OPEN ACCESS
IOP Publishing
European Journal of Physics
Eur. J. Phys. **43** (2022) 015601 (22pp)
https://doi.org/10.1088/1361-6404/ac2b36

Adaptive polygon rendering for interactive visualization in the Schwarzschild spacetime

Thomas Müller[1,*], **Christoph Schulz**[2] and **Daniel Weiskopf**[2]

[1] Max Planck Institute for Astronomy/Haus der Astronomie, Königstuhl 17, 69117 Heidelberg, Germany
[2] Visualization Research Center (VISUS), University of Stuttgart, Allmandring 19, 70569 Stuttgart, Germany

E-mail: tmueller@mpia.de, Christoph.Schulz@visus.uni-stuttgart.de and Daniel.Weiskopf@visus.uni-stuttgart.de

Received 19 July 2021, revised 15 September 2021
Accepted for publication 29 September 2021
Published 28 October 2021

CrossMark

Abstract
Interactive visualization is a valuable tool for introductory or advanced courses in general relativity as well as for public outreach to provide a deeper understanding of the visual implications due to curved spacetime. In particular, the extreme case of a black hole where the curvature becomes so strong that even light cannot escape, benefits from an interactive visualization where students can investigate the distortion effects by moving objects around. However, the most commonly used technique of four-dimensional general-relativistic ray tracing is still too slow for interactive frame rates. Therefore, we propose an efficient and adaptive polygon rendering method that takes light deflection and light travel time into account. An additional advantage of this method is that it provides a natural demonstration of how multiple images occur and how light travel time affects them. Finally, we present our method using three example scenes: a triangle passing behind a black hole, a sphere orbiting a black hole and an accretion disk with different inclination angles.

Keywords: general relativity, Schwarzschild spacetime, interactive visualization

(Some figures may appear in colour only in the online journal)

[*] Author to whom any correspondence should be addressed.

1. Introduction

The Schwarzschild spacetime describes the outer space of a static spherically symmetric mass distribution like a star and, in particular, a non-rotating black hole. A consequence of the curved spacetime around such a mass distribution is the deflection of light, which could be proven experimentally for the first time by the Eddington expedition [1] during the solar eclipse in May 1919. While the deflection of light due to the curved spacetime around the Sun is in the order of arcseconds, the bending around a black hole is so strong that light can even travel around it on a circular orbit. But not only that, light can also travel on different paths before reaching the observer, resulting in multiple images of the same object. This has been recently reported by x-ray echoes from behind the supermassive black hole in the nearby Seyfert galaxy I Zwicky 1, as described in Wilkins *et al* [2].

There are different approaches to visually scrutinize the complex nature of the curved spacetime around a black hole and its influence on light rays and the resulting visual impression for professional or educational use. The third-person view looks from outside the spacetime and, for example, makes use of embedding diagrams to understand the curvature itself, [3–5], sector models [6] for a three-dimensional representation, or time- and null-geodesics [7] to understand the motion of particles or light.

Relativistic visualization from an egocentric (first-person) point of view [8, 9] has become rather popular since the movie 'Interstellar' [10] in 2014. However, a first realistic visualization of an accretion disk around a Schwarzschild black hole was already shown by the hand-drawn illustration of Luminet [11] in 1979.

There are different techniques to visualize general-relativistic effects from the egocentric point of view. The most natural and most accurate one is four-dimensional general-relativistic ray tracing [12–14]. It is the generalization of the standard ray tracing method from computer graphics to curved light rays that also takes into account light travel time, frequency shift, and lensing. Dedicated ray tracing codes, like GYOTO [15], GRay [16], ODYSSEY [17], or the code by Pihajoki *et al* [18], are able to visualize the near environment around a black hole with even more effects.

While standard ray tracing can reach interactive frame rates on today's advanced graphics hardware, the general-relativistic counterpart is still far from such rendering speed, although the use of graphics hardware already leads to a large performance gain [19]. To achieve interactive frame rates for general-relativistic visualization, and thus make it more accessible for educational purposes, various techniques have been developed. A point-based approach [20] makes use of a lookup table (LUT) to find the light ray connecting the observer with the infinitely distant star and calculates also its apparent temperature and brightness. An image-based rendering is quite similar and distorts an infinitely distant background image as seen by an observer close to the black hole [21, 22]. There is an interactive visualization of a thin accretion disk around a Schwarzschild black hole based on the analytic solution to the geodesic equation [23]. Interactive frame rates are also achieved by a combination of adaptive ray casting and interpolation to visualize the distant celestial sky [24].

In this paper, we present a further development of the polygon rendering method toward general-relativistic visualization. Polygon rendering is the de facto standard method in computer graphics to visualize objects modeled by a triangle mesh. Each triangle is mapped from world space onto the camera screen using highly parallized processing pipelines containing affine transformations (translation, scaling, rotation) and a projective transformation (orthographic or perspective). All of these rendering operations can be executed on graphics hardware efficiently, even for very large scenes with millions of triangles. For those interested in a more detailed overview of computer graphics, we refer to textbooks like the ones by Angel and Shreiner [25] or Marschner and Shirley [26], or to online resources, e.g. https://open.gl.

Polygon rendering has already been adapted for special-relativistic visualization, where the apparent distortion due to high relative velocities between observer and object is applied to every single vertex of all triangle meshes. In 'A Slower Speed of Light', based on the Open-Relativity toolkit [27] for the *Unity* game engine, this technique was used in a first-person game-like environment. However, since no adaptive tessellation was used and only the vertices were transformed, low-polygon objects appear partially edged if the relative velocity is quite large.

For general-relativistic polygon rendering, we first create a distorted mesh, taking into account the bending of light and the light travel time, which can then be processed with standard polygon rendering. This has the additional advantage that the apparent distortion of an object can be studied in more detail and also the influence of the light travel time is clearly visible. If the given mesh resolution is too low, adaptive subdivision is applied to improve the apparent curvature of straight edges. Additionally, a simple illumination model enhances the spatial perception of an object.

The structure of the paper is as follows. Section 2 briefly reviews the calculation of light rays in Schwarzschild spacetime and how a light ray can be found connecting two arbitrary points by means of bisection and ray shooting. Section 3 describes the standard polygon rendering technique from computer graphics and shows how it can be modified for general-relativistic situations. For that, we discuss how the information about connecting light rays is stored in a LUT and how adaptive subdivision can be realized. We also describe implementation details and a simple illumination model. In section 4, we present a few examples and compare the rendering times of general-relativistic ray tracing with our new approach in section 5.

Our implementation is based on the Open Graphics Library (OpenGL) and the OpenGL Shading Language (GLSL)[3]. The source code is written in C++ and makes use of the scripting language Lua. It is freely available from https://github.com/tauzero7/GRPolyRen.

2. Light rays in Schwarzschild spacetime

2.1. Spacetime of the Schwarzschild black hole

The spacetime outside a static spherically symmetric mass distribution can be described by the Schwarzschild metric in spherical coordinates $(t, r, \vartheta, \varphi)$. In line element notation, the metric reads

$$\mathrm{d}s^2 = -\left(1 - \frac{r_\mathrm{s}}{r}\right) c^2\,\mathrm{d}t^2 + \frac{\mathrm{d}r^2}{1 - r_s/r} + r^2\left(\mathrm{d}\vartheta^2 + \sin^2(\vartheta)\mathrm{d}\varphi^2\right), \tag{1}$$

with Schwarzschild radius $r_\mathrm{s} = 2GM/c^2$, Newton's gravitational constant G, mass M, and speed-of-light c. This radius marks the event horizon, the boundary where the gravitational pull or the curvature of spacetime is so strong that even light cannot escape. If $M \to 0$, the Schwarzschild metric transforms into the flat Minkowski metric.

For later use, we introduce pseudo-Cartesian coordinates (x, y, z) that are related to the spherical coordinates (r, ϑ, φ) as usual

$$x = r\,\sin\,\vartheta\,\cos\,\varphi, \qquad y = r\,\sin\,\vartheta\,\sin\,\varphi, \qquad z = r\,\cos\,\vartheta. \tag{2}$$

[3] OpenGL 4.6 specification, https://khronos.org/opengl.

Here, 'pseudo' means that the corresponding coordinate axes are not orthogonal to each other like in Euclidean space. That would become more obvious if we transformed equation (1) into Cartesian coordinates.

2.2. Geodesic equations for light rays

A light ray in curved spacetime follows a null geodesic, which is a generalization of a straight line in curved spacetime. Given a spacetime metric as equation (1), the path of a light ray results from the geodesic equations

$$\frac{\mathrm{d}^2x^\mu}{\mathrm{d}\lambda^2}+\Gamma^\mu_{\nu\rho}\frac{\mathrm{d}x^\nu}{\mathrm{d}\lambda}\frac{\mathrm{d}x^\rho}{\mathrm{d}\lambda}=0,\quad \forall\ \mu\in\{0,1,2,3\},\tag{3}$$

with Christoffel symbols $\Gamma^\mu_{\nu\rho}=\Gamma^\mu_{\nu\rho}(g_{\alpha\beta},\partial_\zeta g_{\alpha\beta})$ and affine parameter λ. Note that we use Einstein's convention to sum over identical upper and lower indices. To guarantee a null geodesic, the initial conditions of these second-order differential equations have to fulfill the constraint equation

$$g_{\mu\nu}\frac{\mathrm{d}x^\mu}{\mathrm{d}\lambda}\frac{\mathrm{d}x^\nu}{\mathrm{d}\lambda}=0,\tag{4}$$

with metric coefficients $g_{\mu\nu}$. These can be directly read off from equation (1) in the form of the coefficients in front of the differentials $\mathrm{d}x^\mu$.

