Computer Physics Communications 185 (2014) 2301–2308

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

GeoViS—Relativistic ray tracing in four-dimensional spacetimes☆

Thomas Müller

Visualisierungsinstitut der Universität Stuttgart, Allmandring 19, 70569 Stuttgart, Germany

ARTICLE INFO

Article history:
Received 7 January 2014
Received in revised form
7 April 2014
Accepted 17 April 2014
Available online 1 May 2014

Keywords:
Relativity and gravitation
Relativistic visualization

ABSTRACT

The optical appearance of objects moving close to the speed of light or orbiting a black hole is of interest for educational purposes as well as for scientific modeling in special and general relativity. The standard approach to visualize such settings is ray tracing in four-dimensional spacetimes where the direction of the physical propagation of light is reversed. *GeoViS* implements this ray tracing principle making use of the *Motion4D* library that handles the spacetime metrics, the integration of geodesics, and the description of objects defined with respect to local reference frames. In combination with the GeodesicViewer, *GeoViS* might be a valuable tool for graduate students to get a deeper understanding in the visual effects of special and general relativity.

Program summary

Program title: GeoViS

Catalogue identifier: AESY_v1_0

Program summary URL: http://cpc.cs.qub.ac.uk/summaries/AESY_v1_0.html

Program obtainable from: CPC Program Library, Queen's University, Belfast, N. Ireland

Licensing provisions: Standard CPC licence, http://cpc.cs.qub.ac.uk/licence/licence.html

No. of lines in distributed program, including test data, etc.: 241796

No. of bytes in distributed program, including test data, etc.: 1667246

Distribution format: tar.gz

Programming language: C++, Scheme.

Computer: Linux platforms.

Operating system: Linux.

Has the code been vectorized or parallelized?: Yes, Parallelized using MPI.

RAM: ?? Bytes

Classification: 1.5.

External routines: Gnu Scientific Library (GSL), Motion4D, MPI.

Nature of problem:
First-person visualization of four-dimensional spacetimes in the theory of relativity.

Solution method:
Integration of ordinary differential equations for ray tracing.

Running time:
Scene dependent—from seconds to hours.

1. Introduction

Relativistic ray tracing is the most straightforward approach to visualize what an observer could see, for example, if he/she moves with nearly the speed of light or he/she watches a star orbiting a black hole. The idea behind this first-person view is to

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

E-mail address: Thomas.Mueller@vis.uni-stuttgart.de.

http://dx.doi.org/10.1016/j.cpc.2014.04.013
0010-4655/

start a light ray at the observer's position and to trace it back into the scene until it hits an object, leaves the region of interest, or violates the area of validity of the spacetime coordinates. In the geometric optics approach, light rays follow null geodesics. Hence, the geodesic equation has to be solved for the corresponding spacetime metric which is usually done by numerical integration. Besides the mere geometric distortion of the view due to curved light rays, apparent length contraction, and the finite speed of light, the light transport also has to be taken into account resulting in frequency shifts and lensing effects.

First detailed studies of the optical appearance of special-relativistically moving objects were done, amongst others, by [1–4]. Hsiung and Dunn [5] presented the first ray traced images of a set of moving bars. The optical appearance of a star orbiting a Kerr black hole was discussed by Cunningham and Bardeen [6,7]. General relativistic ray tracing images were announced by Ertl et al. [8], Nollert et al. [9], Nemiroff [10], and Weiskopf [11].

Freely available general relativistic ray tracing codes were published only recently. The *GYOTO* code by Vincent et al. [12] concentrates on astrophysical applications and numerically given spacetimes, while the code by Kuchelmeister et al. [13] demonstrates how to accelerate general relativistic ray tracing by exploiting the performance of graphics processing units (GPUs). If there exists an analytic solution to the geodesic equation for a particular spacetime, the ray tracing principle can be bypassed and the emitter–observer problem can be solved instead as shown by Dexter and Agol [14] in the case of the Kerr spacetime. Grave [15] explored visual effects within the Gödel spacetime by means of interactive relativistic visualization based on the analytic solution to the geodesic equation. Müller and Frauendiener [16] presented an interactive thin disk around a Schwarzschild black hole. For high-quality relativistic visualization of point-like objects an analytic solution is inevitable as shown by Müller and Weiskopf [17].

Special-relativistic ray tracing can also be made interactive by restricting the linear ray tracing method to local domains as shown by Müller et al. [18].

The ray tracing code *GeoViS* is dedicated to graduate students as a complementary tool to the much theory-loaded introductory courses to special or general relativity. It is not as sophisticated as previous codes but has the advantage of a simple scene description language and is being based on the *Motion4D* library [19] which already implements a large number of spacetime metrics, and hence, can model several different relativistic scenarios. Additionally stored data like light travel time, frequency shifts, or lensing effects can be visually explored in a post-processing step by means of the *gvsViewer* tool. In combination with the *GeodesicViewer* by Müller and Grave [20] the relativistic effects can be studied in all details.

The structure of the paper is as follows. In Section 2 the basic structure of *GeoViS* is presented. A brief introduction to the scene description language (SDL) is given in Section 3. Further examples are discussed in Section 4. The additional tool *gvsViewer* for post-processing the ray tracing data is introduced in Section 5.

GeoViS is implemented in C++ and is freely available for Linux. The source code with several examples as well as high-resolution images of this article are available from
http://go.visus.uni-stuttgart.de/geovis.

2. Basic structure of *GeoViS*

The basic structure of *GeoViS* is shown in Fig. 1. The central image generation unit is the `GvsSampleMgr` which organizes the sampling for each individual image pixel and which instructs writing the image and the additional image data to a file.

