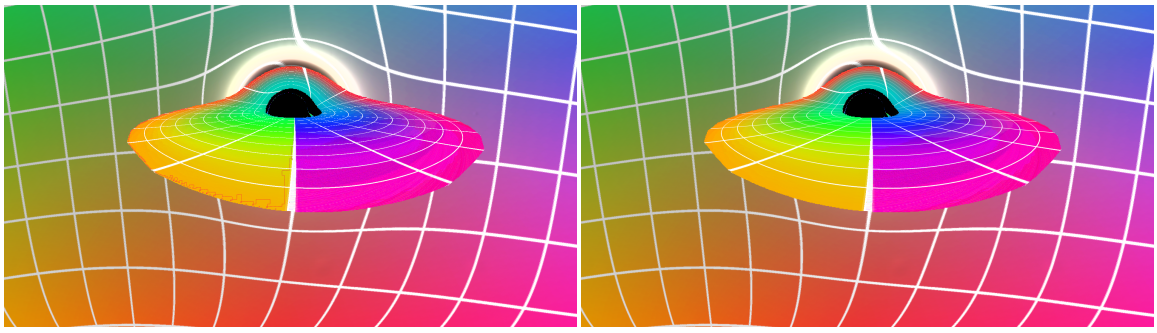
(b) A grid image showing the result of the extra refining.

Figure 4.8: Comparing the grid image before and after fixing the aliasing. The image is generated with the camera at $r = 50, \theta = 0.495\pi$ and $\phi = 0$, the disk is ranges from $r = 6$ to $r = 40$. The red colored points are points where the ray did not hit the accretion disk. The blue color channel corresponds to the ϕ coordinate and the green one to the r coordinate.

4.2.4. Texture mapping

In addition to calculating the temperature and corresponding color values, textures can be mapped on the disk and will appear correctly warped on the disk. An example of this is shown in Figure 4.1. Here, we map

the ϕ and R coordinates on the disk to the x and y coordinates of the texture respectively. However, the wildly varying pixel size causes significant aliasing artifacts shown in Figure 4.9a. Therefore, we can make use of the already calculated pixel corners. Since the corner coordinates on the disk are known, the texture coordinates of the pixel corners can also be found and thus the area that each pixel covers. Now, to introduce antialiasing, the texture area of the pixel can simply be integrated. To mitigate the amount of performance impact, instead of integrating over the exact area, the axis-aligned bounding box of the pixel in texture space is integrated with the result as shown in Figure 4.9b.



(a) The unfixed aliasing issues. The issue is most clear near the top of the disk where each pixel covers a lot of source pixels.

(b) The fixed aliasing issue. by integrating. The edges are much smoother yet also lighter as they are corrected for the entire area each pixel covers.

Figure 4.9: Comparing the accretion disk texture before and after fixing the aliasing. The image is generated with the camera at $r = 50, \theta = 0.45\pi$ and $\phi = 0$ and the disk is ranges from $r = 6$ to $r = 40$. A snippet from the full image without lensing on the disk is shown here to make the issues clearer.

Integration over the bounding box is much faster than integrating over the exact area because we can pre-calculate a lot of work. We can store the texture as a cumulative sum, with each texel storing the sum of all pixels with an x and y coordinate smaller than it. Integration is now as simple as subtracting and adding the values at the corner of the bounding box, in contrast to carefully adding all the texels which fall inside the pixel in the final image.

4.2.5. Texture transparency

A last an interesting feature to add to the disk is transparency. The separation of the celestial sky and accretion grids allows the accretion disk to be (semi) transparent. If the image is generated back to front with the celestial sky first and the disk after it, simple alpha blending achieves the goal of transparency as shown in Figure 4.10.

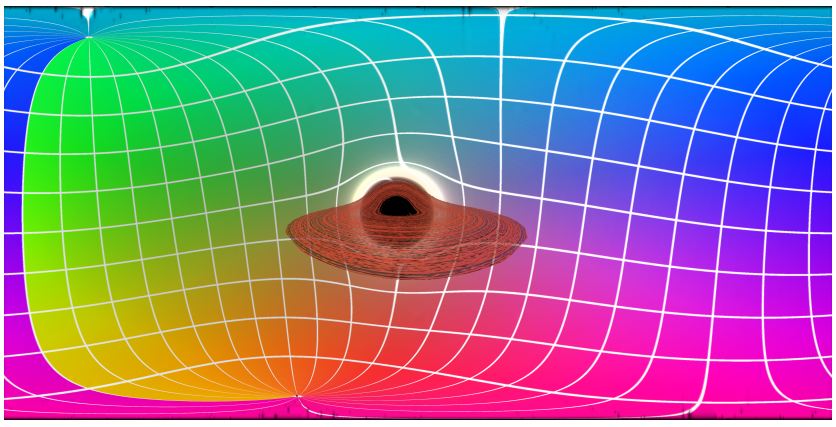


Figure 4.10: Image generated with a semi transparent accretion disk. The image is generated with the camera at $r = 50, \theta = 0.45\pi$ and $\phi = 0$ and the disk is ranges from $r = 6$ to $r = 40$.

5

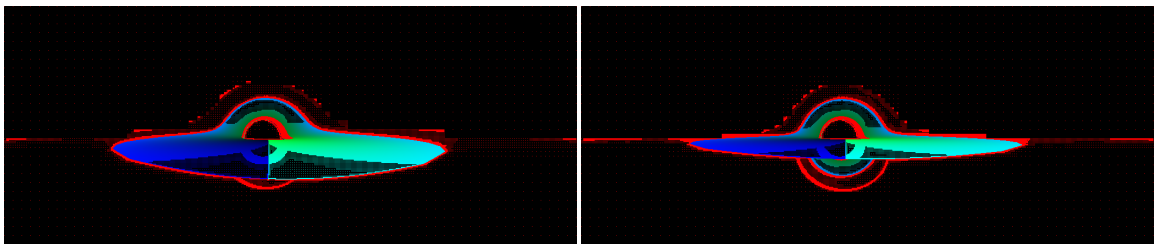
Our Method - Interpolation for Navigation

When the user wants to approach the black hole continuously, multiple grids need to be generated. Naively we would need a grid for every frame, however, it is possible to apply a grid interpolation [1]. With grid interpolation, the requirement to generate a grid for every frame can be dropped, avoiding a recomputation in each frame. Hereby, we can trade off accuracy to performance by manipulating how many frames are produced per grid generation. This is an interesting feature, which is not trivial to implement for the accretion disk.

For the celestial sky, a linear interpolation between the two relevant grids is mostly acceptable. In places away from the black hole, two points on the grid directly correspond, and only their value will shift. Closer to the black hole, a more suitable interpolation was employed to stretch coordinates in order to obtain an improved interpolation. Neither approach works directly for the accretion disk. Intuitively, the accretion disk morphs between different shapes and new parts may show up when moving around the black hole. If two grids are interpolated, data for a point on one grid might simply be missing on the other. An example of this can be seen in Figure 5.1.

Another difficulty is due to the fact that there is a single accretion disk, but it shows up multiple times in the same result. This is due to the fact that there is an infinite amount of geodesics connecting a point on the disk and the camera. Each geodesic produces a different image of the disk in the final result, these images can be ordered. Images of lower order are created by geodesics traveling fewer times around the black hole, while higher order images are created by geodesics travelling more often around the black hole. This can also be seen in Figure 5.1 when observing the area between the disk and the black hole, we can see another line of geodesics hitting the disk.

We attempt to overcome these difficulties by creating a summary which can be interpolated. The creation of this summary is discussed in section 5.1



(a) Accretion disk at $\theta = 0.48\pi$ (b) Accretion disk at $\theta = 0.49\pi$

Figure 5.1: Renders at $\theta = 0.48\pi$ and $\theta = 0.49\pi$ the ϕ and r values are constant at 0 and 50 while the disk ranges from $r = 6$ to $r = 40$. Interpolating between these two grids is nontrivial, since previously occluded parts of the celestial sky become visible and a new segment of the disk appears.

5.1. Creating the grid summary

To avoid having to directly interpolate the disk, we introduce a grid summary, a description of the disk which can easily be interpolated. This is done by measuring where the disk is from the center of the black hole at

multiple angles as described in subsection 5.1.1. After this the coordinates need to be sampled and saved which is described in subsection 5.1.2, and lastly some matching needs to be done to simplify the interpolation process subsection 5.1.3.

