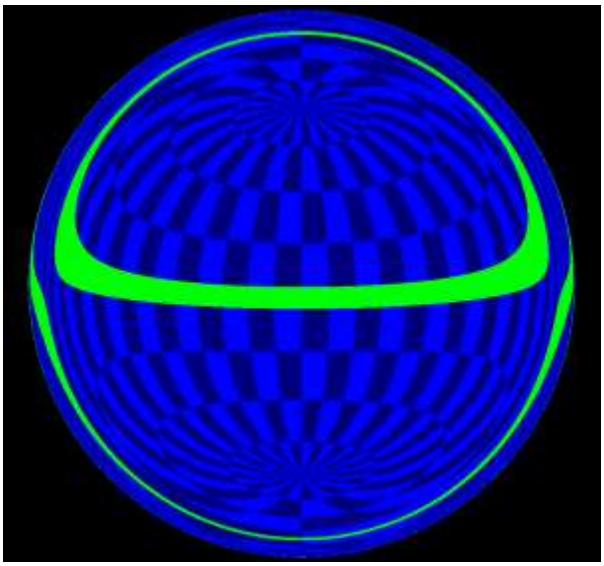
Bachelor Thesis:

Realtime Pseudo Raytracing within Schwarzschild Metric



Author: [redacted]

Supervisor: A.Univ.Prof.Dipl.-Ing.Dr. Michael Schmuckenschläger

January 13, 2017

Contents

1 Introduction **1**
1.1 Abstract 1
1.2 Related work 1

2 Prerequisites **3**

3 Theoretical Considerations **6**

4 Numerical geodesic solving **8**

5 Transformations **11**
5.1 The canonic observer 11
5.2 The central falling observer 14
5.3 Arbitrary Observers 15

6 Four modes of observers **19**
6.1 Canonic-observer 19
6.2 Central freely falling observer 19
6.3 Controlled central freely falling observer 20
6.4 Freely falling observer with angular momentum 20

7 Outlook and results **21**
7.1 Integration in conventional Raytracers 21
7.2 Implementation details 21
7.3 Visual artifacts 22
7.4 Presentation in school 22

1 Introduction

1.1 Abstract

The aim of this bachelor thesis is to intuitively visualize what a black hole would look like if we would move arbitrarily. More formally the goal is to depict a time static sphere centered around a black hole as seen from an arbitrary observer, where the space-time is modeled with the Schwarzschild-metric.

In my approach instead of conventional geodesic solving for each pixel, the rotational symmetry is exploited, which combined with graphic card support allows more than 200 frames per second on ordinary computers. The program is also capable of rendering scenes past the event horizon, which is not handled in most approaches.

As a result the user can seamlessly interact with the simulation and explore the physical effects on his own, which promises potential for educational purposes and visualizations in games and similar applications.

1.2 Related work

This fascinating idea has been around for a long time and since the hardware allowed it, many scientists have implemented visualizations of it in diverse ways. Here are a few examples of previous works:

- In this master thesis the geodesic solving (calculating the orbit of one light ray) happened for each pixel and it took hours for a computer grid to finish one single frame. Though it could render simple but arbitrary objects and it was one of the first of its kind. [8]
- This is a accumulation of different works, including the section 4 "General Relativistic Ray Tracing", where they performed per pixel ray-tracing in Kerr and Schwarzschild-metric in a geometry rich setting. As mentioned one frame typically took about one hour on 56 processors. [4]
- Some further state of the art work can be found at a homepage of the University Stuttgart [1]
- In this example the focus of the live demo lies on platform independence (even works on smartphones) and realtime-rendering. However the implementation suffers from well visible artifacts and lacks advanced user interaction. The author also made a static ray tracer which is similar to most other ones. Partially due to the hardware advances five minutes on one computer are already sufficient for a picture. [7]
- One of the most sophisticated approaches of its time can be found here: Andrew J. S. Hamilton from the University of Colorado implemented a highly versatile ray-tracer . Sadly this piece of software is not free for public access. [5]

To differentiate my approach from others, some unique goal had to be set:

- The main goal is to render a fluent video in **real-time**.
- The resulting program should be **distributable** on most computers and free for access.
- It should be **GPU** supported for high performance.
- It should give almost the same degree of **freedom** as modern first person games.
- Almost all parameters should be modifiable.
- All computations should be sufficiently **optimized**
- Rendering should also work **past the event-horizon** (which many works don´t feature)
- The resulting **quality** should be decent

This work is based on the knowledge presented in the lecture *Schwarzschild Metric* by the supervisor of this paper, held in the winter term of 2015. To achieve this goal we need several considerations and some sacrifices:

- We do not want to solve the geodesic equation for each pixel.
- Therefore the rendered objects have to be spheres.
- The geodesics will be 1D ordinary differential equations.
- Solving about 2000 geodesics on the CPU will yield decent quality.
- The scene must be time static (avoids 4D ray-tracing).
- The pseudo ray-tracing per pixel will be a fixed function pipeline implemented with an Open-GL fragment shader.
- The used coordinates are the standard Schwarzschild ones (t, r, ϕ, θ).
- All computations are based on a defined canonic observer, which cause instable rendering at the event horizon
- Doppler effects are not considered in this work because it can´t be well approximated with RGB data.

2 Prerequisites

This thesis relies on basic notions of general relativity, especially important are the definitions of a tangential space, a metric, measured angles, the restspace of an observer and the geodesic equations.

To be clear all considerations in this thesis are based on the Schwarzschild metric.

Remark 1 (Notation)**.** With the notion e^t, we mean a tangential vector, which points in the direction of increasing t, and not the exponential function. Also dt denotes a projection, that returns the coefficient of e^t for some tangential vector, i.e. $dt(1/2e^t + e^r) = 1/2$.

Definition 1 (Schwarzschild metric)**.** For $(r, t, \phi, \theta) \in (0, \infty) \times (-\infty, \infty) \times [0, 2\pi) \times (-\pi, \pi)$ the Schwarzschild metric in each point is defined as:

$$\langle \cdot, \cdot \rangle := -h(r)dt \otimes dt + \frac{1}{h(r)}dr \otimes dr + r^2(d\theta \otimes d\theta + cos^2(\theta)d\phi \otimes d\phi) \tag{2.1}$$

With $h(r) := 1 - R/r$.