The `GvsProjector` represents the observer within its local reference frame. The projector requests the local initial light ray direction from the chosen camera model and transforms it to

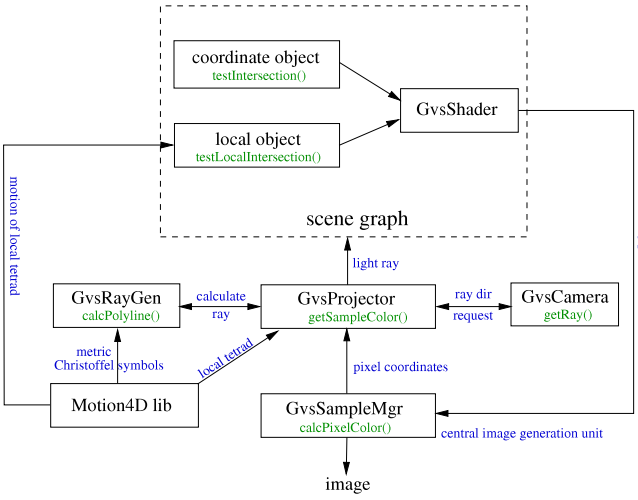

Fig. 1. Basic structure of *GeoViS*.

coordinate representation depending on its local tetrad. Then, the projector instructs the ray generator `GvsRayGen` to integrate the light ray depending on the position of the local tetrad, the initial light ray direction, and the given spacetime. This ray generation step is delegated to the *Motion4D* library which provides the metric coefficients and the Christoffel symbols for the geodesic equation and accomplishes the ray integration. Thus, any metric defined in the *Motion4D* library can, in principle, be used in *GeoViS*.

The computed light ray is then handed over to the scene graph where it is tested for intersections with all scene objects. If an intersection is found, a shading algorithm determines the color that is returned to the sample manager. Additionally, the position of the intersection point, the local light direction, the gravitational frequency shift, and the Jacobi parameters can be stored for later processing.

Because every image pixel is independent of all other pixels, ray tracing is trivially parallelizable via the Message Passing Interface (MPI) [21]. So far the most simple strategy is implemented where the image is split into several horizontal stripes which can be rendered by different threads.

2.1. Spacetime metrics and local tetrads

The *Motion4D* library is a collection of spacetimes whose metric tensors are given in closed form, $\mathbf{g} = g_{\mu\nu} \mathrm{d}x^\mu \otimes \mathrm{d}x^\nu$, with metric coefficients $g_{\mu\nu}$ being functions of the coordinates x^μ, $\mu, \nu \in \{0, 1, 2, 3\}$. Every metric is derived by the base class `m4dMetric` and has to overwrite, at least, the methods to calculate the metric coefficients and the Christoffel symbols, the method to check the validity of the coordinates, and the transformation between the intrinsic coordinates and a "natural" local tetrad.

In case of the Schwarzschild metric in spherical coordinates $x^\mu = (t, r, \vartheta, \varphi)$, the metric coefficients follow from the line element

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

where $r_s = 2GM/c^2$ is the Schwarzschild radius, G is Newton's constant, c is the speed of light, and M is the mass of the black hole. The validity of the coordinates is limited by the Schwarzschild radius. Thus, $r > r_s$, otherwise the integration of a light ray is aborted. The non-vanishing Christoffel symbols can be found, for example, in the *Catalogue of Spacetimes* [22]. Here, a "natural" local tetrad can be defined considering the spherical symmetry of the

Schwarzschild spacetime,

$$\hat{\mathbf{e}}_t = \frac{1}{c\sqrt{1 - r_s/r}}\partial_t, \qquad \hat{\mathbf{e}}_r = \sqrt{1 - \frac{r_s}{r}}\partial_r, \tag{2a}$$

$$\hat{\mathbf{e}}_\vartheta = \frac{1}{r}\partial_\vartheta, \qquad \hat{\mathbf{e}}_\varphi = \frac{1}{r\sin\vartheta}\partial_\varphi. \tag{2b}$$

In *GeoViS*, the local reference frame of the observer, who is represented by the projector, as well as the initial orientation of an object frame can be defined with respect to such a "natural" local tetrad, see also Section 3.

2.2. Camera models

GeoViS has four camera models implemented that differ in how a viewing direction is mapped onto the plane. The most natural one, `GvsPinHoleCam`, simulates a pinhole camera, where the relation between the viewing direction $\vec{k}$ and the pixel $P = (i, j)$ is given by $\vec{k} = (k_d, k_r(i, j), k_u(i, j))^T$ with $k_d = 1$,

$$k_r(i, j) = \rho\left(2\frac{i}{\mathrm{res}_h} - 1\right)\tan\frac{\mathrm{fov}_v}{2}, \tag{3a}$$

$$k_u(i, j) = \left(1 - 2\frac{j}{\mathrm{res}_v}\right)\tan\frac{\mathrm{fov}_v}{2}, \tag{3b}$$

and $\rho = \mathrm{res}_h/\mathrm{res}_v$. The pixel resolution of the image plane is defined in the horizontal and vertical directions by res_h and res_v. The vertical field of view is given by fov_v.

The `Gvs4PICam`, on the other hand, maps the full sky onto the image plane. For that, the pixel coordinates (i, j) are related to spherical coordinates (ϑ, φ) by means of $\vartheta = (j/\mathrm{res}_v)\cdot\pi$ and $\varphi = (i/\mathrm{res}_h - 1/2)\cdot 2\pi$. Then,

$$\vec{k} = (\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta)^T. \tag{4}$$

A compromise between the pinhole camera and the full sky camera is the `GvsPanoramaCam`, where the relation between pixel coordinates (i, j) and spherical coordinates (ϕ, λ) is given by $\phi = \arcsin[(1/2 - j/\mathrm{res}_v)\cdot\mathrm{fov}_v]$ and $\lambda = (i/\mathrm{res}_h - 1/2)\cdot\mathrm{fov}_h$,

$$\vec{k} = (\cos\phi\cos\lambda, \cos\phi\sin\lambda, \sin\phi)^T. \tag{5}$$

The `Gvs2PICam` can be used to produce "DomeMaster" images suitable for planetarium projection. Here, the pixel coordinates (i, j) are first mapped to the direction $\vec{s} = (s_x, s_y, s_z)^T$ with $s_x = 2(i/\mathrm{res}_h - 1/2)$, $s_y = 2(1/2 - j/\mathrm{res}_v)$, and $s_z = (1 - s_x^2 + s_y^2)^{1/2}$. Then, $\vec{s}$ is normalized and rotated by the pitch and heading angles p and h to obtain

$$\vec{k} = R_Z(h)R_Y(-p)\frac{\vec{s}}{|\vec{s}|}, \tag{6}$$

see Appendix for the definition of the rotation matrices R_Z and R_Y.