Because of the spherical symmetry of the Schwarzschild spacetime, a geodesic remains in the same hyperplane spanned by the black hole, its starting point and its initial direction. Hence, it is sufficient to consider light rays in the $\vartheta=\pi/2$ hyperplane. Then, the geodesic equations for the Schwarzschild metric simplify to

$$\frac{\mathrm{d}^2t}{\mathrm{d}\lambda^2}=-\frac{r_\mathrm{s}}{r(r-r_\mathrm{s})}u^tu^r,\tag{5}$$

$$\frac{\mathrm{d}^2r}{\mathrm{d}\lambda^2}=-\frac{c^2r_\mathrm{s}(r-r_\mathrm{s})}{2r^3}u^tu^t+\frac{r_\mathrm{s}}{2r(r-r_\mathrm{s})}u^ru^r+(r-r_\mathrm{s})\,u^\varphi u^\varphi,\tag{6}$$

$$\frac{\mathrm{d}^2\varphi}{\mathrm{d}\lambda^2}=-\frac{2}{r}u^ru^\varphi,\tag{7}$$

where $u^\mu=\mathrm{d}x^\mu/\mathrm{d}\lambda$, $\mu=\{t,r,\varphi\}$, represents the tangent to the geodesic. The initial conditions to this system can be defined with respect to a local reference system $\mathbf{e}_{(i)}=\mathrm{e}^\mu_{(i)}\partial_\mu$, where

$$\mathbf{e}_{(t)}=\frac{1}{c\sqrt{1-r_\mathrm{s}/r}}\partial_t,\qquad \mathbf{e}_{(r)}=\sqrt{1-\frac{r_\mathrm{s}}{r}}\partial_r,\qquad \mathbf{e}_{(\varphi)}=\frac{1}{r}\partial_\varphi,\tag{8}$$

which represents a locally flat coordinate system with mutually orthogonal basis vectors,

$$\eta_{(i)(j)}=\langle\mathbf{e}_{(i)},\mathbf{e}_{(j)}\rangle=g_{\mu\nu}\mathrm{e}^\mu_{(i)}\mathrm{e}^\nu_{(j)}\quad\forall\ i,j\in\{t,r,\varphi\}.\tag{9}$$

The derivative operators $\partial_\mu\equiv\partial/\partial x^\mu$ represent coordinate directions, and $\eta=\mathrm{diag}(-1,1,1)$ is the flat Minkowski metric for 2 + 1D subspace.

Now, an initial direction $\mathbf{y}$ can be given either with respect to this basis or with respect to their coordinate representation,

$$\mathbf{y}=y^{(t)}\mathbf{e}_{(t)}+y^{(r)}\mathbf{e}_{(r)}+y^{(\varphi)}\mathbf{e}_{(\varphi)}=y^t\partial_t+y^r\partial_r+y^\varphi\partial_\varphi.\tag{10}$$

For a past-directed light ray starting at $r = r_\mathrm{i}$, $\varphi = \varphi_\mathrm{i}$ at time $t = t_\mathrm{i} = 0$ with direction $y^{(t)} = -1$, $y^{(r)} = \cos\xi$, and $y^{(\varphi)} = \sin\xi$, the corresponding coordinate directions read

$$y_\mathrm{i}^t = -\frac{1}{c\sqrt{1 - r_\mathrm{s}/r_\mathrm{i}}}, \qquad y_\mathrm{i}^r = \sqrt{1 - \frac{r_\mathrm{s}}{r_\mathrm{i}}}\cos\xi, \qquad y_\mathrm{i}^\varphi = \frac{\sin\xi}{r_\mathrm{i}}. \tag{11}$$

The initial conditions $(t_\mathrm{i}, r_\mathrm{i}, \varphi_\mathrm{i}, y_\mathrm{i}^t, y_\mathrm{i}^r, y_\mathrm{i}^\varphi)$ can be used to integrate the geodesic equations (5)–(7). An interactive application to study null and timelike geodesics in many different spacetimes is described by Müller [7].

2.3. Shadow of a black hole

By means of the Euler–Lagrangian formalism, see e.g. Rindler [28], the energy balance equation for null geodesics within the $\vartheta = \pi/2$ hyperplane,

$$\frac{1}{2}\dot{r}^2 + V_\mathrm{eff} = \frac{1}{2}\frac{k^2}{c^2}, \tag{12}$$

with the effective potential $V_\mathrm{eff} = (1 - r_\mathrm{s}/r)h^2/(2r^2)$ and $\dot{r} = \mathrm{d}r/\mathrm{d}\lambda$ can be derived. The constants of motion $k = c^2\dot{t}(1 - r_\mathrm{s}/r)$ and $h = r^2\dot{\varphi}$ for an observer located at $r = r_\mathrm{i}$ are given by

$$k = c\sqrt{1 - \frac{r_\mathrm{s}}{r_\mathrm{i}}} \quad \text{and} \quad h = r_\mathrm{i}\sin\xi. \tag{13}$$

As can be easily shown, the effective potential has a maximum at $r_\mathrm{po} = 3r_\mathrm{s}/2$, which equals the photon orbit, where a light ray can travel around the black hole on an unstable circular orbit. The photon orbit also marks the boundary between light rays that merely pass the black hole and those that fall into it. Solving the energy balance equation for a light ray that approaches the photon orbit asymptotically, i.e. $\dot{r} = 0$, yields the corresponding initial angle ξ_crit for an observer at $r = r_\mathrm{i}$, where

$$\sin^2(\xi_\mathrm{crit}) = \frac{27}{4}\frac{r_\mathrm{s}^2}{r_\mathrm{i}^2}\left(1 - \frac{r_\mathrm{s}}{r_\mathrm{i}}\right). \tag{14}$$

The angle ξ_crit also describes the half opening angle of the shadow of the black hole.

2.4. Light ray connecting two arbitrary points

The theory of the previous subsections at hand, we can now integrate a light ray starting at point p with an initial angle $\bar{\xi} = \pi - \xi$. Depending on how close this light ray passes the black hole, it will be bent more or less. As long as the initial angle $\bar{\xi}$ is larger than the critical angle, $\xi_\mathrm{crit} < \bar{\xi} \leqslant \pi$, the light ray will tend toward infinity, $r \to \infty$, for $\lambda \to \infty$. If $0 \leqslant \bar{\xi} < \xi_\mathrm{crit}$, in contrast, the light ray will inevitably plunge into the black hole. Only in the limiting case where $\bar{\xi} = \xi_\mathrm{crit}$, the light ray will approach the circular photon orbit at $r_\mathrm{po} = 3r_\mathrm{s}/2$ asymptotically.

While this backward ray tracing by integrating the geodesic equations is straightforward, finding a light ray connecting two arbitrary points, also called emitter-observer problem, is highly non-trivial in curved spacetimes.

Figure 1 shows the situation where two points p and q have to be connected by a light ray. The most striking feature is that not only one light ray connects the two points, *path 1*, but also another one, *path 2*, which travels around the black hole on the opposite side. In principle, there is an infinite number of light rays connecting these points that orbit the black hole one or

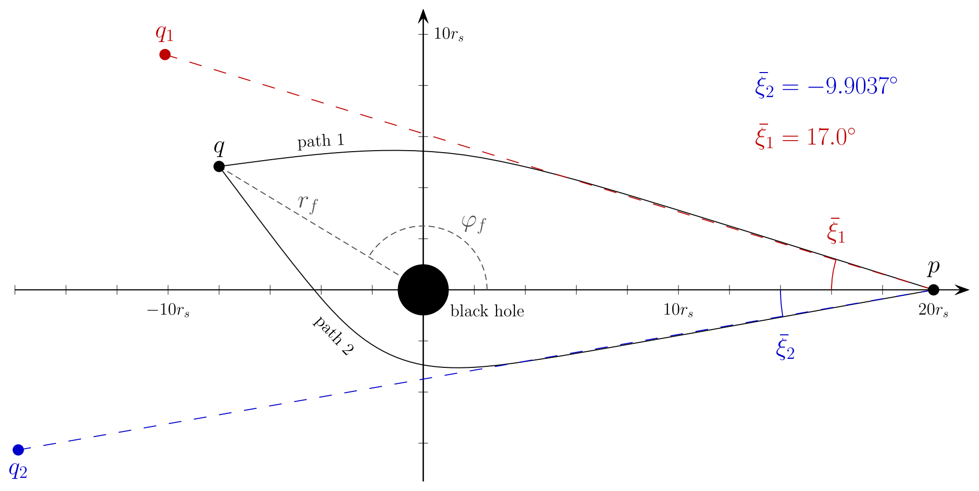


Figure 1. Due to the deflection of light close to a black hole, an observer at point $p(x_p = 20r_\mathrm{s}, y_p = 0)$, will see the point $q(x_q = -8r_\mathrm{s}, y_p = 4.83r_\mathrm{s})$ at the apparent positions q_1 and q_2.

multiple times before reaching the target. But for the rest of this paper, we will concentrate on these two.

Now, if an observer is located at point p, they would see the point q along the directions of the incoming light rays. Thus, q appears at q_1 and q_2. However, the distance is somewhat arbitrary as long as there is no additional information like light intensity, which we will ignore here. As a natural choice, we use the light travel time multiplied by the speed of light to assign a distance. In figure 1, the light travel time for *path 2* is a little bit longer than for *path 1*, and thus, q_2 is slightly farther away from p as q_1. We will use this distance also to enumerate the images as first- and second-order images.