Definition 2 (Interior, exterior and event horizon)**.** For the Schwarzschild metric the domain is categorized by r in three parts:

- Exterior: $r > R$
- Event horizon: $r = R$ (metric undefined here)
- Interior: $r < R$

These definitions correspond with the sign of $h(r)$.

The following definitions are summarized from the lecture notes, for more details, one can refer to the online script [6], Wikipedia articles [3] [2].

Definition 3 (Lorenz product)**.** Let $\langle \cdot, \cdot \rangle$ be an inner product on a vector space. It is an Lorenz product if its index is 1. This is the case if there exists a basis $e_1, ..., e_n$ such that:

- $\langle e_i, e_i \rangle = -1$ for some i
- $\forall j \neq i : \langle e_j, e_j \rangle = 1$
- $\forall k \neq l : \langle e_k, e_l \rangle = 0$

This definition requires a kind of orthonormal basis, where one element results in -1 instead of 1.

Of course the Schwarzschild metric is a Lorenz product in each point (with $r \neq R$), because e^r, e^t, e^ϕ e^θ are mutually orthogonal, either e^r or e^t result in negative sign and proper scaling can be applied. Therefore it is called a Lorenz metric.

Definition 4 (time-, light-, space-like vectors)**.** For a Lorenz product $\langle \cdot, \cdot \rangle$ any vector x (excluding 0) can be categorized in the following groups:

- Time like if: $\langle x, x \rangle < 0$
- Light like if: $\langle x, x \rangle = 0$
- Space like if: $\langle x, x \rangle > 0$

Definition 5 (Time cone)**.** In a Lorenz space we define for a time-like vector x its time cone:

$$C(x) := \{y \; time \; like | \langle x, y \rangle < 0\}$$

Lemma 1. In a Lorenz space for some time like vectors x, y it holds that:

$$\text{either } C(x) = C(y) \text{ or } C(-x) = C(y).$$

Proof: Lemma 1.3.3 etc. [6]

This means that there are two distinct time cones, that do not depend on the choice of the vector, except for its sign. This motivates the following definition:

Definition 6 (Time orientation)**.** For a Lorenz product a time like vector z can chosen as time orientation. For the Schwarzschild metric the time orientation is given by e^t for the exterior and by $-e^r$ for the interior.

All vectors in $C(z)$ are called *future pointed* and all in $C(-z)$ are called *past pointed*. In common sense each particle moves in a future pointed direction as time passes.

Definition 7. A curve k is called an observer if $\langle k', k' \rangle = -1$ in all points.

Definition 8 (Geodesic)**.** Let k be a curve in $(0, \infty) \times (-\infty, \infty) \times [0, 2\pi) \times (-\pi, \pi)$. k is a geodesic if:

$$\text{k is a critical curve of the energy functional } \int \langle k', k' \rangle$$

If $\langle k', k' \rangle = 0$, k is called a null geodesic and k' is a light like tangential vector in each point. These geodesics represent the trajectories of light.

If an observer k is a geodesic, it is called a freely falling observer.

Definition 9 (Instantaneous observer)**.** A tangent vector u_x in some point x, which fulfills $\langle x, x \rangle = -1$, is an **instantaneous observer** and represents an observer in a single point. Also a vector field, that is a instantaneous observer in each point is called a **observer field**.

From an observer field we can derive observers by its integral curves.

Definition 10 (Canonic observer)**.** This definition is exclusive to this thesis.
We want to define a simple observer field on the exterior and interior of the Schwarzschild metric, which is a non continuous expansion of the Schwarzschild observer to the interior.

$$O_{can} := e^t / \sqrt{h(r)} \; for \; r > R, \; -e^r \sqrt{-h(r)} \; for \; r < R \tag{2.2}$$

All calculations will be based on its instantaneous observers.

Definition 11 (Restspace)**.** For an instantaneous observer its restspace is defined as all vectors, which are orthogonal to it. All vectors of the restspace are space like, thus with a proper basis this space is an euclidean space, which will be intensely used in this thesis.

Definition 12 (Measured angle)**.** For some instantaneous observer the measured angle of two tangential vectors is defined as orthogonal projection on its restspace, and consequently applying the metric like a scalar product:

$$cos(\alpha) = \frac{\langle x,y\rangle}{\sqrt{\langle x,x\rangle\langle y,y\rangle}}$$

Remark 2. The Schwarzschild metric is singular at $r = R$, but this is only because of the used coordinates. Indeed it can be made regular by the use of Kruskal coordinates. Though as my calculations are based on the Schwarzschild coordinates, this coordinate singularity will still pose a problem. Also at $r = 0$ the metric becomes singular in any coordinates, and visualization is completely impossible.

3 Theoretical Considerations

Given the theoretical concepts the first question that rises is:
"What´s the meaning of looking to the center of the black hole?"
This question is of course trivial in the exterior domain of the metric, but not that easy to answer in the interior, because the direction one would expect to be looking into the center $-e^r$ is future pointing and obviously not a suitable direction.

But the precise question depends in fact on the observer: For a given instantaneous observer, which direction on the local rest space should be considered as the direction for looking into the center?
Let´s consider the introduced canonic observer in the Schwarzschild-metric in t, r. If $r > R$, looking away from the black hole means that we are seeing light with positive energy and decreasing r. With $r < R$ these rays have a decreasing r and t coordinate as clearly seen from the equations. Therefore looking into the direction e^t means that we are seeing light falling into the black hole, thus $-e^t$ is the direction "towards the center".
Of course this result seems strange, but this is caused by the unnatural choice of the observer.