5.1.1. Creating segment-boxes

The disk parts are found by walking a ray in small steps over the 2D grid. These rays are directed in uniformly separated angles away from the black hole. These angles will form the backbone of the summary. Each angle records, for every part of the disk intersecting its path, where the disk starts along its path, where it ends along its path, and lastly all the samples taken along its path.

When walking a ray over the 2D grid the ray will intersect the disk at least once. However, when intersecting multiple parts of the disk each intersection can either be from the same disk image, a higher-order image, or lower-order image as shown in Figure 5.4. Because intersecting a new part of the disk does not guarantee finding an image of a different order, these found parts will be referred to as segments.

When a ray is walked from the center of the black hole over the 2D grid, there are three ways the ray can interact with a disk segment. The ray can enter a segment, it can leave a segment and lastly, it can jump from one segment to another without going to the celestial sky.

When a ray enters a disk segment, it takes a step from a red grid-point indicating the absence of the accretion disk to a blue-green grid-point, which represent coordinates of the disk intersection. At this point, the distance traveled for this ray is stored in a “segment-box” attached to this ray.

When the ray leaves a disk segment, it takes a step from a blue-green grid-point to a red grid-point. Again the distance is stored, but since we can guarantee that this interaction can only happen after also entering a segment, it stored in the same segment-box.

The last interaction is the most interesting, in this case the ray steps from a blue-green grid-point to another blue-green grid-point, but the grid-points belong to different segments. One can imagine this happening in Figure 5.1b. At the bottom of the accretion disk two, discontinuous segments are visible. We can recognize this since the color is different, and thus the coordinates changed discontinuously.

However, from just the found disk coordinates for each grid point, we cannot determine when we change from one continuous segment to another as the step might be inside part of the disk where the coordinates change fast but continuously, for example in the second order image of the disk, which is usually not larger than a few grid-points.

To be able to discriminate between a different segment or a rapidly changing segment, we can use the length of the geodesic until the intersection represented by the grid-point. This length traveled before the intersection should be wildly different for different segments. intersections part of the same segment should have around the same length geodesic, and a segment jump would have varying length. In this, the distance the 2d-grid ray has walked is stored both as the endpoint in the previous segment-box and as the start in a new one. Using the geodesic length does not incur a performance penalty, since these lengths are already required for calculating gravitational lensing.

In the end, we thus end up with a number of segment-boxes for each angle, and this can be considered a basic disk summary, an example of such a structure can be seen in Figure 5.2 and an example of a 2D ray traveling over the grid can be seen in Figure 5.4. Reconstruction of the current grid summary would involve to find the closest angles for each pixel and checking if the distance to the black hole is contained in the ranges in any of its segment boxes.

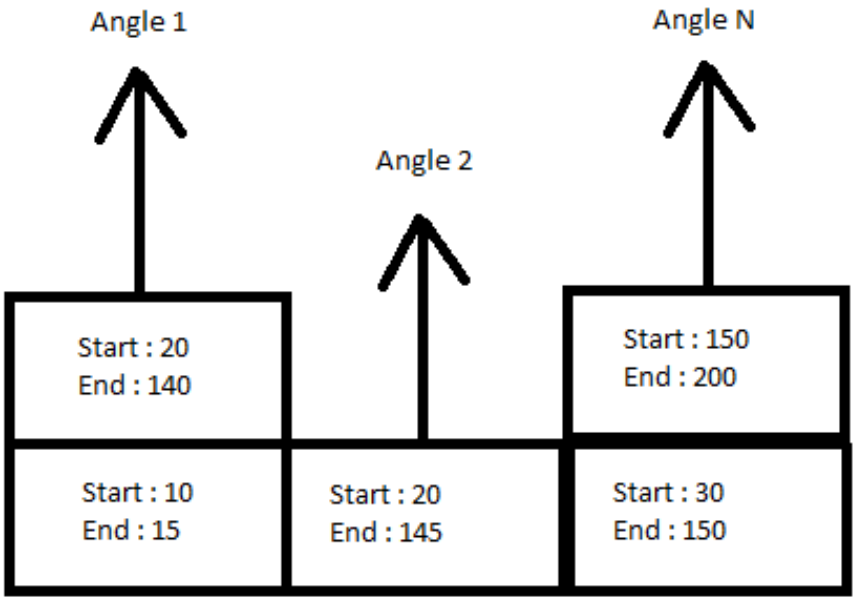


Figure 5.2: A plausible example of a basic grid summary. Each angle contains a stack of segment-boxes containing ranges in which the accretion disk is visible. Angles 1 and 2 only containing boxes generated from the first 2 types of interaction. The boxes in angle 3 on the other hand contain the third type of interaction, the range of the first box ends at the same place the second box starts. Angle 3 could be the result of red the ray shown in Figure 5.4, while angles 1 and 2 could be the pink and yellow rays respectively.

5.1.2. Sampling

Once the segments are found, they can be sampled. To keep the summary small and its creation fast, we need to keep the amount of samples taken to a minimum. Thus, it was decided to allow the user to determine amount of samples to be taken per angle/stack of segment-boxes. Each segment-box gets assigned an amount of samples weighted by the length of the segment it represents. This means that small one-point-wide second-image segments of the disk are not sampled more than needed, while the wider segments are sampled enough to reduce artifacts. All the samples are stored together with the range in the segment-box.

5.1.3. Matching

The last operation that needs to be done is to connect the found segments. As the ray travels outward, it finds the nth image of the disk first and the first image last. However, not all images of the disk are visible at every angle. This is very well visible in the mock disk summary in Figure 5.2. In this summary, angles 1 and 2 are next to each other, but angle 1 finds a higher order image first, misaligning the stack of boxes.

However, when reconstructing the disk it is assumed that if two boxes exist at the same layer in the summary that they are connected. This is a simple optimization, which means we do not have to search through the stack of boxes to find segments which connect, saving precious time.

The misalignment is inherent to storing the boxes in the order they are found. However, if we are observant, we can see that the order we want is exactly the reverse order. the first image of the disk is the largest and is always visible as a ring around the black hole. higher order segments are possibly found closer to the disk, but if the nth order image is not visible, neither will the (n+1)th image be, thus those boxes will all also be aligned. A reversed version of the mock summary is visible in Figure 5.3.

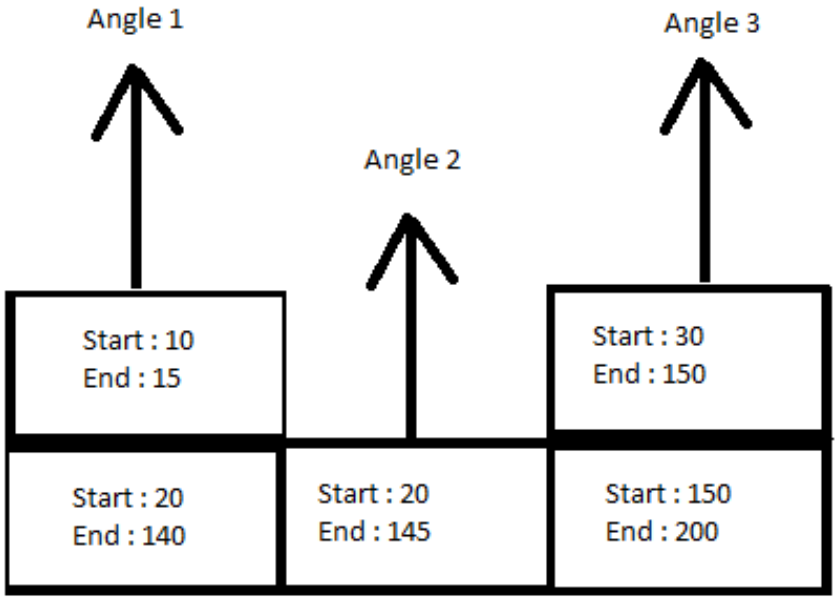


Figure 5.3: The grid summary viewed in Figure 5.2, but the stack order is reversed. This fixes angles 1 and 2 but angle 3 is not quite correct yet.

The last obstacle in generating the summary is that close to the equator, the disk images each consist of two segments; a main continuous segment which is the “front” side of the disk and an adjacent disconnected “back” side of the disk. This is what happens at angle 3 of the summary in Figure 5.2 and Figure 5.3.