The situation would become even more complicated if one or both points were to move in time. But for simplicity, we consider only points that are either static or move quasi-statically. In the latter case, the points move much more slowly than the speed of light or the light travel time between them, so they can be considered static while searching the connecting light rays.

The spherical symmetry of the Schwarzschild spacetime considerably simplifies the search for a light ray connecting two arbitrary points. Let us assume two points p and q outside the event horizon that are not collinear and have position vectors $\vec{p}$ and $\vec{q}$ in pseudo-Cartesian coordinates. The hyperplane $\mathcal{H}$ containing the light ray between these two points can be spanned by the two base vectors $\vec{e}_1$ and $\vec{e}_2$, see figure 2, where

$$\vec{e}_1 = \frac{\vec{p}}{\|\vec{p}\|}, \qquad \vec{n} = \vec{e}_1 \times \vec{q}, \qquad \vec{e}_2 = \frac{\vec{n} \times \vec{e}_1}{\|\vec{n} \times \vec{e}_1\|}. \tag{15}$$

If we identify $\vec{e}_1$ with the x'-axis and $\vec{e}_2$ with the y'-axis, we end-up with the standard situation of figure 1. In this system, $\vec{q}$ has the coordinates $x' = \langle \vec{e}_1, \vec{q}\rangle$ and $y' = \langle \vec{e}_2, \vec{q}\rangle$, where $\langle \cdot, \cdot \rangle$ is the usual dot product in Euclidean space, and the polar coordinates read $r = \sqrt{x'^2 + y'^2}$ and $\varphi = \arctan(y', x')$.

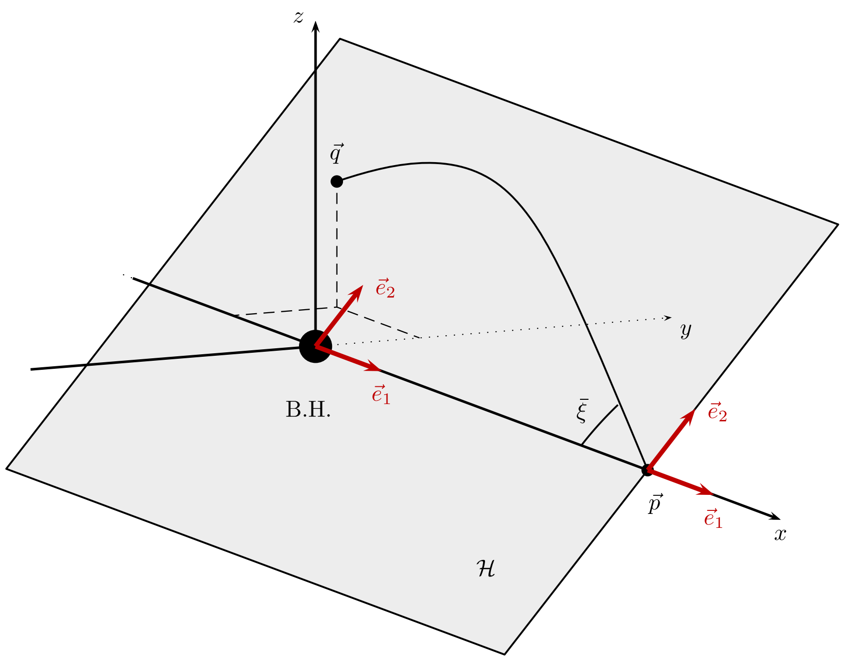


Figure 2. Hyperplane $\mathcal{H}$, spanned by the two base vectors $\vec{e}_1$ and $\vec{e}_2$, contains the light ray between $\vec{p}$ and $\vec{q}$ that has initial angle $\bar{\xi} = \pi - \xi$

2.5. How to find a light ray connecting two arbitrary points

In flat space, a light ray connecting two arbitrary points can be easily found by just drawing a straight line between them. In curved spacetime, however, the light ray must be a solution to the geodesic equations. If there is an analytic solution to the geodesic equations, like for the Schwarzschild spacetime, finding a light ray leads to an implicit equation for the initial direction ξ. The mathematical details rely on elliptic integrals that are difficult to handle and rarely covered in introductory courses on relativity theory. That is why we choose a different approach called *shooting method* to find the connecting light rays.

For that, without loss of generality, we fix the point p to coordinates $(r = r_\text{i}, \varphi = 0)$, and the point q is located at $(r_\text{f}, \varphi_\text{f})$, where $0 \leqslant \varphi_\text{f} \leqslant \pi$ for the first-order image and $\pi < \varphi_\text{f} < 2\pi$ for the second-order image, see figure 1. To find the geodesic connecting p and q, we first select an initial domain $[\xi_0^\text{low}, \xi_0^\text{high}]$ within which we expect the angle ξ that connects both points. For both boundary values as well as for the mean value $\xi_0^\text{mean} = (\xi_0^\text{high} + \xi_0^\text{low})/2$, we integrate a light ray from $\varphi = 0$ (point p) up to $\varphi = \varphi_\text{f}$ (point q) and compare the radial values r_0^low, r_0^high and r_0^mean at φ_f with the target value r_f, see figure 3. If the geodesic does not reach φ_f after a predefined number of integration steps, then the integration will be stopped and an arbitrary large radial value will be assigned.

Depending on where r_f is located, we limit the current domain to the new domain $[\xi_1^\text{low}, \xi_1^\text{high}]$. This iteration will be continued until $|r_n^\text{high} - r_n^\text{low}| < \epsilon_r$ or $|\xi_n^\text{high} - \xi_n^\text{low}| < \epsilon_\xi$ with some thresholds $\epsilon_r \ll 1$ and $\epsilon_\xi \ll 1$.

While this bisection method is rather straight-forward in general, we have to take care about the initial boundary values because of the ambiguity of light rays connecting two points in the Schwarzschild spacetime.

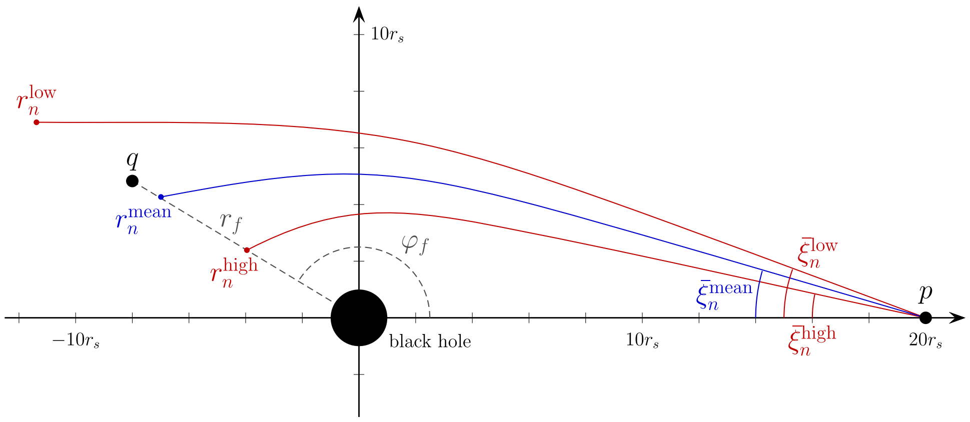


Figure 3. Bisection iteration step 'n' to find the geodesic connecting p and q. The current domain is given by $[\bar{\xi}_n^{\rm low}, \bar{\xi}_n^{\rm high}]$ and the geodesics are integrated up to $\varphi = \varphi_{\rm f}$. As $r_{\rm f}$ is in $[r_n^{\rm mean}, r_n^{\rm low}]$, the new boundary values for iteration step '$n+1$' read $\bar{\xi}_{n+1}^{\rm low} = \bar{\xi}_n^{\rm low}$ and $\bar{\xi}_{n+1}^{\rm high} = \bar{\xi}_n^{\rm mean}$.

In case of the second-order image, the initial boundary values for ξ can be restricted to $[\pi/2, \pi - \xi_{\rm crit}]$, but for the first-order image, the boundaries read $[0, \pi]$ to take into account also the general case where $r_{\rm i}$ could be smaller than $r_{\rm f}$. In both cases, we use a random value in $[\xi_n^{\rm low}, \xi_n^{\rm high}]$ in order to select a mean initial angle $\xi_n^{\rm mean}$ as long as the boundary values do not yield meaningful radial values $r_n^{\rm low}$ or $r_n^{\rm high}$. But then, bisection works as usual.

3. Adaptive polygon rendering

Polygon rendering is the de facto standard rendering method in computer graphics if an object's surface is modeled by a triangle mesh. With dedicated graphics processing units (GPUs), even complex scenes with millions of triangles can be rendered with more than 30 frames per second (fps) for a smooth presentation.

The rendering pipeline of the polygon method consists mainly of three stages: vertex manipulation, rasterization, and pixel generation. While the rasterization is fixed, the other two stages are customizable using shader programs. In brief, the vertex shader is responsible for projecting the triangle mesh from world space to screen space. The rasterizer resolves each triangle in screen space into multiple fragments (pixels with additional attributes), and the fragment shader (FS) takes care of the illumination and coloring of each fragment, which then become pixels.

For our general-relativistic visualization of an object's surface, we have to take the bending of light into account when mapping the triangle mesh onto the virtual screen of an observer. This can essentially be done in two steps. First, every single vertex of the triangle mesh is transformed as discussed in the previous section. Then, the transformed mesh can be mapped into screen space with the standard polygon method. Finally, the mesh is colorized in the FS using standard techniques like Phong-shading. However, as rays from light sources also follow null geodesics, we have to find the geodesic that connects the fragment point and the light source in order to determine the incoming direction of light. As there are multiple images due to the strong bending of light, we have to repeat this procedure for the higher-order images of the object as well.