Definition 13 (Direction of center)**.** We define the direction of center wrt. the canonic observer as vector on its rest space:

$$C_{can} := -e^r\sqrt{h(r)}\ for\ r > R,\ -e^t/\sqrt{-h(r)}\ for\ r < R \tag{3.1}$$

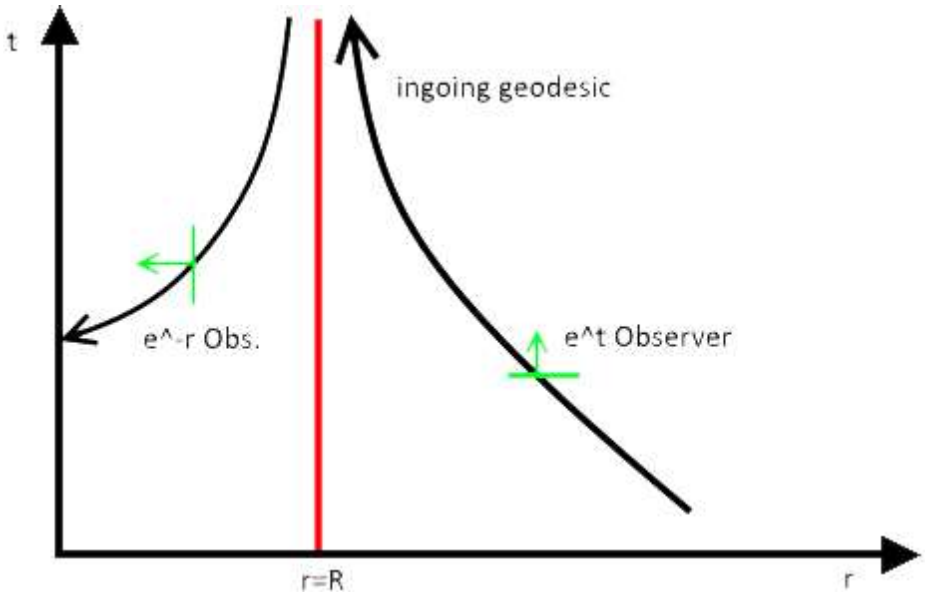


Figure 3.1: The meaning of the center direction. One ingoing light geodesic and its relation to the canonic observer and its rest space is displayed

An alternative approach would be to model the direction of center as a light like vector, which would give the same results for the canonic observer and deviate a bit from the later results for moving observers. However this approach is more focused on the restspace.

The next problem is modeling the screen:
We define that the middle of the screen is the direction where we are looking on the local rest space and that the projection of the screen as plane onto the unit-sphere yields the angles to C_{can}. For each pixel we subtract from the normed direction on the local rest space the observer, which yields a past pointed light ray, which we trace back in time. However to achieve this we still have to do some more transformations.

The most important consideration in this thesis is, that if two light-rays enclose the same angle with C_{can} (wrt. the canonic observer), then their geodesics will behave identically and can be treated symmetrically. This follows from the rotational invariance of the Schwarzschild metric. Therefore we will calculate how the geodesics behave with respect to the angle with C_{can} and then for each pixel we calculate this angle and apply interpolated results from the previous calculations, so we finally know where and if the light-ray hits the sphere.

4 Numerical geodesic solving

In this chapter we assume that we know the radial position r of the observer and for the light ray the energy E, the angular momentum L and if r is increasing or decreasing. From this information we want to determine the orbital angle the photon traversed to get from the sphere to the observer.

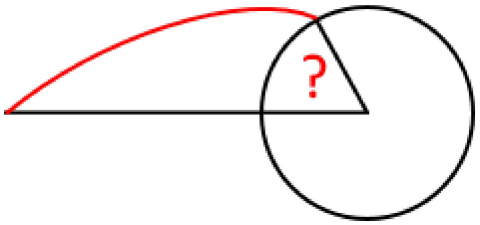


Figure 4.1: Illustration of the angle to be calculated.

For this purpose we will use the orbit equation derived in the lecture:

$$u'' + u = 3Ru^2/2 \tag{4.1}$$

$$\begin{aligned} With\ u(0) &= 1/r \\ u'(0) &= \pm\sqrt{(1/b^2 - h(r)/r^2)} \end{aligned}$$

Here u is $1/r$ as a function depending on ϕ and $b := L/E$ is called impact-parameter.
One might think that using only Schwarzschild-coordinates can´t work for calculating geodesics over the event horizon, but in fact the radial component is completely regular opposed to t the time for a far away Schwarzschild observer, which is indeed singular at the event horizon. Because t is ignored as an assumption, we do not need to use Kruskal-coordinates. However this approach will fail exactly if the instantaneous observer is located at the event horizon.

Because (4.1) can´t be solved with analytic means, we have to apply numeric integration schemes. The two major investigated ones were the **Verlet scheme** and the classical **Runge Kutta 4** method. As it turns out Runge Kutta 4 was better suited.

One might think that the Verlet scheme would profit from the quasi-symplectic nature of the equation, because if the linear part is small, it can be treated as such. As it turns out light-rays tend to hit the surface even before one full orbit and the nonlinear part is too strong in the more extreme cases, so the advantages can´t be seen in the results. Later we will see that the fixed step-width is a severe drawback, which the other Runge-Kutta scheme does not impose. The comparison also shows that the Verlet scheme is only slightly better than a 2nd order Runge Kutta scheme and that RK4 achieves a much higher precision. The following comparison features a step-width of $\pi/100$ and a typical scene where the observer stands close to the black hole and the sphere represents a distant background. The reference scheme was a Runge Kutta scheme with very small step-width, which might not be fully accurate in certain points.

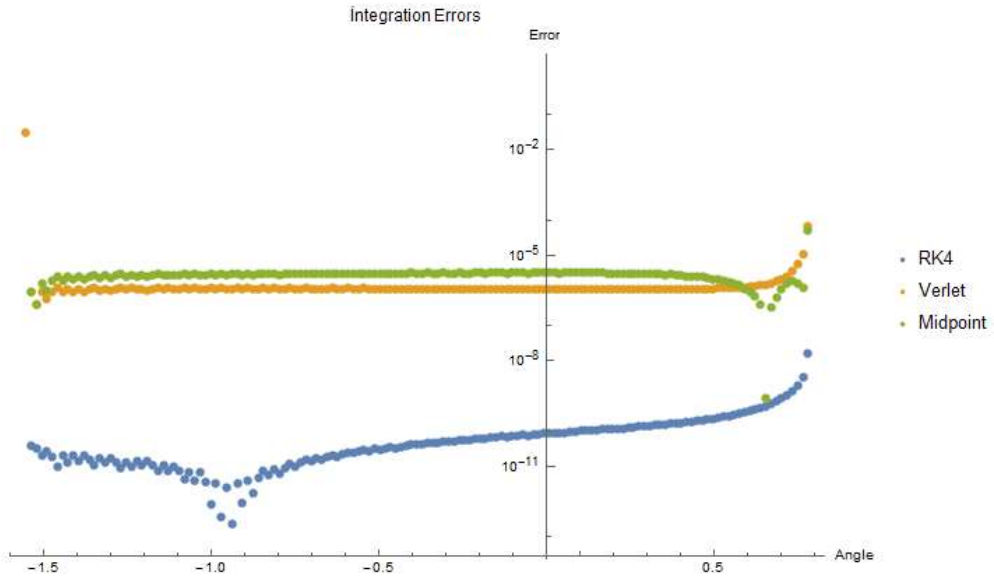


Figure 4.2: A comparison of different integration scheme errors

Other not included test were also performed and gave similar results.