Given the local ray direction $\vec{k}$, the corresponding initial four-direction reads $\mathbf{k} = (-1, k_d, k_r, k_u)_{\mathcal{C}}$ with $\mathcal{C}$ being the local tetrad of the projector:

$$\mathbf{k} = -\mathbf{e}_{(0)} + k_d\mathbf{e}_{(1)} + k_r\mathbf{e}_{(2)} + k_u\mathbf{e}_{(3)}. \tag{7}$$

The camera is also responsible to select the filtering method. When only an `RGB`-filter is chosen, *GeoViS* renders the scene using shading methods as described in Section 2.4. Besides the RGB image, also the position of the intersection point, the four-vector of the light direction, and the corresponding frequency shift can be logged without any overhead using the `RGBpdz` filter. To obtain also the Jacobi parameters [23], the `RGBjac` filter has to be chosen. But note that, because of the additional equations that have to be integrated, the computation time strongly increases.

2.3. Scene objects

A scene object can be either a coordinate object or a local object. A coordinate object can only be used if the metric delivers a transformation from its intrinsic coordinates to pseudo-Cartesian coordinates. Here, the prefix "pseudo" means that the resulting coordinate space does not necessarily represent a Euclidean space. In case of the Schwarzschild metric, see Eq. (1), the intrinsic spherical coordinates (r, ϑ, φ) are transformed to pseudo-Cartesian coordinates as usual: $x = r\sin\vartheta\cos\varphi$, $y = r\sin\vartheta\sin\varphi$, and $z = r\cos\vartheta$. Then, the intersection of the light ray and the coordinate object is determined with these pseudo-Cartesian coordinates.

A local object, on the other hand, is described with respect to a local tetrad, see Section 2.1. Within this local reference frame (LRF), the object is handled like being in standard three-dimensional Euclidean space. The tetrad itself, however, can be either static or can move freely following a timelike geodesic. Starting from some initial position with a velocity defined with respect to the natural local tetrad at this position, the base vectors of the local reference frame will be parallel-transported along the timelike geodesic. The initial orientation of the LRF is also defined with respect to the natural local tetrad.

A fundamental problem with the local object is, that, in principle, the local tetrad is valid only in a first-order neighborhood of a point. As long as the tetrad is far from strongly curved spacetime, this neighborhood can be quite large. Close to a black hole, however, the local tetrad is a valid approximation only for a very small region. Furthermore, we describe a local object with respect to the local tetrad as being a "solid" object. There is no influence of the curved spacetime onto the object itself. Thus, the apparent visual distortion of a local object is only a hint on how such an object would look like in a real situation.

The intersection calculation of a light ray with a local object is split into two parts. First, we have to find the intersection between the light ray and the world-tube of the local object which is build from the spacelike sphere, that encompasses the local object, and that is extruded along the timelike geodesic of the local tetrad. Then, the ray segments that intersect the world-tube have to be transformed into the local tetrad system. There, standard three-dimensional ray tracing regarding the light travel time can be used.

2.4. Light sources and shading

In general curved spacetimes realistic shading of object surfaces are only possible if there is an analytic solution to the geodesic equation. Otherwise, it is nearly impossible to find a light ray that connects the point of interest with a light source. As there is no analytic solution currently implemented in the *Motion4D* library, only ambient light and the diffuse reflection with a light source at the observer's position is available in *GeoViS*. The diffuse shading is possible because the light ray of the light source follows the same null geodesic that is used to determine the pixel color.

In the special case of a Minkowski spacetime and an object at rest, a shadow ray can be emitted in the direction of the individual point-light sources. As a shadow light ray is integrated as any other light ray, the intersection can be done with any object.

It is also possible to map a texture onto a scene object to show, for example, its orientation. Here, a texture can be either an image or a procedural texture like a checkerboard pattern.

3. Scene description language (SDL)

The scene description language of *GeoViS* is based on the scheme implementation *TinyScheme* by Souflis et al. [24]. Hence, basic methods like numerical calculations or flow-control structures are already available. *GeoViS* specific commands are implemented within `GvsParser`, `GvsParseScheme`, and `parse_...` classes.

In the following, we give a simple example script to render a small checkered sphere orbiting a Schwarzschild black hole on the last stable circular orbit. The complete script can be found in the `examples`-folder of *GeoViS*. All of the object commands have in common that they can be referenced by their "`id`". Please note the different quotation marks which have different semantics in scheme. The standard quotation mark `'` has to be replaced by a left quotation mark `` ` `` if the following expression has to be evaluated first.

A metric is selected by its name, and its parameters are defined as key-value pairs, where the parameter keys equal the parameter names defined in the corresponding metric class of the *Motion4D* library.

```
(init-metric '(type "Schwarzschild")
             '(mass 1.0)
             '(id "metric")
)
```

For integrating the light rays, the Runge–Kutta Cash–Karp integrator of the *Gnu Scientific Library* (GSL) [25] with step size control and an absolute error constraint of $\epsilon_{\text{abs}} = 10^{-8}$ is used.

```
(init-solver '(type "GSL_RK_Cash-Karp")
             '(geodType "lightlike")
             '(eps_abs 1e-8)
             '(step_ctrl #t)
             '(id "raytracing")
)
```

Depending on the type of the camera, parameters like the viewing direction (dir), the vertical up-vector (vup), the field of view (fov), the image resolution (res), and the camera filter have to be set.

```
(init-camera '(type "PinHoleCam")
             '(dir #( 1.0 0.0 0.0 ))
             '(vup #( 0.0 0.0 1.0 ))
             '(fov #( 40.0 40.0 ))
             '(res #( 800 800 ))
             '(filter "FilterRGB")
             '(id "cam1")
)
```