Since finding a connecting light ray is very time expensive, we generate a LUT in a preprocessing step that stores for every combination of r and φ the corresponding viewing angle ξ and light travel time Δt. Then, we only have to determine the hyperplane that incorporates the observer and the mesh vertex, and then can immediately lookup the necessary parameters.

Another problem that we have to face is that the triangle meshes have to be subdivided, even if they come with high polygon counts in order to approximate the strong apparent curvature due to the bending of light.

3.1. Generate lookup table

As already mentioned in the previous section, we can restrict to light rays in the $\vartheta = \pi/2$ hyperplane $\mathcal{H}$. Thus, for a given observer position r_{obs}, we have to sample $\mathcal{H}$ only in coordinates r and φ. As the curvature of spacetime and, thus, the deflection of light increases the closer an object is to the black hole, we use the inverse radius $x = r_{\text{s}}/r$ to sample the radial direction. We also restrict the radial direction to the domain $r \in [r_{\min}, r_{\max}]$ or $x \in [x_{\min}, x_{\max}]$ with $x_{\min} = r_{\text{s}}/r_{\max}$ and $x_{\max} = r_{\text{s}}/r_{\min}$, where $r_{\min} = r_{\text{s}} + \epsilon$ and $\epsilon > 0$. For the φ direction, we split the LUT into two parts: LUT 1 covers $0 \leqslant \varphi < \pi$ and LUT 2 covers $\pi \leqslant \varphi < 2\pi$.

To solve the geodesic equations for any two points $(r_{\text{obs}}, \varphi = 0)$ and $(r_{\text{f}}, \varphi_{\text{f}})$, we first convert them into a system of first-order equations using the coordinates $(t, r, \varphi, u^t = \mathrm{d}t/\mathrm{d}\lambda, u^r = \mathrm{d}r/\mathrm{d}\lambda, u^\varphi = \mathrm{d}\varphi/\mathrm{d}\lambda)$. We integrate the resulting 6D ordinary differential equation with the initial conditions $(t_{\text{i}} = 0, r_{\text{i}} = r_{\text{obs}}, \varphi_{\text{i}} = 0)$ and $(y^t_{\text{i}}, y^r_{\text{i}}, y^\varphi_{\text{i}})$ from equation (11) by using a Runge–Kutta Cash–Karp integrator with step-size control.

As all sampling points are independent of each other, generating the LUT is trivially parallelizable, e.g. using OpenMP[4].

To use the LUT also for a basic illumination model, we have to store the direction of light $\mathbf{u} = u^r\partial_r + u^\varphi\partial_\varphi$ at the final point, which we transform to Cartesian coordinates by means of the transformation of the derivative operators,

$$\partial_r = \cos\varphi\partial_x + \sin\varphi\partial_y, \qquad \partial_\varphi = -r\sin\varphi\partial_x + r\cos\varphi\partial_y. \tag{16}$$

Thus

$$u^x = u^r\cos\varphi - u^\varphi r\sin\varphi, \qquad u^y = u^r\sin\varphi + u^\varphi r\cos\varphi. \tag{17}$$

To summarize, we have two LUTs for the first- and second-order images, where for each sampling point $(r_{\text{f}}, \varphi_{\text{f}})$, we store the initial direction ξ, the light travel time Δt, and the direction of light (u^x, u^y). Each of the LUTs is uploaded as separate 2D texture (see following section), where we use bilinear interpolation to find the respective values in-between the sampling points.

Figure 4 shows the lookup texture for the initial position $r_{\text{i}} = 20r_{\text{s}}$ and a radial range of $r \in [1.25r_{\text{s}}, 15r_{\text{s}}]$ or $x \in [0.0667, 0.8]$, respectively, where only the angle ξ is shown using a gray-scale map.

3.2. Relativistic polygon rendering

Before we setup the pipeline for the relativistic polygon rendering, we first have a look at some special cases requiring specific consideration. In particular, we study how a single vertical line is mapped onto the virtual image plane of the camera. Figure 5 shows the projection of the line $\overline{p_1p_2}$ defined by the two vertices p_1 and p_2 in case of a flat spacetime (black line) or when it

[4] https://openmp.org.

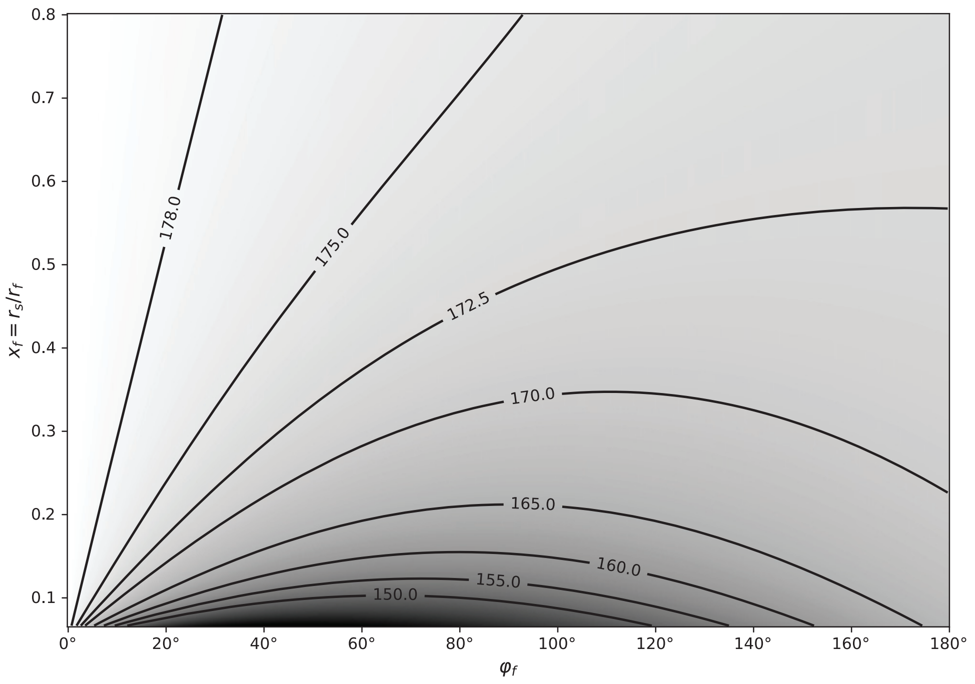


Figure 4. LUT for an initial position $r_{\mathrm{i}} = 20r_{\mathrm{s}}$ and a radial range of $r \in [1.25r_{\mathrm{s}}, 15r_{\mathrm{s}}]$ or $x \in [0.0667, 0.8]$, respectively. The Schwarzschild radius reads $r_{\mathrm{s}} = 2$. The gray value indicates the viewing angle ξ.

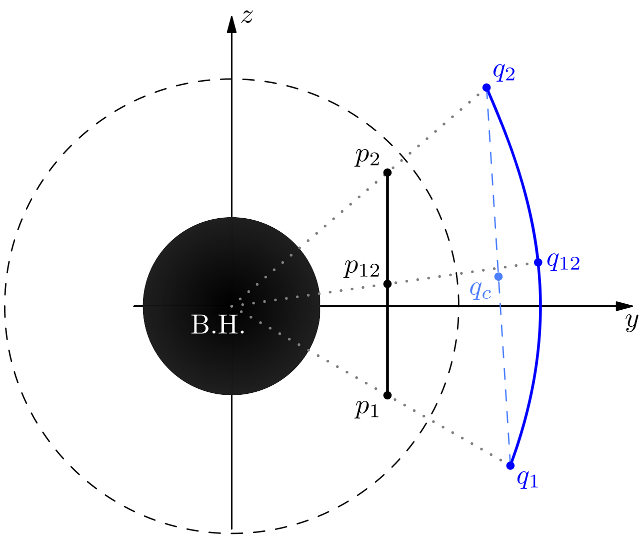


Figure 5. The line $\overline{p_1 p_2}$ is defined by the two vertices $\vec{p}_1 = (-5, 3.5, -2)^{\mathrm{T}}$ and $\vec{p}_2 = (-5, 3.5, 3)^{\mathrm{T}}$ in pseudo-Cartesian coordinates and is apparently distorted to the blue solid curve $\overline{q_1 q_2}$. The center point p_{12} is mapped to q_{12}. The black dashed circle represents the shadow of the black hole, while the radius of the black disk equals the size of the event horizon.

will be apparently distorted due to the black hole spacetime (blue curve). However, as only the two vertices p_1 and p_2 exist and will be mapped to q_1 and q_2, polygon rendering will result in the single straight blueish dashed line connecting q_1 and q_2 that strongly diverges from the correct blue curve.

To circumvent this problem, we could subdivide the line right from the beginning into many segments with multiple vertices, which then could be transformed individually. In fact, the blue curved line is calculated by subdividing the line $\overline{p_1 p_2}$ into 20 segments, and mapping each of the vertices $p_i = p_1 + i/20 \cdot (p_2 - p_1)$ to q_i for $i \in [0, 20]$. In case of a triangle, however, we have to subdivide not only the edges but the whole triangle into many subtriangles, leading to an enormous increase in triangles and vertices. However, this subdivision is superfluous if the triangle is far from the black hole or if it is observed edge-on.