The more interesting question is how to determine the value of ϕ where the function achieves the fixed value of u_{sphere}. Obviously if $u_k < u_{sphere} < u_{k+1}$ or vice versa, then the solution must be in this interval. The approach to simply interpolate the values linearly might sound reasonable, but introduces a big error and causes well visible artifacts as seen in the graphic.

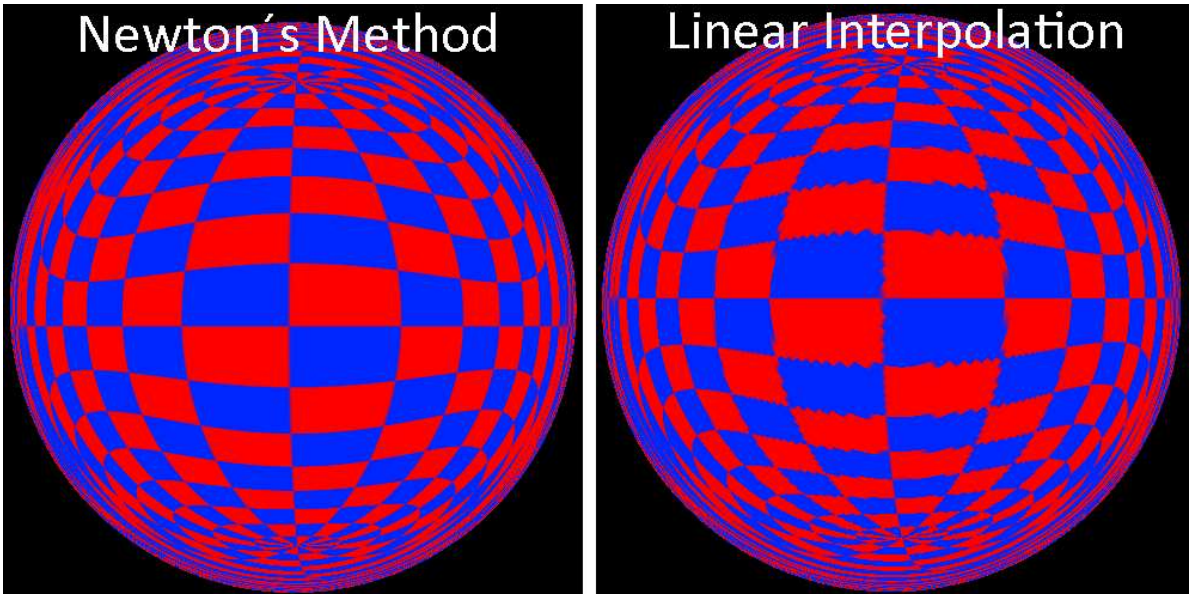


Figure 4.3: Comparison between the two methods, note the visual artifacts in the middle of the right image.

A much better approach to this problem is using **Newton´s method** for finding the zero.

This is very easily implemented if a scheme with variable step width was used (i.e. not Verlet) and the derivative is given. As starting point we choose from u_k, u_{k+1} the one with the bigger derivative for faster convergence. To evaluate u at the estimation for the zero, we redo one RK4 step from u_k to this point and continue with Newton´s method. This method usually yields highly accurate results after roughly 5 iterations, is completely stable opposed to other tried approaches and removes all visual artifacts. One could stop the algorithm by some error bound, however one iteration is so cheap, that a few additional ones would have less impact than considering errors.

In the real implementation we exclude cases where the ray will not hit the surface due to the analysis after (44) in the lecture script [6] and the fact, that rays with negative energy inside the black hole may not come from the exterior.

To achieve decent quality we use 2000 sampled geodesics that uniformly subdivide the interval $[0, \pi]$. Using a high number is important to minimize a effect, when a sampled geodesic changes from hitting the sphere to landing in the singularity. This results in the ring around the black hole suddenly turning black, leading to a "vibration" of its apparent size.

In the final implementation a step-width of $\pi/100$ was used to guaranty still accurate (for human eyes) solutions and high performance. Slow computation times with way smaller step widths showed it was necessary to do these optimizations to achieve fluid frame-rate on most computers, where the computation time for one frame should be below 16 ms for 60 fps.

By testing on an older computer with DDR2 Ram instead of DDR3, a notable slowdown for the calculation of the geodesics could be detected, which suggests that RAM access is a bottleneck of the current implementation, which could be further optimized by loading the heap allocated array into the processor cache. But this is out of the scope of this thesis.

5 Transformations

5.1 The canonic observer

We first want to model a transformation pipeline for the case of the canonic observer ($\propto e^t$ exterior, e^{-r} interior). As mentioned before, we will always identify light-geodesics as unit-vectors on the rest space, therefore we will introduce a right-handed local coordinate system x, y, z on this space and we will need an associated polar coordinate system.

Definition 14 (Associated Polar Coordinates)**.** For unit-vectors in a given coordinate system (x, y, z) we associate a polar coordinate system (r, ϕ, θ). Because r is always 1, we will denote simply (ϕ, θ). For the associated polar coordinate system the following identities must hold:

$$(0, 0) \hat{=} (1, 0, 0)$$
$$(0, \pi) \hat{=} (0, 0, 1)$$

We will further make no differentiation between vectors represented in their original coordinate system and in their associated polar coordinate system.

Definition 15 (Introduced camera like coordinates)**.** For a given coordinate system (x, y, z) a unit vector (ϕ, θ) (in the notion of definition 14), which is not equal to $(\pm 1, 0, 0)$, introduces a coordinate system (x', y', z') which satisfies:

$$e^{z'} = (\phi, \theta)$$
$$e^{x'} = (\phi, \theta - \pi/2)$$
$$e^{y'} = (\phi + \pi/2, 0)$$

Here the right side is in polar coordinates of the original coordinate system.