Note that the camera itself is defined with respect to the local reference frame of the projector. The projector is also responsible to set the background color of the scenario using rgb-values in the interval [0, 1].

```
(init-projector '(localTetrad "locTedObs")
                '(color #(0.0 0.0 0.0))
                '(id "proj")
)
```

Before initializing the projector, the local reference frame with ID `"locTedObs"` has to be defined. Since the intrinsic coordinates of the Schwarzschild metric are spherical, the position of the local tetrad is also given in spherical coordinates $(t, r, \vartheta, \varphi)$. The base vectors `e0,e1,e2,e3` of the reference frame are given with respect to the natural local tetrad at the current position; hence, `incoords` is set to false. Here, we have $\mathbf{e}_{(0)} = \hat{\mathbf{e}}_{(t)}$, $\mathbf{e}_{(1)} = -\hat{\mathbf{e}}_{(r)}$, $\mathbf{e}_{(2)} = -\hat{\mathbf{e}}_{(\varphi)}$, and $\mathbf{e}_{(3)} = -\hat{\mathbf{e}}_{(\vartheta)}$.

```
(local-tetrad '(pos #(0.0 30.0 1.57079 0.0))
              '(e0 #(1.0 0.0 0.0 0.0))
              '(e1 #(0.0 -1.0 0.0 0.0))
              '(e2 #(0.0 0.0 0.0 -1.0))
              '(e3 #(0.0 0.0 -1.0 0.0))
              '(incoords #f)
              '(id "locTedObs")
)
```

The color of the ambient light is defined by the light manager. Here, we use bright white.

```
(init-light-mgr '(ambient #(1.0 1.0 1.0)) )
```

The object that orbits the black hole is represented by a sphere colored by a checkerboard texture, where the two colors of the checkerboard are defined as uniform textures.

```
(init-texture '(type "UniTex")
              '(color #(0.8 0.16 0.16))
              '(id "utex1")
)
```

The surface shader defines the checkerboard texture from `utex1` and `utex2`, where the numbers of checkerboard tiles in horizontal and vertical direction are defined by the `transform` element. The key-value pairs `ambient` and `diffuse` define the reflectance factor for the ambient and the diffuse shading. The left quotation mark `` ` `` in the `transform` line is necessary because the `scale-obj` key-value pair has to be evaluated before it can be set as transformation. The same reasoning is true for `objcolor`, where `init-texture` has to be evaluated first.

```
(init-shader '(type "SurfShader")
   `(objcolor ,(init-texture '(type "CheckerT2D")
              '(texture "utex1")
              '(texture "utex2")
              `(transform ,(scale-obj #(20.0 10.0)))
               )
   )
   '(ambient 0.4)
   '(diffuse 1.0)
   '(id "sphereShader")
)
```

Because the small checkered sphere is a local object (`gpObj TypeLocal`), the center can be located at the origin of the local reference frame.

```
(solid-ellipsoid `(objtype ,gpObjTypeLocal)
                 '(center #(0.0 0.0 0.0))
                 '(axlen #(0.5 0.5 0.5))
                 '(shader "sphereShader")
                 '(id "sphere")
)
```

The motion of the sphere's local reference frame follows a timelike geodesic that will be integrated numerically with a Runge–Kutta Cash–Karp integrator from the GSL. The step control is turned off to obtain a trajectory that is uniformly sampled with respect to the local reference frame's proper time.

```
(init-solver '(type "GSL_RK_Cash-Karp")
             '(geodType "timelike")
             '(eps_abs 0.01)
             '(step_ctrl #f)
             '(step_size 0.1)
             '(id "gsolver")
)
```

The initial position for the timelike geodesic is given with respect to the metric's intrinsic coordinates. The initial velocity and the orientation of the local reference frame are with respect to the natural local tetrad at this initial position. Thus, $\mathbf{e}_{(0)} = \hat{\mathbf{e}}_{(t)}$, $\mathbf{e}_{(1)} = -\hat{\mathbf{e}}_{(\vartheta)}$, $\mathbf{e}_{(2)} = -\hat{\mathbf{e}}_{(\varphi)}$, and $\mathbf{e}_{(3)} = \hat{\mathbf{e}}_{(r)}$. The initial velocity is $\beta = 0.5$ in the direction of $\hat{\mathbf{e}}_{\varphi}$. As the initial position is given for coordinate time $t = 0$, the geodesic has to be integrated forward and backward in time. Note that you have to take care for yourself that a light ray starting from the observer at coordinate time t_{obs} and traveling back in time can find an intersection with the moving object's local reference frame, see also Section 2.3.

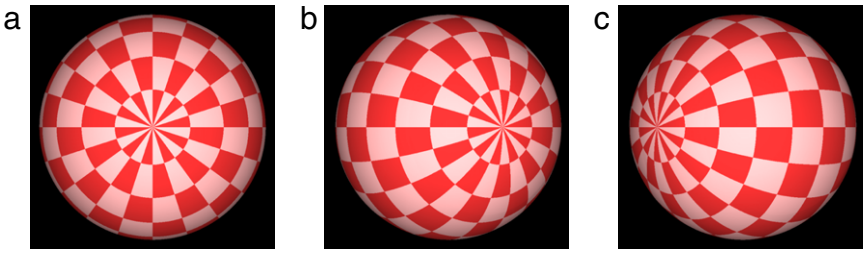

Fig. 2. Small checkered sphere orbiting a Schwarzschild black hole on the last stable circular orbit. (a) At rest, (b) at $t_{\text{obs}} = t_0$, and (c) $t_{\text{obs}} = t_0 + N \cdot t_{\text{step}}$.

```
(init-motion '(type "Geodesic")
       '(solver "gsolver")
       '(pos #( 0.0 6.0 1.5707963 0.0 ))
       '(localvel #( 0.0 0.0 0.5))
       '(e0 #(1.0 0.0 0.0 0.0))
       '(e1 #(0.0 0.0 -1.0 0.0))
       '(e2 #(0.0 0.0 0.0 -1.0))
       '(e3 #(0.0 1.0 0.0 0.0))
       '(maxnumpoints 1000)
       '(forward 200.0)
       '(backward 300.0)
       '(id "motion")
)
```