A better strategy is to subdivide a line or a triangle only when necessary. For that, we need a criterion when to subdivide and how much we have to subdivide. It is obvious from figure 5 that the greater the curvature of the actual distorted line in the camera's viewing plane (screen space) is, the more subdivisions we need. A measure of the curvature can be determined using the following procedure. Besides the two vertices p_1 and p_2, we also calculate the mapped point q_{12} that results from the midpoint $p_{12} = (p_1 + p_2)/2$. Then, the distance between the mapped vertex q_{12} and the midpoint $q_c = (q_1 + q_2)/2$ of the two mapped points q_1 and q_2 is divided by the distance between these two points. Thus,

$$\rho = \frac{\|\vec{q}_{12} - \frac{1}{2}(\vec{q}_1 + \vec{q}_2)\|}{\|\vec{q}_2 - \vec{q}_1\|} = \frac{\left\|\mathcal{P}\left(\frac{1}{2}\left(\vec{p}_1 + \vec{p}_2\right)\right) - \frac{1}{2}\left(\mathcal{P}(\vec{p}_1) + \mathcal{P}(\vec{p}_2)\right)\right\|}{\|\mathcal{P}(\vec{p}_2) - \mathcal{P}(\vec{p}_1)\|}, \tag{18}$$

where $\mathcal{P}$ is the mapping from world space to screen space including the bending of light. The number of subdivisions N_{sub} can be heuristically estimated by

$$N_{\text{sub}} = \text{MaxTessLevel} \cdot \text{clamp}(a \cdot \rho^b, 0, 1) \tag{19}$$

with customizable parameters a and b. 'MaxTessLevel' is the maximum number of subdivisions a given graphics board can handle. Hence, we have an upper limit on how often we are able to subdivide a line or triangle edge.

Next, we have to check what happens with a rod that is behind the black hole as shown in figure 6.

Here, the line consists of 11 segments where only the vertices are shown as black dots. Following the procedure of section 2.4 to determine the light rays connecting the rod vertices with the observer at p, we find that for vertices 1–10, the first-order images (red dots) appear above the black hole as expected. For vertex 12, however, the base vector $\vec{e}_2$ points downward, and the first-order image of it appears below the black hole. In contrast, the second-order images (blue dots) for vertices 1–10 appear below the black hole, whereas the second-order image of vertex 12 appears above it. For vertex 11, there is no one single hyperplane as the vertex is collinear with p and the black hole. Rather, vertex 11 degenerates to an Einstein ring. Even though, from the numerical point of view, it is rare for a vertex to lie exactly on the connecting line between observer and black hole, we still have the problem that there are segments or triangle faces crossing this line. Polygon rendering—even with the aforementioned adaptive subdivision procedure—cannot handle this situation where a single point degenerates to a ring. Thus, we have to remove those segments or triangles. While this leads to some loss of image quality close to the black hole, the overall visual appearance is only marginally affected, as we will see in section 4.

Now we can setup our adaptive polygon rendering pipeline. Figure 7 shows the stages of the rendering pipeline that are programmable using GLSL (OpenGL Shading Language).

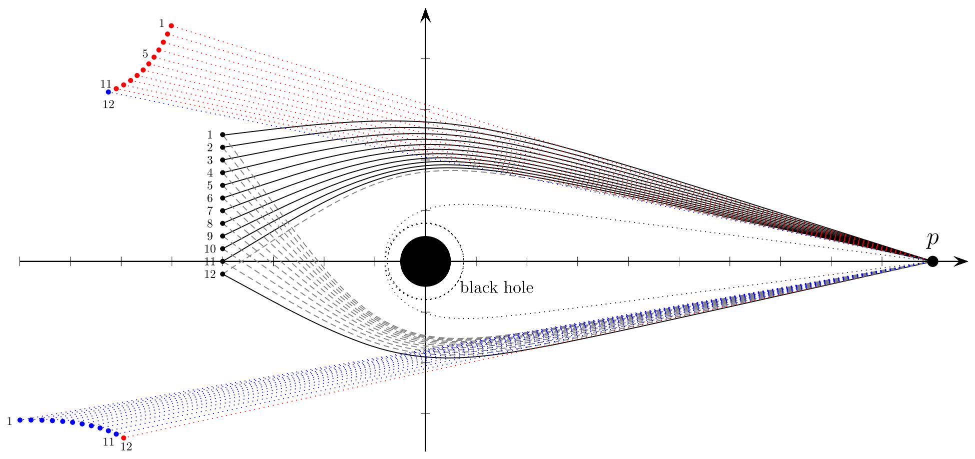


Figure 6. A rod is represented by the black enumerated dots (1–12). The apparent first (second) order positions of these dots are shown as red (blue) dots. The black solid (dashed) lines are the connecting null geodesics of first (second) order. The observer is located at p.

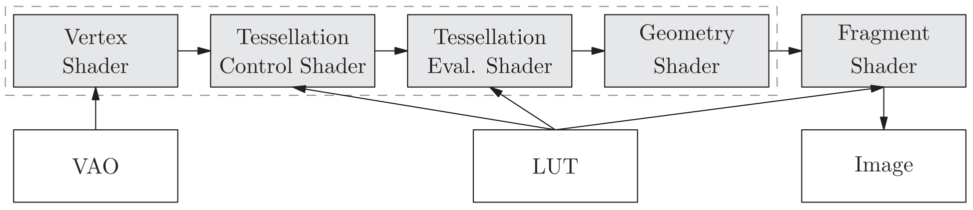


Figure 7. The adaptive polygon rendering pipeline consists of 5 customizable shader stages, the geometry input (VAO = vertex array object), the LUT, and the resulting image output. The rasterizer between geometry and FS is not shown.

The triangle mesh of an object—the positions of the triangle vertices, normal vectors, and texture coordinates—is stored within a vertex array object (VAO) and is uploaded to the GPU once. The two LUTs are stored as 2D floating point textures with the four entries $(\xi, \Delta t, u^x, u^y)$ per texel and bilinear interpolation turned on.

The only task of the vertex shader is to translate, rotate, and scale the object in world coordinates, which equal pseudo-Cartesian coordinates in our case. That means, if we move an object around the black hole, the new positions of the triangle vertices will be calculated in this first stage.

The tessellation control shader (TCS) is responsible for the adaptive subdivision. It has access to all three vertices of a triangle and for each triangle edge, the TCS determines the number of subdivisions according to equation (19) and the aforementioned algorithm, which is why it is necessary to have access to the LUT. The mean value of all three (outer) subdivision levels define the inner subdivision of the triangle face.

The result of the TCS is the subdivision of an original triangle into several subtriangles. These subtriangles now enter the tessellation evaluation shader (TES), where all vertices are mapped to screen space taking into account the bending of light. For the next step, the TES also calculates the screen space coordinates of the undistorted triangle vertices.

The degenerate triangle problem is handled within the geometry shader. Here, the circumferences of a distorted triangle and its undistorted counterpart are compared. If the relation of both circumferences exceeds a customizable threshold, the triangle will be discarded.

Finally, the FS is responsible for colorizing all fragments of a projected triangle in screen space. For that, the FS calculates the interaction between light and material surface, which here is represented by a simple checkerboard texture. The output of the FS is the final image shown on screen.

3.3. Illuminating the mesh

If we apply only the pure colors from the checkerboard pattern, the resulting image looks quite flat. While a fully realistic illumination with lensing effects and frequency shift is out of the scope of this study, we could nevertheless apply simplified diffuse shading. However, we still want to take the bending of light into account. That means that we have to find the null geodesics connecting the point of interest (point to be illuminated) with a point light source, which leads us back to our initial problem. If we put a point light source somewhere on a sphere with the same radius as the observer, we can apply the same procedure as to find the null geodesics between a vertex and the observer using the same LUT as before.

First, we determine the hyperplane $\mathcal{H}$ spanned by the light source at p and the point of interest at q. Next, we set the base vectors $\vec{e}_1$ and $\vec{e}_2$ such that the light source is toward $\vec{e}_1$ and calculate the polar coordinates (r, φ) for q. From the LUT, we can now read the light direction (u^x, u^y) at q, see equation (17). Then, the *outgoing* light direction at q in pseudo-Cartesian coordinates follows from

$$\vec{d} = -\left(u^x\vec{e}_1 + u^y\vec{e}_2\right). \tag{20}$$

To take a distance attenuation into account, we use the light travel time Δt from the LUT multiplied by the speed of light c. Together with the surface normal vector $\vec{n}$ at q, we obtain the light intensity

$$l = l_0\frac{\max(0, \vec{n}\cdot\vec{d})}{|\Delta t\cdot c|^2}, \tag{21}$$

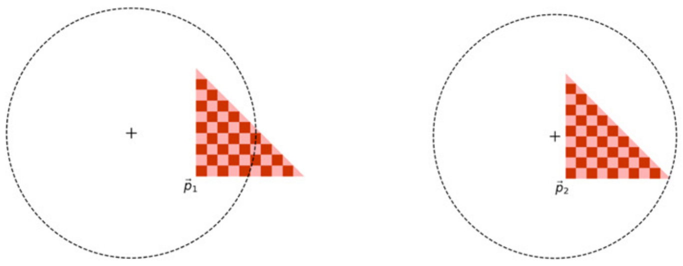
with a customizable light source intensity l_0. This calculation has to be done for both null geodesics connecting q and the light source. Furthermore, the distance attenuation is taken into account only for light rays from the source to the object, and not from the object to the observer. Finally, the Euclidean dot product in equation (21) has to be replaced by the dot product with respect to the Schwarzschild metric.