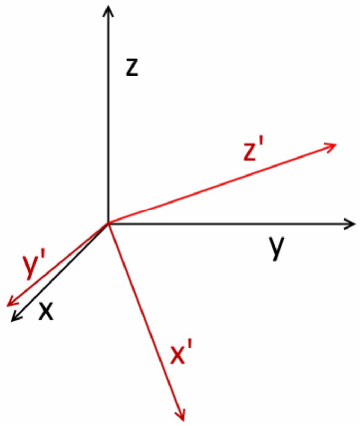


Figure 5.1.1: Example for coordinates introduced by the bright red vector

Remark 3. We will model a standard coordinate system (x, y, z), which satisfies, that at the global position (r, ϕ, θ) the direction of center is equal to $(\phi + \pi, -\theta)$. Further e^z should be orthogonal to e^ϕ and with positive e^θ component.

Alternatively if we associate the global Schwarzschild coordinates without the t component with Cartesian coordinates like in definition 14, then for an infinitely far away observer the introduced above standard coordinates are nothing else than the normalized basis of the tangential space. This way the user will experience a spatial orientation of up and down.

Further the camera is modeled as a coordinate system introduced by a vector in the standard coordinates. As usual in computer graphics a horizontal/vertical movement of the mouse will change the ϕ/θ component.

By simple considerations we can conclude, that a transformation to any coordinates from coordinates introduced by a vector (ϕ, θ) has the following matrix representation (c, s denote *sine* and *cosine* respectively):

$$\begin{pmatrix} c_\phi c_{\theta-\pi/2} & c_{\phi+\pi/2} & c_\phi c_\theta \\ s_\phi c_{\theta-\pi/2} & s_{\phi+\pi/2} & s_\phi c_\theta \\ s_{\theta-\pi/2} & 0 & s_\theta \end{pmatrix}$$

To conclude, we now can perform all necessary transformations on the rest space of the canonic observer by just one matrix multiplication. As we arrive with the angle to C_{can} we can use the results from chapter 4 for transition to the position on the sphere. The results are again given in a coordinate system introduced by the ϕ and θ component of the observer´s position and with its y coordinate inverted.

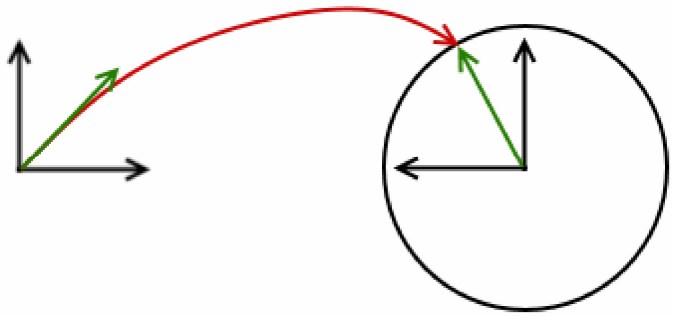


Figure 5.1.2: The transition from the coordinates introduced by the direction of center to the global coordinates on the sphere, note the change in orientation.

Therefore after inverting y and applying the corresponding matrix we arrive at points on the sphere in the Schwarzschild coordinates. The final color of the pixel is then determined by reading the texture, which should be given as a rectangular projection.

Finally we must consider the coordinates of pixels on the screen. In Open-GL these are given by normalized coordinates with the left lower corner being $(-1, -1)$ and the upper right corner being $(1, 1)$. First we multiply the x (width) coordinate with the aspect ratio of the window, next we multiply (x, y) with $\tan(FOV/2)$ to model the field of view (which describes the angle between the middle upper and middle lower points of the screen). From this setup we can derive polar coordinates by projecting the plane onto the sphere.

$$(x, y) \to (atan_{OGL}(y, x), \arctan \|(x, y)\|)$$

$atan_{OGL}$ is a function provided by Open GL, that takes two values and determines their polar angle.

As next step we transform to the coordinates introduced by the camera (for which x is pointing down and y left)

$$(x, y, z) \to (-y, -x, z)$$

In summary this is the modeled transformation pipeline:

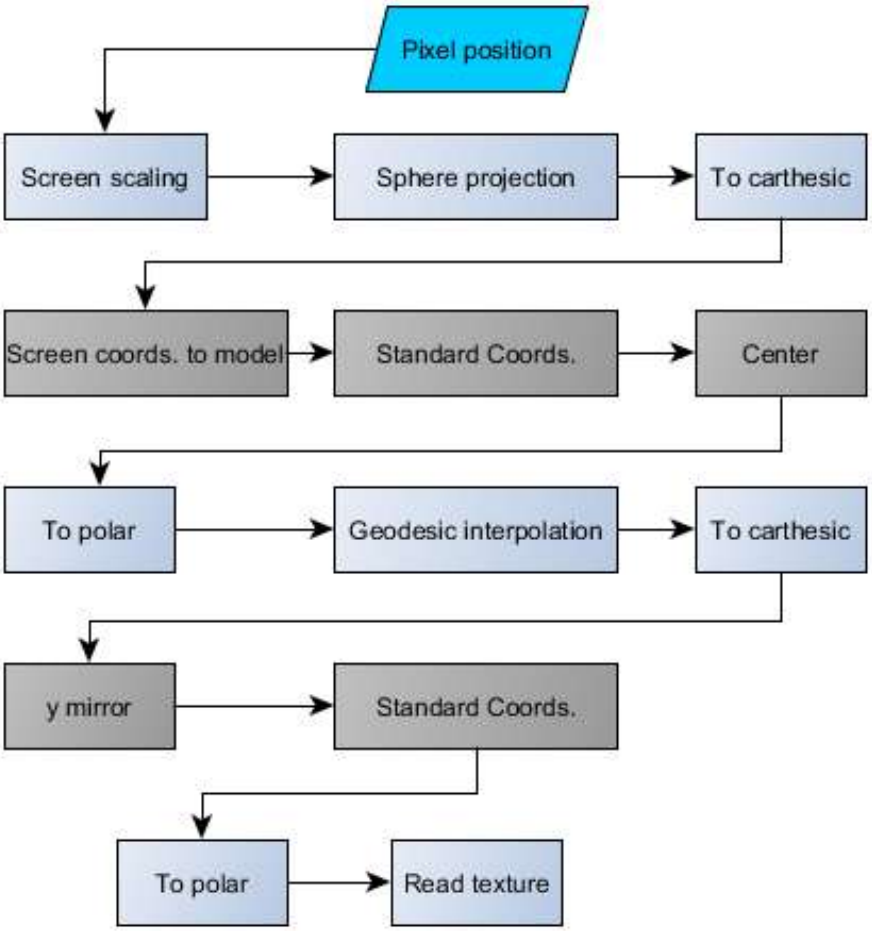


Figure 5.1.3: The simple GPU transformation pipeline, gray blocks are realized as matrix