The `"sphere"` object and the local reference frame represented by `"motion"` can now be combined into a local object.

```
(local-comp-object '(obj "sphere")
                   '(motion "motion")
                   '(id "lco1")
)
```

For each image of an image sequence a `device` object has to be defined that links all objects together. Here, the scene graph consists only of the single object `"lco1"`. Other objects have to be set only if there are more than one of them. For example, if there are two cameras, one of them can be selected by means of a key-ID pair. The `setparam` key can be used to change some parameters of previously defined objects. Here, the observer's coordinate time is modified to simulate an elapsing observation time.

```
(define t_start 27.8918 )
(define t_step 0.307812 )
(define t_count 300 )
(do ((count 0 (+ count 1))) ((= count t_count))
   (init-device '(type "standard")
      '(obj "lco1")
      '(camera "cam1")
      `(setparam ("locTedObs" "time"
            ,(+ t_start (* t_step count))))
   )
)
```

More details can be found also in the corresponding parser methods `parser. . .` within the `Parser` sub-directory.

4. Examples

4.1. Small checkered sphere orbiting a black hole

The first example shows the result of the scene described in the previous section, where a small checkered sphere orbits a Schwarzschild black hole on the last stable circular orbit. A detailed discussion of the visual effects of this scene can be found also in Müller [26].

The small checkered sphere is given with respect to a local reference frame whose motion follows a timelike geodesic with initial values ($t_{\text{obj}} = 0$, $r_{\text{obj}} = 3r_s$, $\vartheta_{\text{obj}} = \pi/2$, $\varphi_{\text{obj}} = 0$), and initial frame vectors $\mathbf{e}_{(0)} = \hat{\mathbf{e}}_{(t)}$, $\mathbf{e}_{(1)} = -\hat{\mathbf{e}}_{(\vartheta)}$, $\mathbf{e}_{(2)} = -\hat{\mathbf{e}}_{(\varphi)}$, and $\mathbf{e}_{(3)} = \hat{\mathbf{e}}_{(r)}$. The observer is located at ($r_{\text{obs}} = 15r_s$, $\vartheta_{\text{obs}} = \pi/2$, $\varphi_{\text{obs}} = 0$) and the observation times $t_{\text{obs}}(n)$ are selected such that in the first image ($n = 0$), the sphere appears at its initial position right between the black hole and the observer,

$$t_{\text{obs}}(n) = t_0 + n \cdot t_{\text{step}}. \tag{8}$$

Due to the finite speed of light, t_0 is the light travel time from $r = 3r_s$ to $r = 15r_s$ given by

$$t_0 = \frac{1}{c}\int_{r=3r_s}^{15r_s} \frac{\mathrm{d}r}{1 - r_s/r} = \frac{1}{c}\left[r + r_s \log(r - r_s)\right]_{r=3r_s}^{15r_s}. \tag{9}$$

The time step t_{step} is defined here by the number of images N that should cover a complete orbit of the sphere, $t_{\text{step}} = t_{2\pi}/N$, where $t_{2\pi} = 2\pi/\Omega$ is the time for one orbit and $\Omega = 2\pi\sqrt{54}r_s/c$ is the angular velocity at the last stable circular orbit, see e.g. Müller [26]. With $c = 1$, $r_s = 2m$, $m = 1$, $N = 300$, we obtain $t_0 \approx 27.8918$, $t_{2\pi} \approx 92.3436$, and $t_{\text{step}} \approx 0.307812$.

Fig. 2 shows close-up views of the checkered sphere (a) as long as it is at rest at its initial position, (b) at observation time $t_{\text{obs}} = t_0$, and (c) after one full revolution around the black hole. The difference between (a) and (b) follows from the apparent rotation due to the finite speed of light which is a pure special relativistic effect. Because in (b) the sphere moves with 50% the speed of light orthogonal to the line of sight, it appears to be rotated by 30° with respect to its actual orientation that is shown in (a). During the orbital motion, the sphere undergoes a geodesic precession that explains the additional rotation of about 105.44° as shown in Fig. 2(c).

4.2. Simplified accretion disk model

As a first approximation, a thin disk around a rotating black hole can be represented by a ring defined as a coordinate object with $r_{\text{in}} \le r \le r_{\text{out}}$ and $0 \le \varphi < 2\pi$ in the equatorial plane of a Kerr black hole whose metric is given here in Boyer–Lindquist coordinates

$$\begin{aligned}\mathrm{d}s^2 = &-\left(1 - \frac{r_s r}{\Sigma}\right)c^2\mathrm{d}t^2 - \frac{2r_s a r \sin^2\vartheta}{\Sigma}c\,\mathrm{d}t\,\mathrm{d}\varphi + \frac{\Sigma}{\Delta}\mathrm{d}r^2 \\ &+ \Sigma\mathrm{d}\vartheta^2 + \left(r^2 + a^2 + \frac{r_s a^2 r \sin^2\vartheta}{\Sigma}\right)\sin^2\vartheta\,\mathrm{d}\varphi^2,\end{aligned} \tag{10}$$

where $\Sigma = r^2 + a^2\cos^2\vartheta$, $\Delta = r^2 - r_s r + a^2$, and $r_s = 2GM/c^2$, see e.g. Bardeen et al. [27]. M is the mass and a is the angular momentum per unit mass of the black hole (see Figs. 3 and 4).