As visible in these summaries, the boxes still do not line up well if we simply reverse the order. At angles 2 and 3 the top “front” side is connected to the bottom “back” side, which, if ignored, leads to artifacts as shown in Figure 5.6b. To fix this we introduce a zeroth box, this box is usually of zero length but if the backside is visible, every box is moved down such that the backside is in this newly created box. Cases of in which the ray has interactions with the backside can be discriminated by using the difference in length of the geodesic represented by the grid point. Normally as we move down over the stacks of boxes the length of the geodesics only decreases as the nth image requires the ray to travel (n / 2) times to travel around the black hole, however if the backside is visible this length increases again at the last box as we compare the second to last found front side to last found backside.

Lastly we see that the stack is bigger at angle 1, Here the second order image is also found. If we blindly assume the boxes at the same layer to represent the same segment we would run into issues when interpolating angels 1 and 2, therefore we also even out the stack with zero length elements.

When applying this procedure to the summary from the previous figures, we get the summary shown in the top right of Figure 5.4, in this figure the absolute numbers are interchanged for the letters representing the interactions.

5.2. Interpolation and Reconstruction

Next, we describe how to generate an accretion disk from the summary and how to interpolate two summaries.

To only reconstruct the disk, we need to run the following procedure for all pixels : First find at which angle and distance from the center of the black hole the pixel lies. Use the angle to determine the two closest angles in the summary.

Next, to determine if the pixel lies on the disk, loop over the stack of boxes for both adjacent angles at the same time, and interpolate the start and end position for each box at the same height to the angle of the pixel. This gives us a 2d-distance from the black hole center on the grid at which the disk is supposed to be, if the distance of the pixel to the black hole center is within this distance the pixel lies on the disk, and specifically on the segment stored at the height of the box.

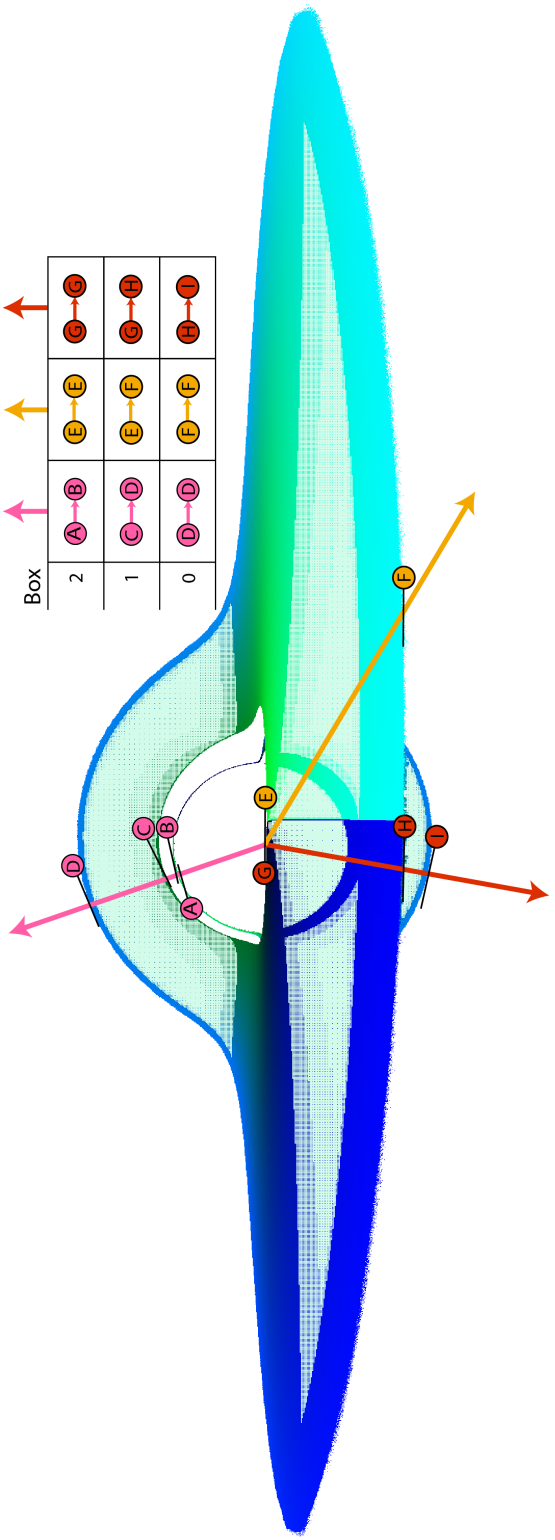


Figure 5.4: Visualization of a ray traveling outwards from the center of the black hole on the same grid as Figure 5.1b. The red ray hits all three types of boundaries, first it enters a segment at G, then it goes from one segment to another at H as it detects a significant change of length in the geodesic and lastly, it leaves a segment at I, initially creating two segment-boxes in the process. The yellow and pink rays each only use the first two types of interactions, but cross a different number of segments. In the top left we see a fully processed disk summary for each of the sampled angles.

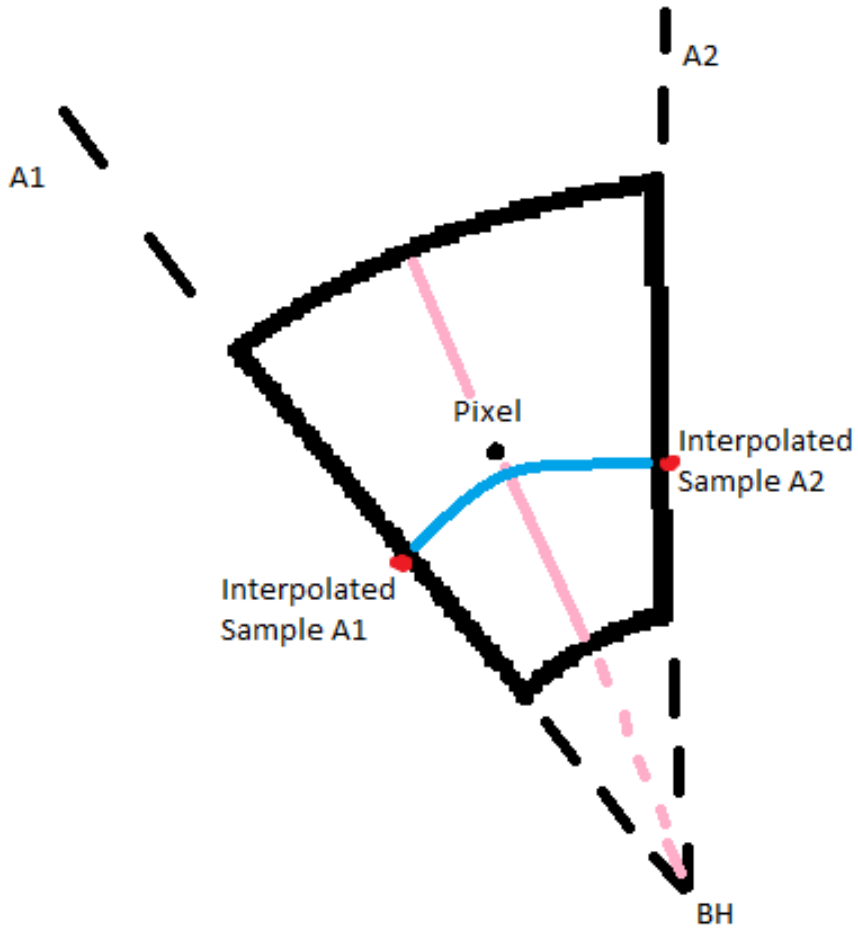
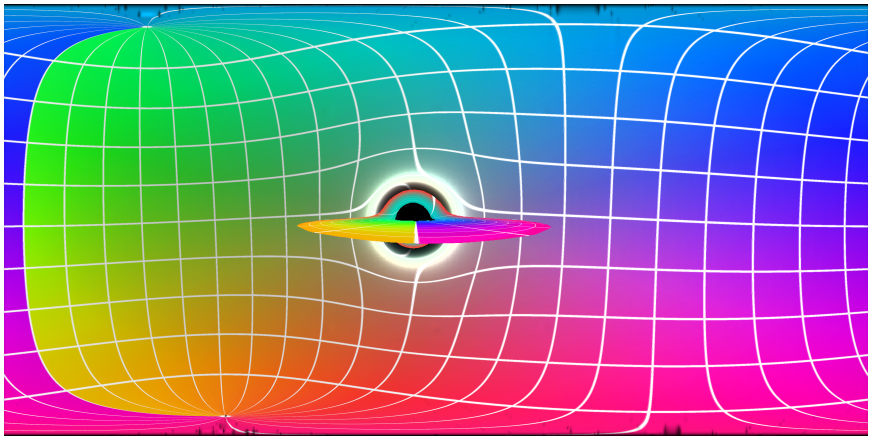