This transformation pipeline seems a bit excessive, but if all matrices are multiplied in advance the overall effort is:

- 2 matrix multiplications
- 5 times switching from carthesic to polar coordinates or vice versa
- once interpolating a one dimensional function (using a texture)
- reading a texture

This workload for each pixel is arguably manageable by todays graphic hardware.

5.2 The central falling observer

In this section we want to expand the previous transformation pipeline to the case where the observer is falling directly into the black hole, which is the case with $L = 0$.

As commonly found in the literature [4, (1)] the effects of speed only cause a simple distortion of the image, which drags the appearance of the surrounding into the direction one is moving. From the definition of measured angles this formula can be derived, which describes the correspondence of the measured angles between a light ray and the direction of movement with respect to the canonic observer (θ') and the moving observer (θ):

$$\sin\theta' = \frac{\sin\theta - \beta}{1 - \beta * \sin\theta} \tag{5.2.1}$$

Therefore we just need to apply this transformation before interpolating the results of the geodesics. As we transform from the rest space of our moving observer (O_{moving}) to the one of the canonic observer, we need to measure their relative speed:

$$\beta = \sqrt{\frac{\psi - 1}{\psi}} \tag{5.2.2}$$

$$with\ \psi := - < O_{can}, O_{moving} >^2 \tag{5.2.3}$$

$$\psi = d_t(O_{moving})^2 * h(r) \text{ for the exterior}$$

$$\psi = \frac{d_r(O_{moving})^2}{-h(r)} \text{ for the interior}$$

So for this case the aberration can be easily implemented in the transformation pipeline. This is done by applying the aberration just before the geodesic transformation, where z is already in the direction of movement and θ measures the enclosed angle.

Please note that the definition of ψ is no common convention.

This will however cause some instabilities at the event horizon where the canonic observer will see the universe under a very small angle and a very small portion of the sampled geodesics hit the surface. When the aberration happens these few samples will be used for a much bigger area, resulting in visual artifacts.

5.3 Arbitrary Observers

In this section we want to apply the results from the previous section to arbitrary observers. For simplification we assume that the instantaneous observer is located in the equatorial plane and moves within it clockwise, i.e. the coefficient of e^θ is 0 and positive for e^ϕ. The only thing we need to do to apply the aberration is applying an equatorial rotation from the coordinate system oriented towards the center to the direction of movement and after transformation a rotation back. But one has to take care because those two angles are not the same for the canonic observer and the moving observer.

The direction of movement will be defined as the orthogonal projection of O_{moving} on the rest space of O_{can} with normalization, which yields D_{can}. Due to the simple nature of the canonic observer this is just removing either the e^t or the e^r component. For the moving observer we again project D_{can} on its own rest space and normalize, yielding D_{moving}.

Because all four vectors now lie in one plane the resulting light-rays will be the same:

$$O_{can} + D_{can} \propto O_{moving} + D_{moving}$$

This result follows from the fact that both observers are future oriented. The different scaling of the light rays is related to the Doppler Effect, but this is not in the scope of this work.

The angle between the direction of center C_{can} and D_{can} with respect to O_{can} is:

$$\cos\hat{\alpha} =< C_{can}, D_{can} >$$

For a observer $O_{moving} = (a, b, c, 0)$ this is equal to:

$$\cos\hat{\alpha} = \frac{-b}{\sqrt{\psi - 1}\sqrt{h(r)}} \quad for\ the\ exterior \tag{5.3.4}$$

$$\cos\hat{\alpha} = \frac{-a\sqrt{-h(r)}}{\sqrt{\psi - 1}} \quad for\ the\ interior \tag{5.3.5}$$

Due to the assumptions of orientation and movement in the equatorial plane the rotation matrix, which transforms from the direction of movement to the center, is:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & c_{\hat{\alpha}} & -s_{\hat{\alpha}} \\ 0 & s_{\hat{\alpha}} & c_{\hat{\alpha}} \end{pmatrix}$$

For the rotation on the rest space of O_{moving} the calculations are a bit more complicated. But still we just need the angle between the direction of center and the direction of movement.

$$\cos\alpha =< C_{moving}, D_{moving} >= -sign(h(r))\frac{ab}{\sqrt{\psi(\psi - 1)}\sqrt{1 + r^2c^2}} \tag{5.3.6}$$

Because the general orientation is the same for both observers, this matrix rotates from the center to the direction of movement:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & c_\alpha & s_\alpha \\ 0 & -s_\alpha & c_\alpha \end{pmatrix}$$

The very last problem is overcoming the restriction of equatorial movement. We can assume that the movement happens in one plane containing the center. We know that the z coordinate points to the center, thus being contained in this plane. Therefore the task is to rotate in the x, y plane such that the new y is contained in the orbit plane.
For a plane which intersects the equatorial plane at $\phi = 0$ and encloses an angle ζ with it, the angle for this rotation is:

$$\gamma = \zeta cos(\phi) \tag{5.3.7}$$

With ϕ being the respective coordinate of the observer.
So this matrix is used for tilting and untilting.

$$\begin{pmatrix} c_\gamma & s_\gamma & 0 \\ -s_\gamma & c_\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