4. Examples

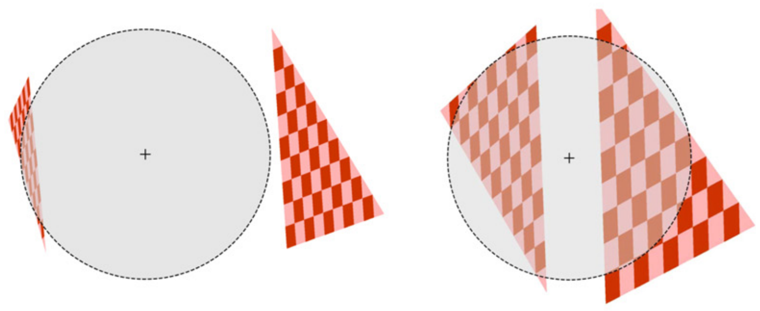
For the following examples, we set the observer to be at $r = 20r_\mathrm{s}, \varphi = 0$, see figure 1, or $x = 20r_\mathrm{s}, y = z = 0$. Then, the half opening angle of the shadow of the black hole can be determined from equation (14), $\xi_\mathrm{crit} \approx 7.274^\circ$.

4.1. Single triangle

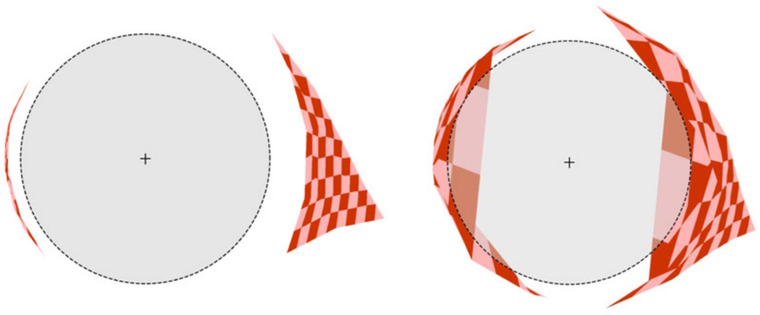
The basic element of a mesh object is a single triangle. As a first example, figure 8 shows a triangle behind a black hole at different positions $\vec{p}$ (lower left corner) and various number of subdivisions (tessLevel). The shadow of the black hole is indicated by the semi-transparent



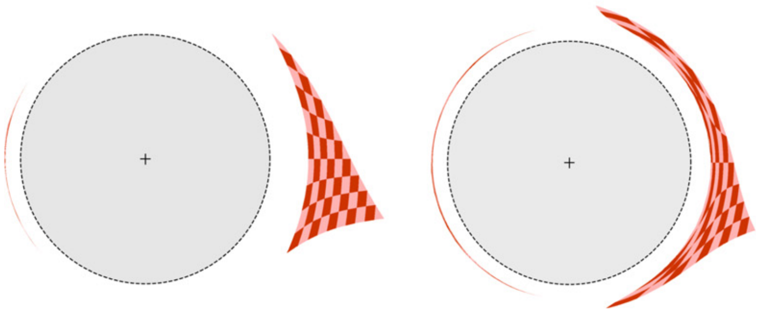
(a) flat spacetime



(b) black hole spacetime, tesslevel = 1



(c) black hole spacetime, tesslevel = 4



(d) black hole spacetime, tesslevel = 64

Figure 8. Single triangle at position $\vec{p}_1$ (left column) and $\vec{p}_2$ (right column) with different subdivisions. The gray-shaded disk represents the shadow of the black hole. The camera's vertical field of view is $\text{fov}_v = 20°$.

gray-shaded disk with dashed circle, and the center of the black hole is marked by a black cross. In the left column the triangle is at $\vec{p}_1 = (-5, 3, -0.5)^\mathrm{T}$, and in the right column it is at $\vec{p}_2 = (-5, 0.5, -0.5)^\mathrm{T}$.

The first row shows the triangle in flat spacetime for comparison. The second to fourth row are in Schwarzschild spacetime where the subdivision of the triangle is given by tessLevel $= \{1, 4, 64\}$. Here, tessLevel $= 1$ means that there is no subdivision. As the triangle consists only of three vertices, only these vertices are transformed and ultimately connected. At tessLevel $=$ 4, a triangle edge is subdivided into at most four segments. While this is already acceptable for the triangle at $\vec{p}_1$, the apparent distortion for the triangle at $\vec{p}_2$ is too strong. For tessLevel $= 64$, the triangle edge close to the black hole shadow is smoothly curved, and also the checkerboard texture appears to be continuously distorted.

4.2. *Sphere orbiting the black hole*

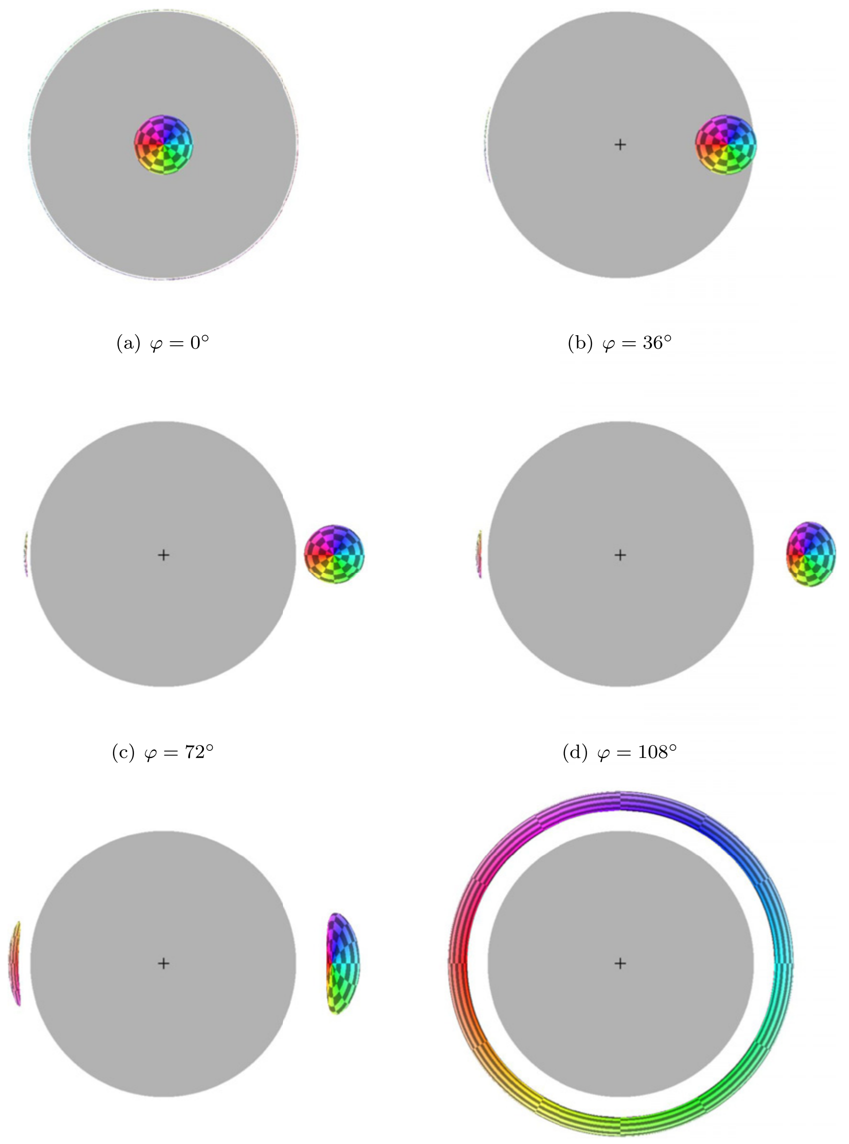
A sphere can be approximated by a geodesic polyhedron—a convex polyhedron made from triangles. One possibility is to start from an icosahedron, where the triangles are subdivided step by step and the new vertices are pushed out radially onto the radius of the sphere. A second possibility is to use a triangle mesh resulting from first creating a spherical grid by latitude and longitude and then dividing each quad into two triangles. The advantage of the second approach are better texture coordinates at the poles even for a low number of triangles. Here, we use 10° separations for latitude and longitude, which results in 1224 triangles.[5]

Figure 9 shows a sphere of radius $R = 0.5 r_\mathrm{s}$ orbiting quasi-statically around the black hole on the last stable circular orbit $r = 3r_\mathrm{s}$. If the sphere is directly between black hole and observer, $\varphi = 0$, figure 9(a), it appears almost undistorted as first-order image. However, a closer look shows that there is also an Einstein ring slightly outside the black hole shadow. This ring (second-order image) is due to light rays that travel around the black hole once before reaching the observer.

When the azimuth angle φ is increasing, figures 9(b)–(e), the first-order image of the sphere becomes more and more distorted and the second-order image grows in size. The colors also show that the second-order image looks similar to a point-mirrored image of the first-order one. When the sphere is directly behind the black hole, figure 9(f), another Einstein ring appears. Here, light rays have to travel only half an orbit around the black hole before reaching the observer. The thickness of the ring originates from light rays that are tangent to both sides of the sphere. Since light rays in case (f) pass the black hole somewhat further away than in case (a), the ring in (f) appears significantly thicker. By the way, in case (f), the next higher-order Einstein ring would be due to light rays traveling around the black hole one and a half times. However, the image resolution is too low for this ring to be resolved.

What we have left out so far is that we do not have the sphere orbiting exactly along the circular orbit around the black hole but give it a tiny offset in z. Otherwise, the poles of the sphere in (a) and (f) would lie exactly on the connecting line between the black hole and the observer, and that would lead to the problem discussed in section 3.2, where a point degenerates to a ring. Furthermore, we defined the large sphere with respect to pseudo-Cartesian coordinates, which leads to additional distortions. A better definition of the sphere would be to use either proper distance or proper light travel time between the center and the surface of the sphere, but that is out of the scope of this study.

[5] The triangle mesh is generated as ‘uvsphere’ in Blender 2.92 with 36 segments and 18 rings.