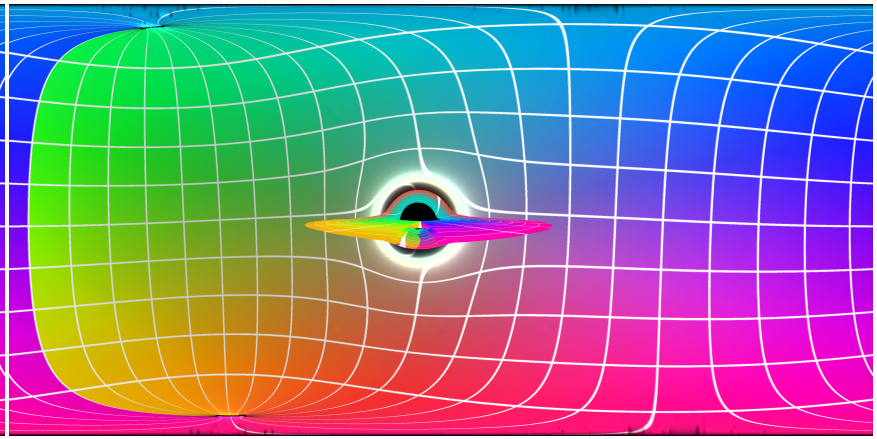
Figure 5.5: A reconstruction of the coordinates. First, we use the length along the full pink line to get a sample for the segment at both angles, in Figure 5.6c these samples are stored together with the ranges in the boxes. After that, we use the angle along the blue line to interpolate the samples again to get the final result.

Lastly, we need to determine where the geodesic disk intersection belonging to the pixel is. To do this we determine first how far along the segment the pixel is as a fraction. We use this information to get a sample value from the box at both adjacent angles. After these values are interpolated to the angle of the pixel we have the final estimated intersection point. This process is visually described in Figure 5.5.

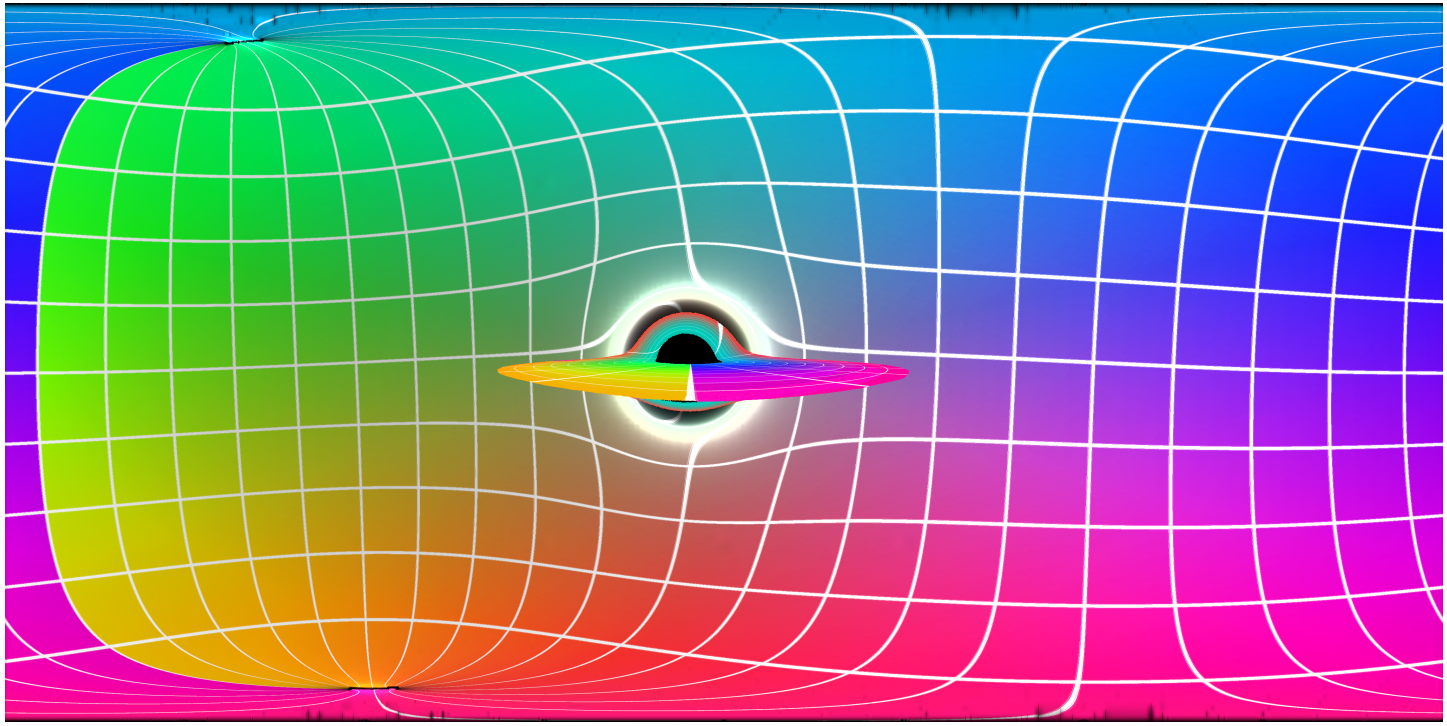
When not just reconstructing but also interpolating grids the process is not so different. The main difference is when checking if a pixel is in a segment, the interpolated range between the two angles is not enough, we also need to interpolate this range between the two grids. the second and last difference are that the final coordinate values which are calculated per grid also need to be interpolated between grids.



(a) An example of the disk as it would be generated using a grid



(b) The result from the interpolation algorithm, but the segments are not connected correctly.



(c) The final result from the interpolation algorithm, no artifacts remain, and it is very close to the original, 200 angles were used, each with 100 samples.

Figure 5.6: Three images with the camera at $r = 50, \theta = 0.485\pi$ and $\phi = 0$, with the disk ranging from $r = 6$ to $r = 40$. The left image shows the disk as generated directly from a grid. The middle and right image are using an interpolated disk summary from almost the same coordinates but instead the first source grid has $\theta = 0.48\pi$ and the later $\theta = 0.49$, which are the grids shown in Figure 5.1.

6

Our method - Free Movement

The main goal of this thesis is to show that a real time accretion disk is possible. While just showing the numbers is impressive as it is, it is way more interesting to show the user interactively moving around the black hole. To achieve this goal two extra parts need to be added, first a viewer to show the result in real time, and secondly a mechanism to automatically request the correct grids.

6.1. Viewer

Showing the results of the rendering process in real time is not such a big problem as it may initially seem. It can be divided into two steps, first get the result of a frame from the GPU global memory into an OpenGL texture and then project the correct part of the 360 degree image onto the screen.

CUDA, the interface used to do the general purpose GPU programming, provides interoperability functions with OpenGL. Writing to an OpenGL frame buffer is as easy as calling these interoperability functions in the correct order. At which point, the frame buffer can be used as any other CUDA memory. Copying from a frame buffer to a texture is also straightforward as calling the correct function.

Using the GLFW library [24] the user input is transferred to OpenGL using uniforms, and thus the shown part of the 360 image can be dynamically changed. The viewing direction is projected in the fragment shader and the screen is entirely covered by a single quad, which means we get exactly one fragment shader instance for every pixel. With one instance of the fragment shader for every pixel and the direction the user is looking at, the problem reduces to simply projecting the pixel to the correct part of the 360 image. The application is shown in Figure 6.1 and shows how a user can look at the black hole.