4.3. Box around a Morris–Thorne wormhole

The most simple wormhole geometry is given by the Morris–Thorne [28] metric

$$\mathrm{d}s^2 = -c^2\mathrm{d}t^2 + \mathrm{d}l^2 + \left(b_0^2 + l^2\right)\left(\mathrm{d}\vartheta^2 + \sin^2\vartheta\,\mathrm{d}\varphi^2\right), \tag{11}$$

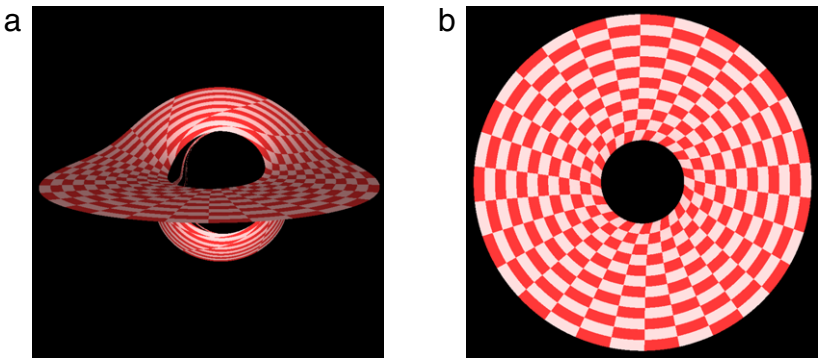

Fig. 3. Thin "accretion disk" in the equatorial plane of an extreme Kerr black hole with $M = 1$ and $a = 1$ as seen from an observer located at $r_{\text{obs}} = 50\,M$ and inclination angles (a) $\iota = 80^\circ$ and (b) $\iota = 2^\circ$ with respect to the disk normal. The inner and outer radii of the disk read $r_{\text{in}} = 3\,M$ and $r_{\text{out}} = 15\,M$.

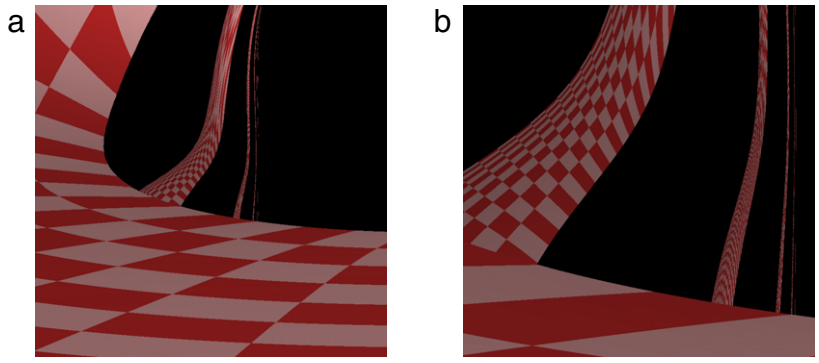

Fig. 4. Close-up view of higher-order images of the disk with (a) fov $= 5^\circ$, (b) fov $= 1.5^\circ$.

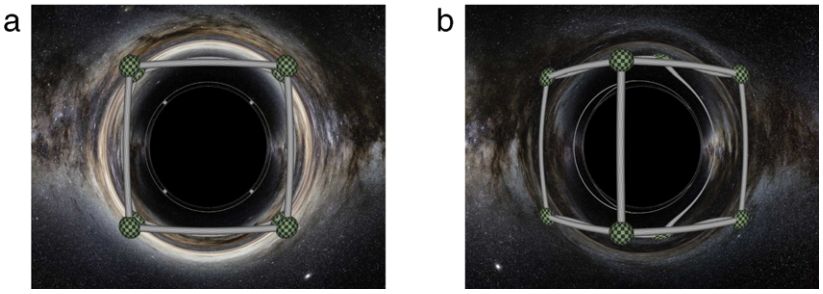

Fig. 5. Box around a Morris–Thorne wormhole with $b_0 = 2$ for observation azimuth angles (a) $\varphi = 0^\circ$ and (b) $\varphi = 35^\circ$. The Milky Way panorama is by ESO/S.Brunier.

where $b_0 > 0$ defines the size of the wormhole throat and $l \in \mathbb{R}$ is the proper radial coordinate. We will call the domain $l > 0$ the upper universe and $l < 0$ the lower universe. Fig. 5 shows a Morris–Thorne wormhole with a box built from cylindrical rods and spheres at the corners surrounding the wormhole throat in the upper universe. There is also a Milky Way panorama representing the asymptotic background of the upper universe. The lower universe is just black without any objects inside. More visualization details can be found in Müller [29].

4.4. Black hole shadows of a Kastor–Traschen spacetime

The Kastor–Traschen spacetime in Cartesian coordinates (t, x, y, z) is represented by the line element

$$\mathrm{d}s^2 = -\Omega^{-2}\mathrm{d}t^2 + a^2\Omega^2\left(\mathrm{d}x^2 + \mathrm{d}y^2 + \mathrm{d}z^2\right), \tag{12}$$

where the scaling factor $a(t) = e^{Ht}$, $\Omega = 1 + \sum_i m_i/(ar_i)$, $r_s = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2}$, and $H = \pm\sqrt{\Lambda/3}$ with cosmological constant Λ. The sum is over all black holes located at positions (x_i, y_i, z_i) having the masses m_i, $i = 1, \ldots, N$. Here, we have only two black holes of the same mass, $m_1 = m_2 = 1$, that are located on the z-axis at $z_1 = 4$ and $z_2 = -4$.

Fig. 6 shows the regions where no light can reach the observer, who is located at $x_{\text{obs}} = 40, y_{\text{obs}} = z_{\text{obs}} = 0$, in black. These regions are also called the shadows of the black holes.

4.5. Falling into a static black hole

Falling freely from rest into a static black hole can be simulated most easily by means of ingoing Eddington–Finkelstein coordinates $(v, r, \vartheta, \varphi)$, where the Schwarzschild coordinate time t is replaced by the ingoing null coordinate v. The corresponding metric in geometric units reads

$$\mathrm{d}s^2 = -\left(1 - \frac{r_s}{r}\right)\mathrm{d}v^2 + 2\mathrm{d}v\mathrm{d}r + r^2\left(\mathrm{d}\vartheta^2 + \sin^2\vartheta\,\mathrm{d}\varphi^2\right) \tag{13}$$

with $r_s = 2m$ being the Schwarzschild radius. The apparent size of the black hole shadow with apex angle ξ_{crit} can be determined