(a) $\varphi = 0°$ (b) $\varphi = 36°$

(c) $\varphi = 72°$ (d) $\varphi = 108°$

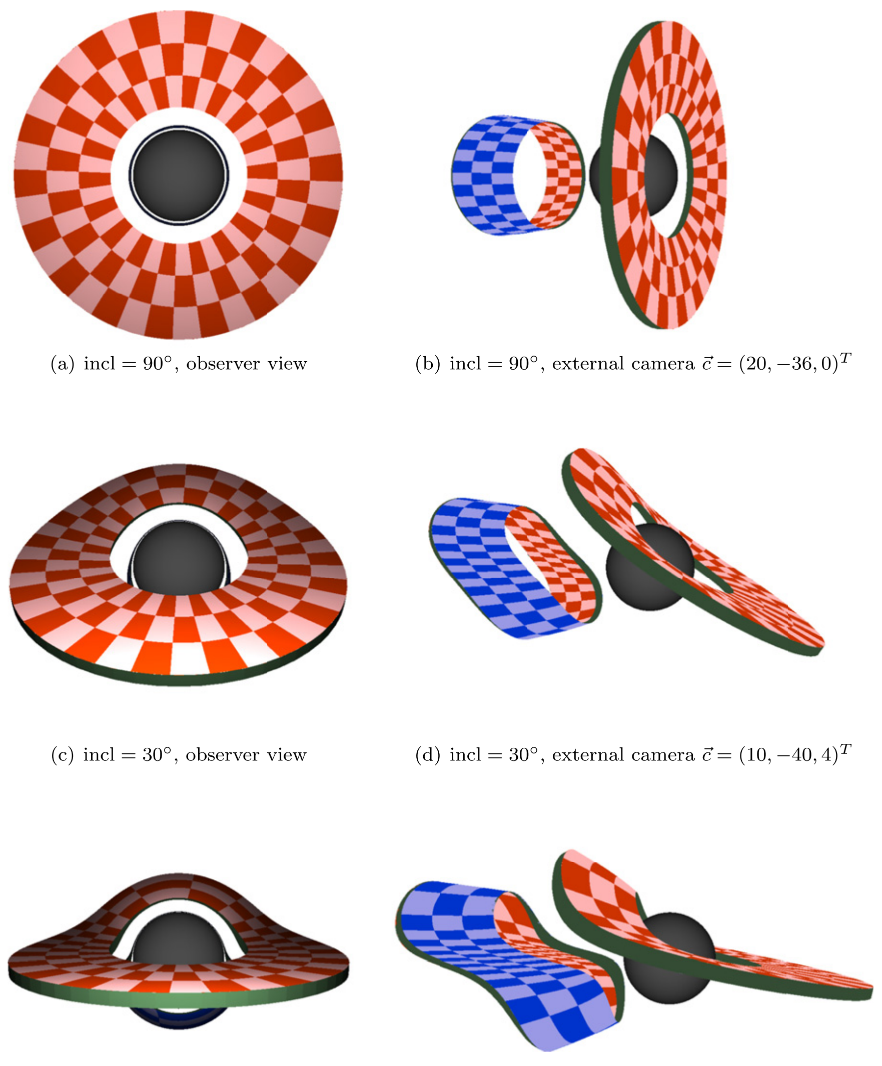
(e) $\varphi = 144°$ (f) $\varphi = 180°$

Figure 9. Sphere orbiting the black hole on the last stable circular orbit, $r = 3r_\mathrm{s}$. Tessellation parameters are set to MaxTessLevel $= 64, a = 5, b = 0.75$, see equation (19). The camera's vertical field of view is $\mathrm{fov}_v = 19°$.

4.3. Accretion disk

The accretion disk in figure 10 has an inner ring radius $r_\mathrm{in} = 3.3r_\mathrm{s}$, an outer radius $r_\mathrm{out} = 8.25r_\mathrm{s}$ and a thickness $d = 0.55r_\mathrm{s}$. It originally consists of 2560 triangles. Top and bottom faces are colored by a polar checkerboard pattern in either reddish or blueish hues.

The left column shows what an observer located at $r_\mathrm{obs} = 20r_\mathrm{s}, \varphi = 0$, will see if a point light source at $r_\mathrm{l} = 20r_\mathrm{s}, \varphi = 0, \vartheta = 80°$ illuminates the disk. In figure 10(a), light hits the disk



(a) incl $= 90°$, observer view

(b) incl $= 90°$, external camera $\vec{c} = (20, -36, 0)^T$

(c) incl $= 30°$, observer view

(d) incl $= 30°$, external camera $\vec{c} = (10, -40, 4)^T$

(e) incl $= 10°$, observer view

(f) incl $= 10°$, external camera $\vec{c} = (0, -40, 4)^T$

Figure 10. Accretion disk with polar checkerboard on the top (red) and bottom (blue) side. The edge has a greenish color. The left column shows the view of the actual observer located at $\vec{p} = (40, 0, 0)^{\mathrm{T}}$ with the light source at $\vec{l} = (39.39, 0, 6.95)^{\mathrm{T}}$. The right column shows the view from an external camera. The black-shaded sphere represents the shadow of the black hole.

nearly perpendicular and the reflected light reaching the observer is also nearly perpendicular. That is why the first-order image (red face) of the disk looks almost uniformly bright. The blueish back side of the disk receives nearly no light, because only light rays that enter the region below the inner radius and that travel around the black hole once can illuminate the back side. Furthermore, these light rays have a longer light travel time and, because of the quadratic distance attenuation, are much weaker. Figures 10(c) and (e) show a similar behavior. A longer

Table 1. Initial angles ξ and light travel times Δt for the inner (a) and (c) and the outer (b) and (d) radii of an infinitely thin disk.

Spacetime		ξ	Δt
Minkowski	(a)	170.6306°	40.5408
	(b)	157.5839°	43.2695
Schwarzschild	(a)	169.0461°	45.3345
	(b)	155.4929°	46.1738
	(c)	172.3350°	66.1959
	(d)	171.9831°	76.8531

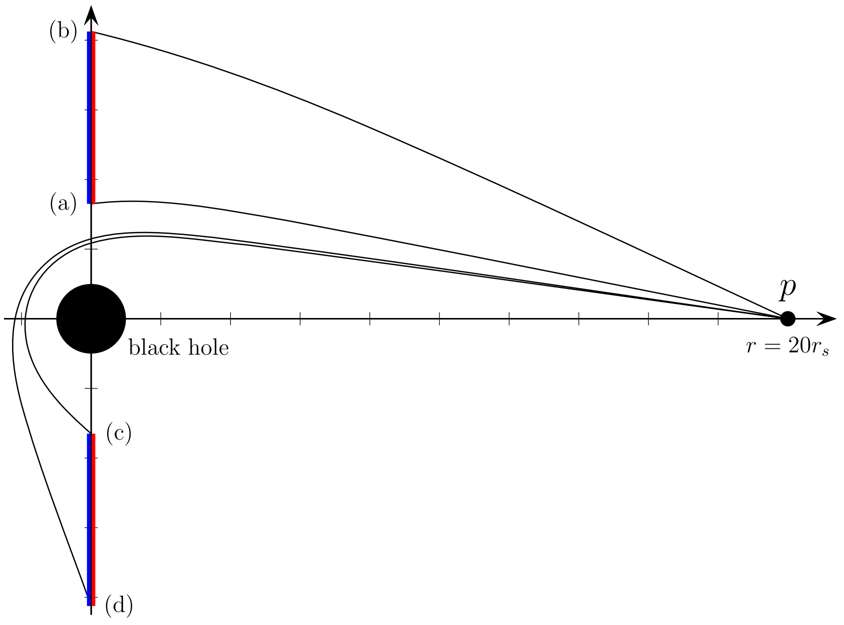


Figure 11. Light rays in Schwarzschild spacetime emitted at the inner and outer edges of a disk (red/blue solid lines). The observer is located at p and has an inclination of 90° to the disk; see also figure 10(a).

light travel time and a larger angle between the incident light direction and the surface normal make the far side of the disk look darker. Note that even the far side of the disk appears to point toward the observer in (e), the inclination angle of the light and the surface normal is almost the same for the entire disk.

The right column of figure 10 shows an abstract view of the situations and cannot be observed in reality. This visualization shows the actual geometry as calculated by the adaptive polygon rendering technique. In particular, the different light travel times that define the distance of a vertex are clearly visible.

In figure 10(b), the inner part of the disk (first-order) is slightly farther away than the outer part. This sounds counterintuitive at first because the outer parts are geometrically further away from the observer than the inner parts. However, gravitational time dilation close to the black hole slows down time. Therefore, light rays take longer to reach the observer the closer they

Table 2. Timings for generating LUT of size $N_r \times N_\varphi$ in seconds.

Observer position	64×128	128×256	256×512
$r_{\text{obs}} = 10r_{\text{s}}$	23.7	86.0	342.4
$r_{\text{obs}} = 20r_{\text{s}}$	34.1	134.0	541.1
$r_{\text{obs}} = 30r_{\text{s}}$	49.0	180.0	714.9

are to the black hole when they were emitted. Table 1 compares the light travel times Δt for the inner and outer disk edges in Schwarzschild and Minkowski spacetime. In flat spacetime, the different distances to the disk edges result in a difference between the light travel times of $\delta t_{\text{M}} = \Delta t_{\text{b}} - \Delta t_{\text{a}} = 2.7287$. In Schwarzschild spacetime, however, the time difference is only $\delta t_{\text{S}} = 0.8393$. Hence, the light travel time to the inner edge is longer than expected from flat spacetime.