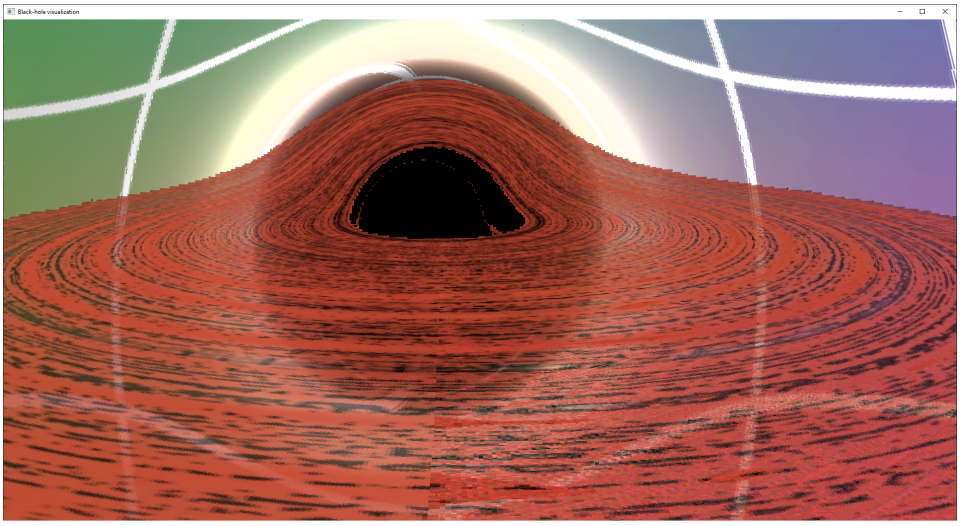


Figure 6.1: Screenshot of the application showing the black hole with an accretion disk. The camera is situated at $r = 50, \theta = 0.45\pi$ and $\phi = 0$. The viewer is looking in the direction of the black hole.

6.2. Grid generation

The second part of generating the grids is less straightforward. The first dimension of movement, moving in the direction of the ϕ axis, is trivial and can simply be done by applying an offset to the coordinates in the grid. This is a valid option since the metric is completely symmetric around the ϕ axis. For the other two axes, two different approaches can be used. In the first approach, four grids are generated around the current camera position. Two for both the θ and R coordinates, one ahead of the camera position and one behind. These four grids can then be bilinearly interpolated in a way similar to what is already done for moving along a predetermined path. The second approach is making the assumption that if a user is moving along a direction, they most likely will keep moving along that same direction, and thus only generate a grid in that direction and interpolate between the two grids.

The second approach was chosen to be implemented mostly due to time constraints, the reasoning being that the first option requires an extensive rewrite of all the interpolation code, while the second approach only requires some extra logic to request the right grids and calculation of the correct grid blending alpha.

The user input determining the position is again delegated to the GLFW library[24], which provides a callback function. This function is then used to modify the camera position which simulates moving around the black hole.

To find the correct grids to use and the alpha blending between them, we need to process the camera position every frame. To do this, each frame is classified into one of three categories.

The first category is the trivial category, there was no movement. In this case, the grids and alpha blending from the previous frame are simply reused.

The second category is that the direction of movement changed. For example, if we were initially moving along the θ-axis and this changed to the r-axis. In this case, we simply need to generate two new grids, as both previous grids are now aligning in the wrong direction.

The last category is that the user keeps moving along the same direction. Here we first need to calculate the alpha blending and then correct it for going outside the range of the two grids. Equation 6.1 shows the calculation.

$$\text{alpha} = ((C - G_l)/\Delta G) \tag{6.1}$$

In this calculation, the camera position is denoted by C, the lower grid location by G_l and the distance between grids by ΔG. The phi coordinate is left out as in these coordinate vectors is it is unused for the grids, and all operations are elementwise.

This process generates an alpha in both the θ and r directions, since we restrict the movement in only one direction we can pick the alpha for that direction, where the other value will be NaN due to division by zero. If this alpha value lies between 0 and 1 everything is fine, and we can simply interpolate using the two already existing grids. If the alpha lies outside the 0-1 range, we need to swap grids, in the same way already done for following pre-determined paths.

With all these procedures implemented, the user can move freely around the black hole. This process works mostly fine, but a noticeable stutter is seen in frames that fall in the second category.

7

Our Method - Stereoscopic Image Generation

Virtual and augmented reality are particularly useful for an immersive experience and can be highly beneficial in an educational context. Nevertheless, it is not clear how to visualize a black-hole phenomenon because of the large distances and natural mismatches in both views due to stereoscopic distortion. Here, we propose a visualization that involves stereoscopic images to produce a 3D impression that is non-photorealistic but is targeted to exaggerate the 3D impression, while ensuring visual comfort by avoiding large stereoscopic mismatches.

To render two images, we need to make a decision about the distance between the eyes. If this distance is chosen too small, both the left and right image will be basically the same, and no depth impression is generated. On the other hand, if this distance is chosen too big the difference in the light paths will create two entirely different images that result in a mismatch and, thus, eliminating the depth impression [12].

Instead, we decided to warp the disk based on the traveled distance to create a fake stereoscopic image. This method allows us to show off the distance to each part of the disk intuitively, without interference of the distortion effect on the image content.

It was chosen to always have the black hole at the center of the scene, and most importantly for it to have zero disparity. This choice was made since the black hole is a point, and what we really see is the shadow caused by relativistic effects. It is also unclear how to deal with the distortion of the background we do see, if the black hole is moved across the image.

If we want the black hole to both be at the center of the scene and have zero disparity, the best option is to target a stereo graphic system which overlays two images over each other, one for each eye. We will use the model described in Pert Kellnhofer in his Applied Image Processing lecture [25] this leads to the situation seen in Figure 7.1.

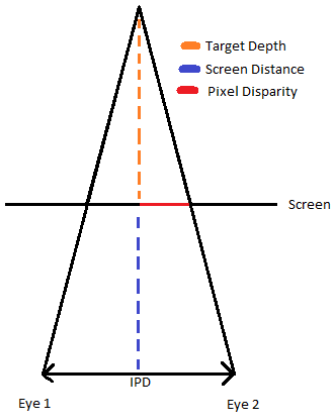


Figure 7.1: A sketch showing the situation if the screen overlays a left and right image over each other. To simulate the object being behind (or in front) of the screen, a pixel disparity is needed.

To calculate the pixel disparity, which is the distance we need to move each pixel, we can use the two similar triangles leading to the formula shown in Equation 7.1.

$$d = \frac{0.5\text{*ipd}}{p} \cdot \frac{z}{z_s + z} \tag{7.1}$$

In Equation 7.1 d is the target pixel disparity, the amount a pixel has to move. ipd Is the inter-pupil distance, or the distance between the pupils, in mm, usually 64 mm is used. p is the size of a pixel in mm, a correction factor as we are looking for a disparity in pixels, while all measurements are in mm. z_s is the distance to the screen in mm. And lastly z is the target depth of the object in mm, the user is asked to give a conversion factor from units R coordinate to mm.

To avoid artifacts from forward warping, and avoiding the search for the correct pixel in backwards warping, it would be ideal if a grid based approach could be used. This method, in which the vertexes of a grid are warped instead of individual pixels, allows the non-local warp function to be applied using forward warping by stretching and squishing a grid to fit the function.

Luckily, a grid is actually available for the most important part of this stereo vision. We can use the grid summary. Each box can be seen as the edge of a quad, resembling a disk segment. if we warp these boxes using the warping function from Equation 7.1, we should get a new disk. In this disk, the quads making it up are no longer aligned with the sample angles, and we need a function for checking if a pixel is in a disk segment, but other than that it just works.

The last obstacle is the celestial background, this is located at infinity and thus needs to move $\frac{0.5\text{*ipd}}{p}$ pixels. It is not immediately clear what this means near the black hole, where the warping is extreme. It was chosen to simply apply this as an offset to the phi coordinate, this gives a reasonable approximation for most of the image, while finding a better solution is left for future work.

With every case dealt with and using the length of the geodesic as a depth estimate we get Figure 7.2.