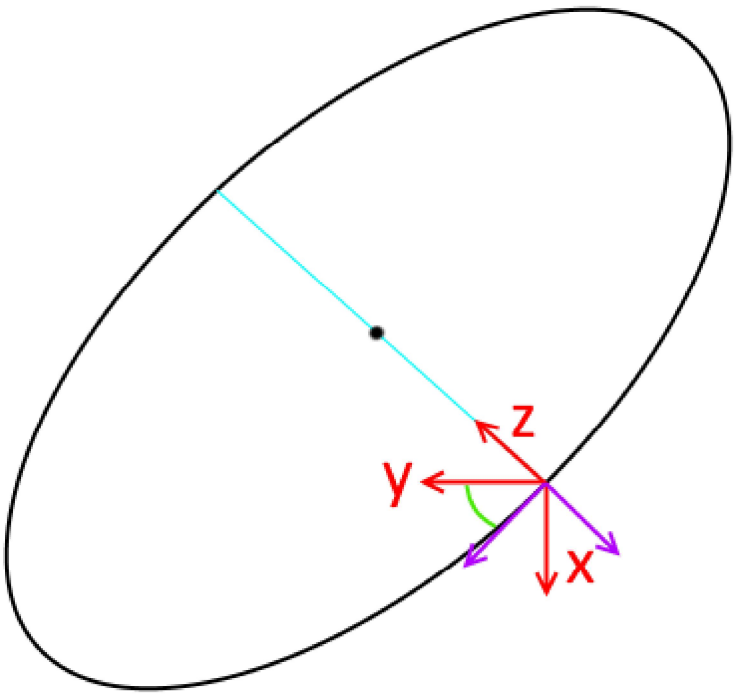


Figure 5.3.4: The needed transformation such that z and y are contained in the plane, which is depicted by a contained circle.

Now we can summarize our final transformation pipeline:

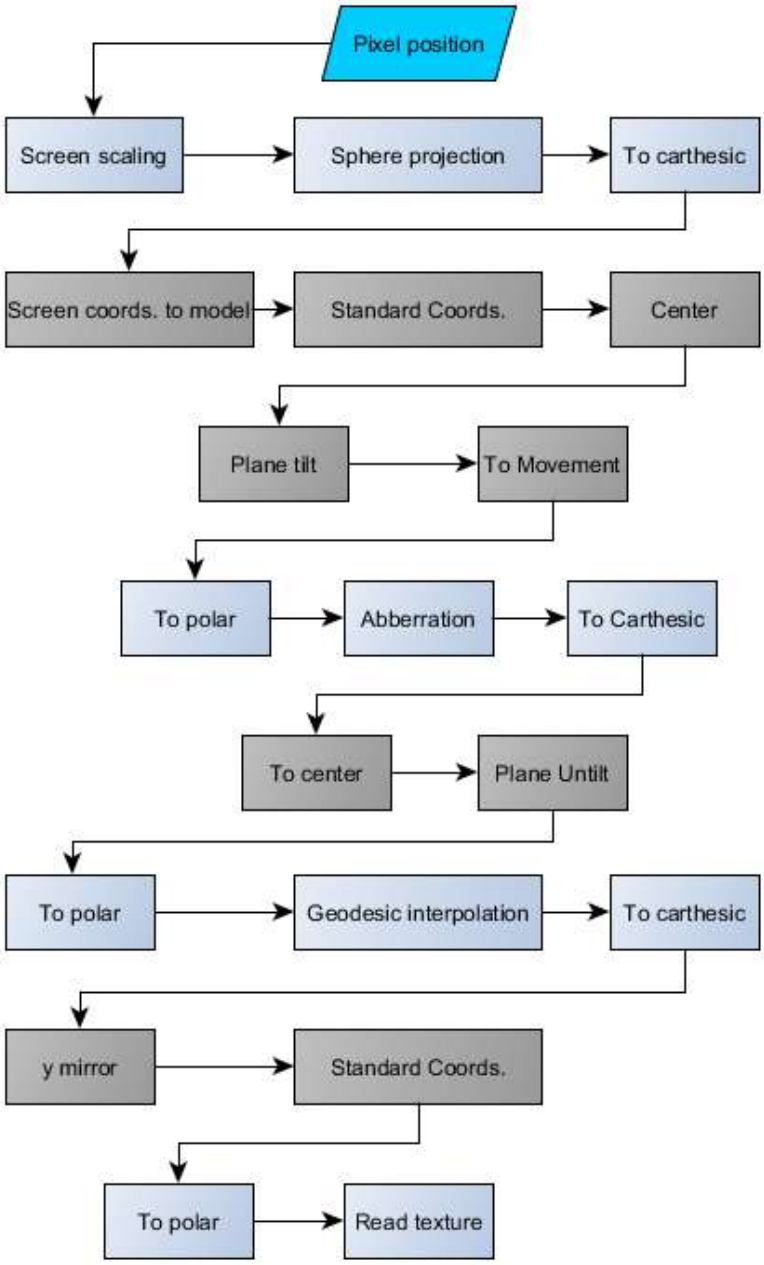


Figure 5.3.5: The full GPU transformation pipeline, gray blocks are realized as matrix

This can be realized with 3 matrix multiplications, 7 polar-carthesic transformations, one simple aberration function and reading interpolated values from a table.
Given a observer that is not in the singularity or on the event horizon, this transformation pipeline can be easily implemented on the GPU. In the next chapter we want to find meaningful modes of interaction for the user.

Remark 4. In Summary: We start with a pixel on the screen, which is contained and transformed in the **restspace of the moving observer**. Then we transition to the **restspace of the canonic observer** and perform some transformations. Finally we follow the geodesic to the **global coordinates on the sphere** and are done after a last transformation.

6 Four modes of observers

In this chapter we present four suggested modes of user interaction with the program. Two of them will allow the user to move around freely, while the other two will force the movement of the spectator to a time-like geodesic. The direction of view can always be controlled by moving the clicked mouse into the direction where the user wants to look.

Each of the modes will yield a instantaneous observer and position at any time, which can be used by calculations of the previous chapters.

6.1 Canonic-observer

Being the least realistic and discontinuous mode it is the most straight-forward one to implement. Time is as always ignored and for the purpose of user controlled movement (r, ϕ, θ) is treated as plain euclidean space, where the user can navigate by using the **WASD** keys for horizontal and **QE** for vertical movement just as in any first-person application.

This mode intuitively captures the act of being stationary outside of the black hole, but while passing the event horizon the stellar background first is reduced to one dot and then expanded again to occupy half the field of view near the singularity.
A further advantage is that this mode has no parameters and can be enabled at any position (i.e. for maneuvering out of the black hole).