The mesh for the second-order image of the disk in figure 10(b), which is responsible for the dark ring just around the black hole's shadow in figure 10(a), is strongly elongated. This shows once more that time dilation has a large influence on the light travel time close to the black hole. But the elongation is also due to a larger geometric distance between a light ray emitted at the outer edge of the disk compared to one emitted at the inner edge, see the light rays connecting p with either (c) or (d) in figure 11.

For lower inclinations, figures 10(d) and (f), light travel times from the bottom side of the disk depend on the point of emission. Thus, also the mesh for the second-order image is distorted.

5. Performance

5.1. Lookup table

The drawback of our general-relativistic polygon rendering technique is the time-expensive precalculation of a LUT for every observer distance. Table 2 summarizes the calculation times for various resolutions and observer distances, and for a radial range given by $[1.25r_{\text{s}}, 15r_{\text{s}}]$. All measurements were performed on an Intel(R) Core(TM) i7-8700 CPU @ 3.20 GHz (Linux) making use of OpenMP parallelization.

As expected, all timings increase linearly with resolution. The increase of computation times with larger distances of the observer to the black hole is just because the integration of light rays take longer and more bisections steps are necessary.

But, even the precalculation takes a lot of time for a fixed observer position, any object can then be moved interactively within the radial range.

5.2. Rendering image sequences

To give an impression of how polygon rendering is pushing general-relativistic visualization toward interactive frame rates, table 3 compares the rendering times for the three example situations of the previous section with either CPU or GPU ray tracing. CPU ray tracing makes use of the MPI-parallelized version of *GeoViS* [14] on the same Intel machine as before. GPU rendering is based on a CUDA implementation of general-relativistic ray tracing similar to GeoViS and was run on an *NVidia Quadro P4000*.

Table 3. Render times for the three example situations of section 4. The number of images calculated for each sequence reads: 300 (triangle), 360 (sphere), 180 (disk).

Example		CPU ray tracing	GPU ray tracing	Polygon rendering
Triangle	640 × 360	36 min 48 s	4 min 34 s	5.2 s
	1280 × 720	138 min 11 s	18 min 20 s	5.2 s
	2560 × 1440	594 min 20 s	72 min 18 s	5.2 s
Sphere	512 × 360	34 min 52 s	4 min 26 s	6.2s
	1024 × 720	138 min 43 s	16 min 57 s	6.2 s
	2048 × 1440	606 min 22 s	66 min 59 s	6.2 s
Disk	400 × 360	12 min 29 s	1 min 31 s	3.2 s
	800 × 720	50 min 44 s	5 min 47 s	3.2 s
	1600 × 1440	204 min 03 s	22 min 05 s	3.2 s

For a better comparison, we render an image sequence for every example. For that, we have implemented a minimal scripting language based on Lua[6] for our polygon renderer.

Actually, the comparison is not quite fair. Polygon rendering is an object space approach, whereas ray tracing is an image space approach. Thus, for the latter case, the larger the image the longer the rendering will take, in general.

Note that most of the time for the polygon rendering is spent on saving the image to file. Even if the tessellation parameters are chosen very high, we still have more than 100 fps.

6. Summary and outlook

Four-dimensional general-relativistic ray tracing yields the most realistic visualization in general relativity. However, because it is very time-consuming, it is not yet suitable for an interactive exploration tool that could help better understand relativistic effects in an educational environment. Polygon rendering, in contrast to ray tracing, is an object space approach, and thus independent of the image resolution, and still the de facto standard in computer graphics to visualize mesh objects. We have demonstrated how to extend standard polygon rendering to be applicable for the interactive visualization of objects within the Schwarzschild spacetime. For that, we calculated how a mesh object would be distorted caused by the bending of light and the light travel time. The distorted mesh could then be rendered with standard polygon rendering.

To accelerate the rendering process, we made use of a LUT that has to be generated in a preprocessing step and stores viewing angle, light travel time, and light direction for every sampled position. So far, we compute the LUT only for a single observer distance, but it could also be extended to cover a range of radial positions. That would also help put a light source at various distances to an object. Furthermore, gravitational lensing and frequency shift could be taken into account, but then also spectral material characteristics of an object would have to be considered. Finally, a virtual environment setting could be realized to study also stereoscopic effects.

Adaptive polygon rendering is not limited to the Schwarzschild spacetime, but it could also be applied to other spacetimes as well. The limiting factors are the LUT, which can be at most

[6] http://lua.org.

three-dimensional, and the feasibility to find connecting light rays if no analytic solution to the geodesic equation is available.

Acknowledgments

We thank Felix Winterhalter for initial prototyping our concept of general-relativistic polygon rendering as part of his BSc thesis [29] and his valuable feedback. Christoph Schulz and Daniel Weiskopf acknowledge funding by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)-project ID 251654672-TRR 161 (Project A01).

ORCID iDs

Thomas Müller https://orcid.org/0000-0002-2003-4465
Christoph Schulz https://orcid.org/0000-0001-5771-3966
Daniel Weiskopf https://orcid.org/0000-0003-1174-1026

References

[1] Dyson F W, Eddington A S and Davidson C IX 1920 A determination of the deflection of light by the sun's gravitational field, from observations made at the total eclipse of May 29, 1919 *Phil. Trans. R. Soc.* A **220** 291–333
[2] Wilkins D R, Gallo L C, Costantini E, Brandt W N and Blandford R D 2021 Light bending and x-ray echoes from behind a supermassive black hole *Nature* **595** 657–60
[3] Flamm L 2015 Republication of: contributions to Einstein's theory of gravitation *Gen. Relativ. Gravit.* **47** 72
[4] Giblin J T Jr, Marolf D and Garvey R 2004 Spacetime embedding diagrams for spherically symmetric black holes *Gen. Relativ. Gravit.* **36** 83–99
[5] Jonsson R M 2005 Visualizing curved spacetime *Am. J. Phys.* **73** 248–60
[6] Zahn C and Kraus U 2014 Sector models-a toolkit for teaching general relativity: I. Curved spaces and spacetimes *Eur. J. Phys.* **35** 055020
[7] Müller T and Grave F 2010 GeodesicViewer-a tool for exploring geodesics in the theory of relativity *Comput. Phys. Commun.* **181** 413–9
[8] Weiskopf D *et al* 2006 Explanatory and illustrative visualization of special and general relativity *IEEE Trans. Visual. Comput. Graph.* **12** 522–34
[9] Kraus U 2007 First-person visualizations of the special and general theory of relativity *Eur. J. Phys.* **29** 1–13
[10] James O, von Tunzelmann E, Franklin P and Thorne K S 2015 Gravitational lensing by spinning black holes in astrophysics, and in the movie Interstellar *Class. Quantum Grav.* **32** 065001
[11] Luminet J-P 1979 Image of a spherical black hole with thin accretion disk *Astron. Astrophys.* **75** 228–35
[12] Weiskopf D 2001 Visualization of four-dimensional spacetimes *PhD Thesis* Eberhard-Karls Universität Tübingen
[13] Müller T and Weiskopf D 2011 General-relativistic visualization *Comput. Sci. Eng.* **13** 64–71
[14] Müller T 2014 GeoViS-relativistic ray tracing in four-dimensional spacetimes *Comput. Phys. Commun.* **185** 2301–8
[15] Vincent F H, Paumard T, Gourgoulhon E and Perrin G 2011 GYOTO: a new general relativistic ray-tracing code *Class. Quantum Grav.* **28** 225011
[16] Chan C-k, Psaltis D and Özel F 2013 GRay: a massively parallel GPU-based code for ray tracing in relativistic spacetimes *Astrophys. J.* **777** 13
[17] Pu H-Y, Yun K, Younsi Z and Yoon S-J 2016 Odyssey: a public GPU-based code for general relativistic radiative transfer in Kerr spacetime *Astrophys. J.* **820** 105

[18] Pihajoki P, Mannerkoski M, Nättilä J and Johansson P H 2018 General purpose ray tracing and polarized radiative transfer in general relativity *Astrophys. J.* **863** 8
[19] Kuchelmeister D, Müller T, Ament M, Wunner G and Weiskopf D 2012 GPU-based four-dimensional general-relativistic ray tracing *Comput. Phys. Commun.* **183** 2282–90
[20] Müller T and Weiskopf D 2010 Distortion of the stellar sky by a Schwarzschild black hole *Am. J. Phys.* **78** 204–14
[21] Kobras D, Weiskopf D and Ruder H 2002 General relativistic image-based rendering *Visual Comput.* **18** 250–8
[22] Müller T 2015 Image-based general-relativistic visualization *Eur. J. Phys.* **36** 065019
[23] Müller T and Frauendiener J 2012 Interactive visualization of a thin disc around a Schwarzschild black hole *Eur. J. Phys.* **33** 955
[24] Verbraeck A and Eisemann E 2021 Interactive black-hole visualization *IEEE Trans. Visual. Comput. Graph.* **27** 796–805
[25] Angel E and Shreiner D 2014 *Interactive Computer Graphics* (*A Top-Down Approach with WebGL*) 7th edn (London: Pearson Education Limited)
[26] Marschner S and Shirley P 2016 *Fundamentals of Computer Graphics* 4th edn (Florida: Apple Academic Press Inc.)
[27] Sherin Z W, Cheu R, Tan P and Kortemeyer G 2016 Visualizing relativity: the OpenRelativity project *Am. J. Phys.* **84** 369–74
[28] Rindler W 2001 *Relativity* (*Special, General and Cosmology*) (Oxford: Oxford University Press)
[29] Winterhalter F 2020 General-relativistic polygon rendering *BSc Thesis* Visualization Research Center, University of Stuttgart