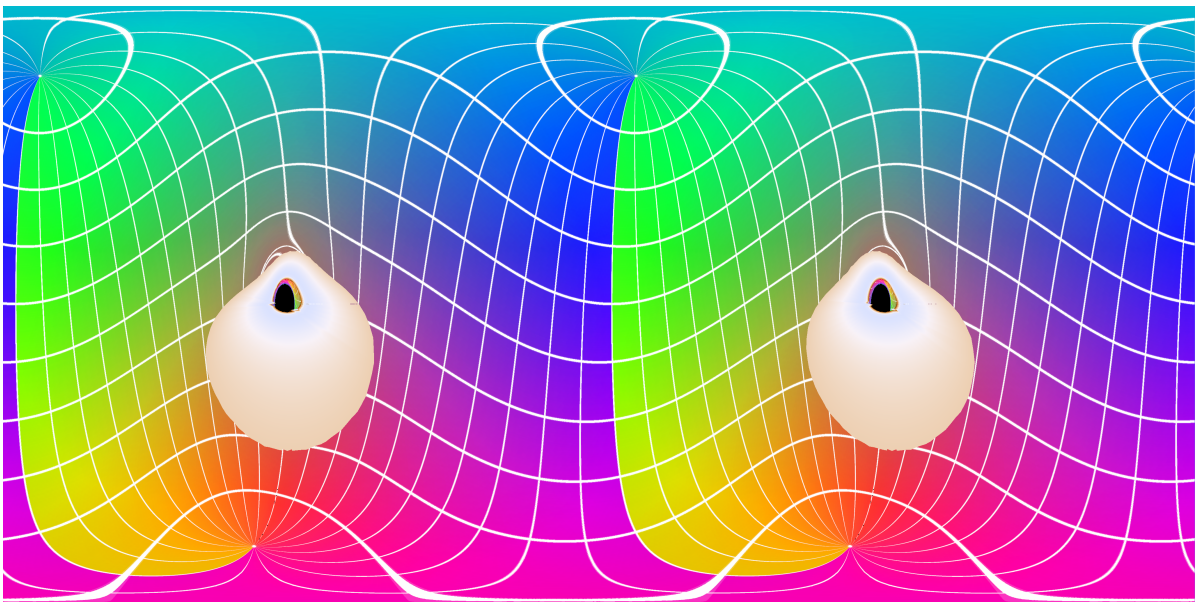


Figure 7.2: Stereo image created using our approach, the image is taken with the camera at $r = 50, \theta = 0.4\pi$ and $\phi = 0$.

The approach is mostly successful in generating a depth impression that feels natural and gives additional insights. Nevertheless, some artifacts remain. It can be seen that the change is about what is expected. The part above the black hole is morphed more, since it is further away.

The asymmetry caused by the rotating black hole is also visible. This asymmetry causes rays travelling on the right side of the black to travel further by deflecting them. This further travel distance is visible in that the right part of the disk is morphed more.

8

Optimizations & Results

In this chapter, we will first follow the optimize-evaluate cycle a bit to show the result of several optimizations in section 8.1. After this, we will also evaluate the performance of the grid summary in reducing grid generation time and its ability to imitate the grid based render in section 8.2.

8.1. Optimizations

The rendering algorithm can be subdivided into two steps, grid generation, where the adaptive grid is generated, and frame generation, where one or more grids are used to create an actual image. In previous work, observers were constrained to a step-wise exploration of the black hole. We allow users to move around the black hole freely. The grids then need to be recomputed at arbitrary positions, making the grid-generation performance a crucial component, and thus the main focus of our optimization effort. The first optimization is achieved by partially offloading grid generation to the GPU. Another optimization we will discuss is the ability to use the faster float datatype instead of doubles.

8.1.1. Grid generation on GPU

Integrating geodesics is a prime task for the GPU. Each geodesic needs to follow the same routine to be traced, and we need to do it with at least 100,000 initial values to get a complete picture of the sky. Previous work relied on a CPU generation, which proved more efficient at the time, yet, with novel hardware and drivers, we can show that a GPU construction becomes beneficial.

Due to the adaptive nature of the grid it is not possible to know which exact positions need to be evaluated, we get the integration jobs in batches, one batch for each grid layer. This batch-like behavior allows us to also experiment with integrating smaller batches on the CPU to avoid overhead from calling the GPU. Since the code for the GPU is compatible with the CPU, we can also use the same code, avoiding code duplication issues.

The experiment was run as follows : generate 51 frames where the camera rotates from $theta = 0.4\pi$ to $\theta = 0.45\pi$, with the other coordinates remaining constant at $r = 50$ and $\phi = 0$. Where possible, an accretion disk was included, ranging from $r = 6$ to $r = 40$. Furthermore, the minimum batch size to use the GPU was varied. The results of this experiment can be seen in Figure 8.1.

As can be seen in Figure 8.1, the speedup of using the GPU is significant, the small improvement for the CPU implementation can be related to a few minor optimizations in the original code.

The amount of rays required to be traced is higher in the current implementation. To reduce some artifacts, the minimum required subdivision has been increased. The user can set this value themselves to adapt to the required accuracy at the cost of performance.

Lastly, we can see that as the fraction of batches on the GPU increases, the time per ray decreases, and it becomes faster right up until 204/255 and 255/255 batches on the GPU. Here, the overhead of the very small last batch for all 51 grids does not seem to have a significant impact. This means that the choice of a minimum batch size for the GPU does not matter, as long as it is small enough.

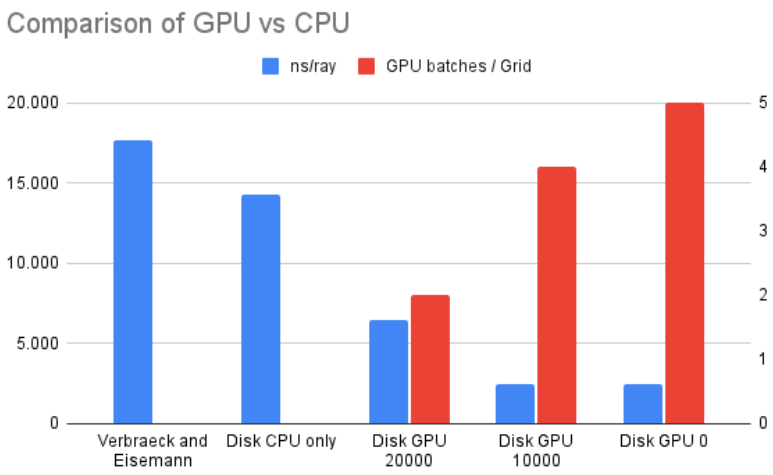


Figure 8.1: Diagram showing the relation between GPU usage for ray integration and the average time per ray. The blue bars show the average time required to trace a ray in ns with the left vertical axis. The red bar show the amount of batches done on the GPU per grid with 5 being all batches as shown to the right axis. The data was averaged from 51 grids and the full dataset is shown in Table A.2.

8.1.2. Floats vs. Doubles

The original implementation, [1], uses double precision floating point numbers, which incur significant performance loss on the GPU. They argue that this is required for enough precision. This might be true in their implementation, however, it can be seen as a parameter that could be used to improve performance.

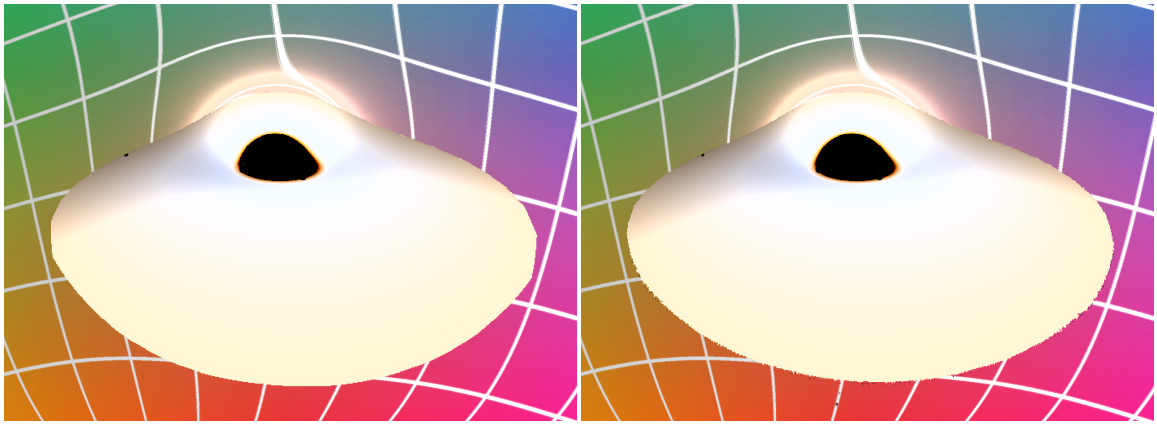
The theoretical performance gain for switching from double to single precision is 64-fold [26]. This performance gain is theoretical, and applies to arithmetic operations only. In our case the performance gain is closer to twofold as shown in Table 8.1.