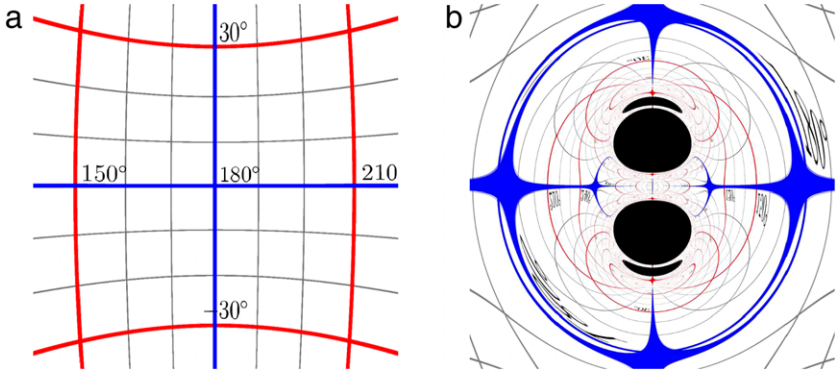

Fig. 6. Shadows of the two black holes in the Kastor–Traschen spacetime with $m_1 = m_2 = 1$ and positions $z_1 = 4, z_2 = -4$. (a) Reference image of the background grid; (b) First-person view at observation time $t = 60$.

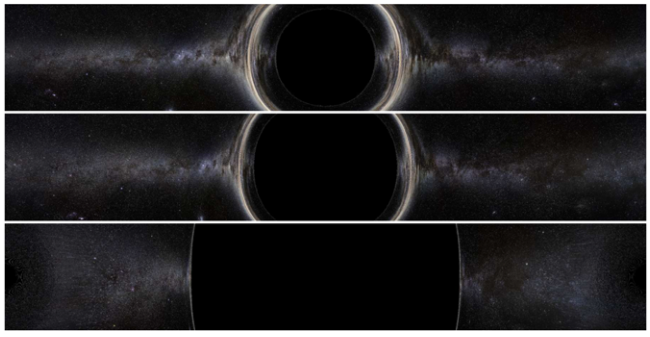

Fig. 7. First-person view of an observer falling freely into a static black hole. Here, a panorama camera with $360^\circ \times 60^\circ$ field of view is used. The proper observation times are $\tau = \{0, 32.395, 35.09\}$ and the current positions read $r_{\text{obs}} \approx \{10.0, 3.01, 0.173\}$ (from top to bottom).

analytically,

$$\cos\xi_{\text{crit}} = \frac{x_{\text{obs}}^2\sqrt{1-x_i}\sqrt{x_{\text{obs}}-x_i} \pm \sqrt{p\left(x_{\text{obs}}^3 - x_{\text{obs}}^2 + p\right)}}{x_{\text{obs}}^3 - x_i x_{\text{obs}}^2 + p} \tag{14}$$

with the observer's initial position r_i, its current position r_{obs}, $x_i = r_s/r_i$, $x_{\text{obs}} = r_s/r_{\text{obs}}$, and $p = 4/27$, see Müller [30].

Fig. 7 shows the view of an observer starting from rest at $r_i = 10$ at its proper times τ. In the first few moments, the black hole appears to shrink although the observer moves closer and closer to the black hole. But that is due to the special relativistic aberration caused by an increasing velocity. Until the observer crashes into the singularity, the black hole's shadow increases in size up to $2\xi_{\text{crit}} = 180^\circ$ at $r \to 0$.

4.6. Rolling wheel in Minkowski

Consider a rolling wheel of radius r whose center moves with a velocity v close to the speed of light c with respect to the ground, see Kraus et al. [31]. Please note that we do not care here about the problems of constructing such a wheel. As the wheel is rolling on the ground, the contact point has zero velocity whereas the topmost point has velocity v with respect to the center. Due to the relativistic velocity-addition, the velocity of the topmost point with respect to the ground is $2v/(1 + v^2/c^2)$.

Fig. 8 shows the wheel rolling from left to right at two observation times as seen by a static observer with a pinhole camera. The light source is located somewhat behind and above the checkered block. All the apparent distortion effects of the wheel and its spokes as well as the apparently non-matching shadows are mainly due to the finite speed of light.

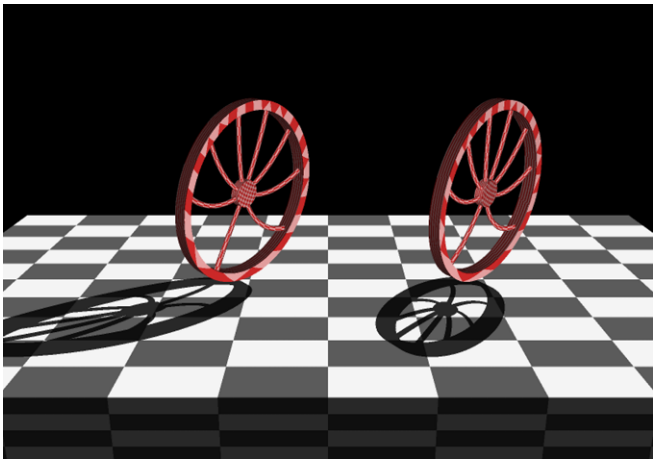

Fig. 8. A wheel of radius $r = 2$ roles with $v = 0.8c$ along the negative y-axis. At $t = 0$, the wheel crosses the origin. The observer is located at $(x = -15, y = 0, z = 4.0)$ and has a camera with $50^\circ \times 35^\circ$ field of view. The light source is positioned at $(x = 20, y = 0, z = 25)$. The observation times are $t = 13.8$ (left) and $t = 20.3$ (right).

5. Post-processing with `gvsViewer`

Direct general relativistic ray tracing is a very time expensive rendering method. Therefore, it is all the more important that not only the RGB image is stored but that also additional data like the coordinates of the intersection points or frequency shift factors are recorded for later post-processing and visual analysis. The additional data can be stored by means of different camera filters, see Section 2.2.

`gvsViewer` is a basic visualization tool based on the Open Graphics Library (OpenGL) [32] and the AntTweakbar [33] for a simple graphical user interface. It can show the RGB images as well as the additional data entries as grayscale or color-coded image. Additionally, the data values can be directly read by moving the mouse ontop of the corresponding image pixel.