6.2 Central freely falling observer

The next simple idea is, that the user cannot manipulate the position, which follows a geodesic without angular momentum in the experienced time of the user to the singularity. This mode starts with $r' = 0$ at some position and advances by Verlet integration of the following equation (derived from the energy-equation by differentiation with τ) :

$$r'' = -\frac{R}{2r^2} \tag{6.2.1}$$

The energy can be easily calculated as $E = \sqrt{h(r_{start})}$ by inserting into the energy-equation. Due to the restriction of starting with r' being 0, this mode can only be enabled in the exterior and only depends on the starting height. Once the geodesic hits the singularity the simulation must end.

It is highly notable that in this mode one won´t notice passing through the event horizon, which relates to the fact, that it is only a coordinate singularity, which is not experienced by a freely falling observer. Even though some image "shock" or ripple can be seen, as the calculations become instable.

6.3 Controlled central freely falling observer

This mode is derived from the previous one, but lets the user freely control his movement like in the Schwarzschild mode. The idea is that for a certain starting height and a fixed position we show the picture which one would see while passing through this point in the previous mode. As we only need to know the energy for reconstructing the spectator this can be easily implemented.
If the user moves to an altitude that is higher than the selected starting height, then we temporarily use the Canonic Observer, which grants a continuous transition.

6.4 Freely falling observer with angular momentum

This mode is the most complicated one and enables the user to orbit around the black hole. We again assume that at the beginning of the orbit r' and θ' are zero and we let the user select the amount of spin. By some euclidean considerations one can determine the plane the orbit will take place in. With this knowledge we can initialize a Runge Kutta 4 scheme for the orbit equation:

$$u'' + u = \frac{R}{2L^2} + \frac{3Ru^2}{2} \tag{6.4.2}$$

The main problem is that this calculates the orbit in (r, ϕ), but not with respect to the perceived time. Therefore we need to estimate the change of ϕ in one time step by using $r^2\phi' = L$ which is again depending on r.
To approximately solve this problem, we calculate $\Delta\phi_0$ with an Euler step, perform one RK4 step to get Δu_0, approximate $\Delta\phi_1$ by using trapezoidal rule and finally calculating Δu_1 by using $\Delta\phi_1$ as step size. This approximation is not too bad for the purpose of this program, but if u' becomes big it is not an accurate scheme.

An approach that could be used is to mostly use the energy equation instead and in the regions where u' is close to 0 , we use the above scheme which remains regular at that point. However this was not implemented, so the used procedure experiences some troubles close to the singularity, where it circles an infinite number of times, but hits $r = 0$ in finite time. Even though, this is barely noticeable to the user, because the end happens very fast.

Again this mode can only be enabled in the exterior. An adaption that lets the user move around freely is barely possible, because at the same height the observer could be falling or rising and for stable orbits there´s no meaningful expansion into the interior.

7 Outlook and results

7.1 Integration in conventional Raytracers

The presented methods are well suited to be integrated in commonly used raytracers. To do this one could define a sphere where the light should be treated as influenced by the black hole and this domain should not contain any other objects. All rays are checked for collisions with this sphere and in case of collision the incoming angle could be mapped to an outgoing angle somewhere else on the sphere (or to the singularity). The overall effort would be interpolating a one dimensional function, which only depends on the Schwarzschild radius.

7.2 Implementation details

The final implementation was realized with Visual Studio 2015 and runs on most 32-bit Windows Systems. But as there is no real dependence on Visual Studio, the project could be easily ported to Linux and other systems.

For Open GL I used GLEW and FreeGLUT, and for image loading SOIL. As the core functionalities are plain C++ without any libraries and a shader for Open GL, only some formal and GUI changes are needed to move to different Open GL loaders.

Figure 7.2.1: A visual artifact in the movement direction. Program executed with *Intel HD Graphics 520*

7.3 Visual artifacts

In the transformation to polar coordinates there is one evaluation of $arcsin(z)$. With double precision this is usually no problem, but with 16-bit floats on the GPU this can cause troubles if z is close to 1 or -1. Interestingly these artifacts as seen on the picture only occur with the *Intel HD Graphics* [redacted] but not on the *NVidea GeForce* [redacted], indicating that the latter features higher precision for these operations.

Another well visible problem of this program is aliasing, which usually happens with high resolution pictures like the Milky Way panorama (6000x3000). Many stars that are almost just one pixel cause medium noise while the image changes. Sadly techniques like mipmapping that deal with this problem are designed for linear vision and are not available in this setting. Possibly some multi-sampling antialiasing approach could solve the problem.

7.4 Presentation in school

On [redacted] 2016 the program was presented to two classes [redacted] The pupils were in the 6^{th} and the 8^{th} grade (respectively 10 or 12 years in school). After some instructions to the cumbersome GUI most of the pupils were able to use the program independently on their own computers and were quite interested in experimenting.

One common misconception arose: When starting a free fall, it appears that the black hole was getting further away and that the background was coming closer, which is due to the acceleration. Common comments were like "But I´m moving away from the black hole."

In conclusion, presenting it to anyone is mostly feasible, but an intense introduction is required. This could be partly relieved by GUI improvements.

References

[1] Overview of relativistic visualization. available at https://www.visus.uni-stuttgart.de/en/research/visualization-and-visual-analytics/visualization-in-special-and-general-relativity.html, October 2016.

[2] Schwarzschild geodesics on wikipedia, 2016. available at https://en.wikipedia.org/wiki/Schwarzschild_geodesics.

[3] Schwarzschild metric on wikipedia, 2016. available at https://en.wikipedia.org/wiki/Schwarzschild_metric.

[4] Müller T. et al. Visualization in the einstein year 2005: A case study on explanatory and illustrative visualization of relativity and astrophysics, 2005.

[5] Andrew J. S. Hamilton. Inside black hole homepage. available at http://jila.colorado.edu/~ajsh/insidebh/bhfs.html, October 2016.

[6] Schmuckenschläger M. Schwarzschild. available at http://www.quixquax.at/bleichling_php/schwarzschild/html/schwarzschild.html, December 2016. JKU Linz.

[7] Antonelli R. Visualizing a black hole, live demonstration. available at http://spiro.fisica.unipd.it/~antonell/schwarzschild/, October 2016.

[8] Corvin Zahn. Vierdimensionales ray-tracing in einer gekrümmten raumzeit, January 1991. University of Stuttgart.