The same experiment as for the CPU and GPU was used, only now the free variable was the datatype. As all functions are templated the datatype is easy to vary by the user and only requires a recompile after the macro is set.

Datatype	Time (ms)	Rays	Average Time / Ray (μs)
Float	18695	10150325	1.842
Double	25944	10039009	2.584

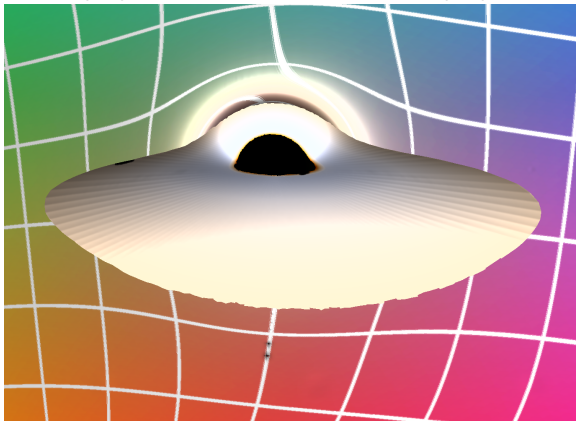
Table 8.1: Table showing the performance changes when changing between doubles and floats. The first column denotes the datatype, the second shown the total time required for all 51 grids in ms. The third column shows the amount of rays traced, and the last column shows the average time required per ray in microseconds.

While the speed-up is significant it obviously is not without drawback, lower accuracy causes artifacts. These artifacts can be seen near the edge of the disk. The edge of the disk is not clean at all, as seen in Figure 8.2b. A full comparison between doubles and float is seen in Figure 8.2. The disk summary in this case actually helps to get a cleaner picture. The summary assumes that the disk edges to be well-behaved and smooth, and thus automatically cleans up the edges as shown in Figure 8.2c. It also introduces small artifacts due to the gravitational lensing, but other than that the difference between doubles and floats is often minimal while providing a major performance boost.



(a) Example of using double precision floats for grid generation

(b) Example of using single precision floats for grid generation.



(c) Using floats but showing an interpolated image at same coordinates, The camera moved from $\theta = 0.445$ to $\theta = 0.455$.

Figure 8.2: Comparing the visual results of using float and doubles. The camera is located at $r = 50$, $\theta = 0.45\pi$ and $\phi = 0$. The accretion disk ranges from $r = 6$ to $r = 40$.

8.2. Grid interpolation evaluation

Grid interpolation (chapter 5) can be seen as another form of optimization. It relates to the number of grids that are generated, which can be controlled by the user. A relatively simple experiment was chosen to test the performance. The camera moved from $R = 50, \theta = 0.4$ and $\phi = 0$ to $R = 50, \theta = 0.45$ and $\phi = 0$ over the course of 200 frames. The 360 texture is 4096×2160, while the minimum grid level is 2 and the maximum level is 11. The results of this can be seen in Figure 8.3 and Table A.1.

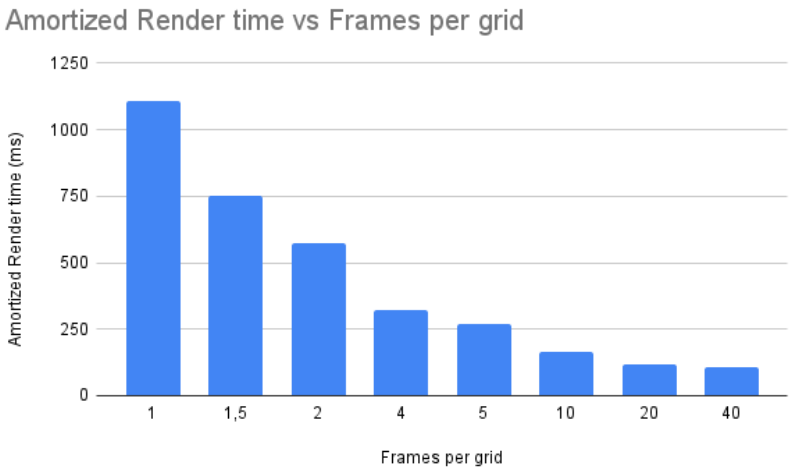
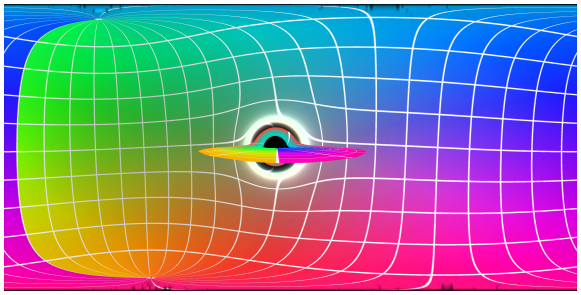


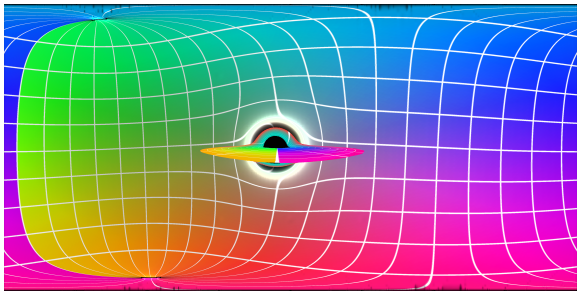
Figure 8.3: graph showing data for running the program, generating frames for the viewer using grid interpolation. The camera moves from $R = 50, \theta = 0.4$ and $\phi = 0$ to $R = 50, \theta = 0.45$ and $\phi = 0$. The 360 texture is 4096×2160, while the minimum grid level is 2 and the maximum level is 11. The axis shows the amount of frames being generated per grid, while the y-axis is the time used rendering one frame amortized over all 200 frames.

In Figure 8.3 it can be seen that, for lower amounts of frames per grid, the relation between the render time and the time per grid is pretty much linear; using a grid for twice as long halves the render time. However, we run into diminishing returns when keeping a grid for too long. the grid generation time is around 1100ms, which means that amortized over several frames, the costs become negigible in comparison to the rendering of the frame, which is around 60ms.

To compare the quality of the generated image, we investigate the scene in Figure 5.6. The image will be generated with the camera at $r = 50, \theta = 0.485\pi$ and $\phi = 0$, with the disk ranging from $r = 6$ to $r = 40$. The images were first converted to grayscale before computing the difference. The result is shown in Figure 8.4.



(a) An example of the disk as it would be generated using a grid



(b) An example of the disk as it would be generated by interpolating two grids

(c) The absolute difference between the two images

Figure 8.4: A comparison of an image generated from a grid directly, and one from two interpolated grids. The images used are the same as in Figure 5.1.

The most apparent difference is that the lines parallel to the coordinate axis do not line up. This is to be expected, as the interpolation only approximates the grid result. It is especially pronounced near the top, where each pixel represents a smaller section of the 360 image and is thus much more error-prone.

An interesting observation can be made at the bottom section of the disk, which shows that the shape of the back of the disk is distorted (made visible by the large area of high difference). This can be accredited to the interpolation of the disk size. It assumes the disk sections disappear linearly between grids, which is not the case in reality, and a more suitable form of interpolation might mitigate this issue.

9

Discussion & conclusion

9.1. Discussion

Comparing the overall shape of the accretion disk to previous work, it seems to concur very well. James et al. [4] show a similar image as generated by us with the coordinate grid. The same can be said for other attempts [1].

However, if we look at the work by Luminet [6] and, Fukue and Yokoyama [15], it can be seen that the actual temperatures/colors do not align with previous work. This is most likely because our calculation of the relativistic effects is unsuitable, or too approximate. This discrepancy is also a given as a recommendation for future work in subsection 9.2.1.

Nevertheless, the time required to generate an image has been drastically reduced by about 2 times. Mostly due to offloading grid generation to the GPU.

9.2. Future Work

9.2.1. Fixing relativistic effects

While the relativistic effects added to the disk are founded in theory, they do not seem to agree with other work. This was mentioned for both the Doppler and relativistic redshifts, however lensing is similarly an approximation. All information required to calculate the redshift and lensing should be available since it simply cannot depend on more than the black hole coordinates, the intersection point with the disk, and the incoming ray direction. Our main focus was on the accelerated computation and thus lack the deep relativistic knowledge required to analyze this problem, but someone with more experience in general relativity might be able to adjust the model.

9.2.2. Free movement

The free movement works, but could be refined. The approach which allows for the most movement without additional grid generation, bilinear interpolation, was deemed too big of a change to be implemented late in the project. Secondly, artifacts show up when moving to the extreme ranges. When moving near the equator, the same issues that plague the disk interpolation are very clearly visible. when moving close to the poles artifacts appear and, to separate grids completely the black hole center is no longer dynamically located, however when moving very far away, this black hole center might not be close enough to the actual center. All these issues remain for future work

9.2.3. Stereoscopic images

Time constraints also had effect on this feature. The first and most obvious point is that it works with the depth map from integration but not with the depth calculated if we ignore relativistic effects, and it does not work when grids need to be interpolated. Besides this, an entire master thesis could be dedicated to this feature, as it is not clear what to do with the disparity of the background, or otherwise the offset of the black hole, as both of these intricate problems do not have a clear answer yet.

9.3. Conclusion

The goal of this thesis was to generate real time imagery of a physically correct accretion disk, while, at least, maintaining previously achieved speeds. This first part of the goal has for the most part been achieved as the relativistic effects are not entirely correct but the proper temperature, shape, and data to calculate the effects are. The second part of the disk about maintaining speeds has definitely been achieved and is even surpassed as significant speed-ups were achieved. Lastly, the extra features were also added in the form of free movement and stereo vision. All in all, we believe that this thesis has helped the field of relativistic renderers by introducing a novel way to render the accretion disk, and showing, that this way of rendering allows for significant speed-up. Furthermore, several clear ways of continuing this research exists, namely, in refining free-movement around the black hole and stereo vision.

A

Optimization Data

A.1. Grid interpolation performance

Frames per grid	Amortized Render time (ms)
1	1105,885
1,5	752,03
2	574,17
4	323,935
5	271,1
10	166,87
20	115,13
40	108,66

Table A.1: Table showing data for running the program, generating frames for the viewer using grid interpolation. The camera moves from $R = 50, \theta = 0.4$ and $\phi = 0$ to $R = 50, \theta = 0.45$ and $\phi = 0$. The 360 texture is 4096×2160, while the minimum grid level is 2 and the maximum level is 11. The first column shows the amount of frames being generated per grid, while the second is the time used rendering one frame amortized over all 200 frames.

A.2. GPU vs CPU integration

Table for the GPU and CPU integration data

Version	Time (ms)	Rays	Average Time / Ray (μs)	Integration Batches	GPU Batches
Grid min 6, Verbreack and Eisemann.[1]	14909	844729	17.6494	561	0
Grid min 7, Disk CPU only	67087	4694028	14.292	255	0
Grid min 7, Disk GPU 20000	30389	4683528	6.4885	255	102
Grid min 7, Disk GPU 10000	11534	4694128	2.4571	255	204
Grid min 7, Disk GPU 0	11525	4683528	2.4607	255	255

Table A.2: Table showing the data of experimenting with running the integration on GPU for generating 51 grids. The left most column denotes the method used the first is the method of Verbraeck and Eisemann [1], the later rows are for various minimum amounts of batch-size required for the GPU. Verbraeck and Eisemann used a minimum grid level of 6, while our method is using 7 for this experiment. The second column shows the total amount of time used in ms to create all grids. The third column shows the total amount of rays traced. the fourth column shows the resulting average time per ray. and the fifth and sixth show the total amount of integration batches and how much of them were integrated on the GPU respectively.

Bibliography

[1] A. Verbraeck and E. Eisemann, "Interactive black-hole visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 796–805, 2021.

[2] M. OpenCourseWare, "General relativity lecture 22: Black holes," 2020. MIT OpenCourseWare. Accessed on: September 20, 2023.

[3] T. E. H. T. C. et al., "First m87 event horizon telescope results. i. the shadow of the supermassive black hole," *The Astrophysical Journal Letters*, vol. 875, p. L1, apr 2019.

[4] O. James, E. von Tunzelmann, P. Franklin, and K. S. Thorne, "Gravitational lensing by spinning black holes in astrophysics, and in the movie interstellar," *Classical and Quantum Gravity*, vol. 32, p. 065001, feb 2015.

[5] A. Verbraeck, "A fast general relativity raytracing algorithm," *TU Delft educational repository*, 2017.

[6] J. P. Luminet, "Image of a spherical black hole with thin accretion disk.," *Astronomy and Astrophysics*, vol. 75, pp. 228–235, May 1979.

[7] T. Müller and J. Frauendiener, "Interactive visualization of a thin disc around a schwarzschild black hole," *European Journal of Physics*, vol. 33, p. 955, may 2012.

[8] Y. Yamashita, "Implementing a rasterization framework for a black hole spacetime," *Journal of Information Processing*, vol. 24, no. 4, pp. 690–699, 2016.

[9] A. Mandal., K. Ayush., and P. Chaudhuri., "Non-linear monte carlo ray tracing for visualizing warped spacetime," in *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - IVAPP*, pp. 76–87, INSTICC, SciTePress, 2021.

[10] T. Müller, C. Schulz, and D. Weiskopf, "Adaptive polygon rendering for interactive visualization in the schwarzschild spacetime," *European Journal of Physics*, vol. 43, p. 015601, oct 2021.

[11] A. Riazuelo, "Seeing relativity-i: Ray tracing in a schwarzschild metric to explore the maximal analytic extension of the metric and making a proper rendering of the stars," *International Journal of Modern Physics D*, vol. 28, no. 02, p. 1950042, 2019.

[12] A. J. S. Hamilton and G. Polhemus, "Stereoscopic visualization in curved spacetime: seeing deep inside a black hole," *New Journal of Physics*, vol. 12, p. 123027, dec 2010.

[13] C. kwan Chan, D. Psaltis, and F. Özel, "GRay: A MASSIVELY PARALLEL GPU-BASED CODE FOR RAY TRACING IN RELATIVISTIC SPACETIMES," *The Astrophysical Journal*, vol. 777, p. 13, oct 2013.

[14] S. U. Viergutz, "Image generation in Kerr geometry. I. Analytical investigations on the stationary emitter-observer problem," *Astronomy and Astrophysics*, vol. 272, p. 355, May 1993.

[15] J. Fukue and T. Yokoyama, "Color photographs of an accretion disk around a black hole," *Astronomical Society of Japan*, vol. 40, pp. 15–24, Jan. 1988.

[16] D. N. Page and K. S. Thorne, "Disk-Accretion onto a Black Hole. Time-Averaged Structure of Accretion Disk," *The Astrophysical Journal*, vol. 191, pp. 499–506, July 1974.

[17] K. S. Thorne, "Disk-Accretion onto a Black Hole. II. Evolution of the Hole," *The Astrophysical Journal*, vol. 191, pp. 507–520, July 1974.

[18] O. James, E. von Tunzelmann, P. Franklin, and K. S. Thorne, "Visualizing interstellarś wormhole," *American Journal of Physics*, vol. 83, pp. 486–499, jun 2015.

[19] A. K. Dubey and A. K. Sen, "Gravitational redshift in kerr field," *Journal of Physics: Conference Series*, vol. 481, p. 012010, mar 2014.

[20] J. Walker, Jan 1996.

[21] CIE, "CIE 1931 colour-matching functions, 2 degree observer," tech. rep., CIE, 2019.

[22] M. Stokes, M. Anderson, S. Chandrasekar, and R. Motta, "A standard default color space for the internet - srgb," Nov 1996.

[23] T. D. Matteo, S. W. Allen, A. C. Fabian, A. S. Wilson, and A. J. Young, "Accretion onto the supermassive black hole in m87," *The Astrophysical Journal*, vol. 582, p. 133, jan 2003.

[24] GLFW Contributors, "GLFW: A multi-platform library for opengl, opengl es, vulkan, window and input." `https://www.glfw.org`, 2021. Accessed: June 23, 2023.

[25] P. Kellnhofer, "Cse4365 applied image processing lecture 4 : Image transformations," 2022. Closed lecture, October 2022.

[26] NVIDIA Corporation, "Nvidia cuda c programming guide." `https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#arithmetic-instructions`, 2021. Accessed: June 23, 2023.