The color-coding is realized in the following way. A window-filling rectangle representing the camera's viewing plane is drawn. This rectangle is split into individual pixels (also called fragments) via the rasterizer stage of the OpenGL graphics pipeline. Afterwards, each pixel/fragment can be handled by a fragment shader program which has access on the data values that are stored in texture memory. Then, for example, the intersection time can be visualized by gray values after being mapped into the interval $[0, 1]$.

A more advanced example shows a thin accretion disk around a Schwarzschild black hole similar to the one of Section 4.2. For that, we have to use the `RGBpdz` camera filter which additionally

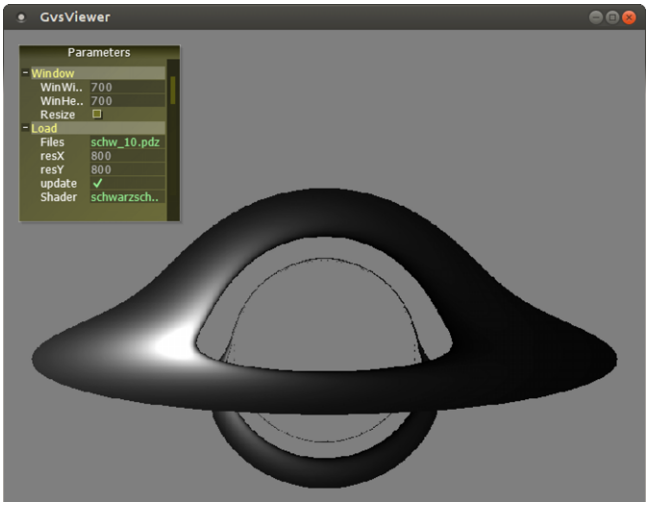

Fig. 9. Screenshot of `gvsViewer` showing a thin accretion disk around a Schwarzschild black hole post-processed with the `schwarzschildDisc` fragment shader. The inclination to the disk normal is $\iota = 80°$ and the disk covers the radial domain $3r_s \le r \le 7.5r_s$.

stores the intersection position and the intersecting light direction. Following Luminet [34], the disk can be described by idealized non-interacting particles moving on timelike circular geodesics and radiating isotropically. The local velocity β of a particle at radius r is given by $\beta = 1/\sqrt{2(r/r_s - 1)}$. The corresponding four-velocity $\mathbf{u}$ with respect to the local tetrad (2) then reads $\mathbf{u} = \gamma\left(\mathbf{e}_{(t)} + \beta\mathbf{e}_{(\varphi)}\right)$ with $\gamma = 1/\sqrt{1-\beta^2}$. Here, r can be determined from the position-texture. The frequency-shift z due to gravitation and due to the Doppler effect, follows from the scalar product between the four-velocity $\mathbf{u} = u^\mu\partial_\mu$ and the light ray direction $\mathbf{k} = k^\mu\partial_\mu$ at the intersection point

$$1 + z = g_{\mu\nu}u^\mu k^\nu. \tag{15}$$

The bolometric flux of radiation is taken from Page and Thorne [35],

$$F = \frac{F_0}{(R-3/2)R^{5/2}}\left[\sqrt{R} - \sqrt{3} + \frac{\sqrt{3/2}}{2}\ln\left(\frac{\sqrt{R}+\sqrt{3/2}}{\sqrt{R}-\sqrt{3/2}}\frac{\sqrt{3}-\sqrt{3/2}}{\sqrt{3}+\sqrt{3/2}}\right)\right] \tag{16}$$

with $F_0 = 3GM\dot{M}/(8\pi r_s^3)$, accretion rate $\dot{M}$, and $R = r/r_s$. The maximum $F_{\max}/F_0 \approx 9.167 \cdot 10^{-4}$ is located at $R_{\max} \approx 4.776$. The observed bolometric flux F_{obs} is related to the flux F_{src} emitted by the disk via the transformation $F_{\text{obs}} = F_{\text{src}}/(1+z)^4$ (see Luminet [34]).

The fragment shader program has to implement the above calculations. First, it has to test if a pixel is related to a disk intersection. For a valid pixel, it can immediately read the intersection position r and the intersecting light ray direction k^μ in spherical coordinates. To determine the frequency shift z, the metric coefficients $g_{\mu\nu}$ have to be evaluated at the intersection position. Then, F_{obs} can be mapped to a gray value, see Fig. 9. However, the high dynamic range of the bolometric flux cannot be mapped realistically.

6. Outlook

GeoViS allows to study the visual properties of a spacetime from a first-person point of view. To better understand the resulting images, a process communication between `gvsViewer` and the GeodesicViewer [20] could show to each pixel the corresponding light ray.

So far, objects in *GeoViS* are given either with respect to coordinates or with respect to a local tetrad. However, describing an object with respect to a local tetrad is valid only for very small objects. A large sphere could be defined more realistically by a radius that is given in proper length units or that follows from a predefined light travel time.

Correct illumination of objects is an unfeasible task in pure four-dimensional ray tracing, because it is almost impossible to find a light ray that connects the light source with the point of interest. For that, an analytic solution to the null geodesic equation is indispensable.

The parallelization strategy by splitting the image into stripes is by no means balanced. A much better load balancing could be achieved by tabulating the rendering times and splitting the image into small tiles accordingly.

Appendix. Rotation matrices

Rotation matrices used to define the light direction of the `Gvs2PICam` in Section 2.2:

$$R_Z(\alpha) = \begin{pmatrix}\cos\alpha & -\sin\alpha & 0\\ \sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{pmatrix},\qquad R_Y(\alpha) = \begin{pmatrix}\cos\alpha & 0 & -\sin\alpha\\ 0 & 1 & 0\\ \sin\alpha & 0 & \cos\alpha\end{pmatrix}. \tag{A.1}$$

References

[1] A. Lambda, Zeitschrift f. Phys. 27 (1924) 138–148 (in German).
[2] R. Penrose, Proc. Camb. Philos. Soc. 55 (1959) 137–139.
[3] J. Terrell, Phys. Rev. 116 (1959) 1041–1045.
[4] G.D. Scott, H.J. van Driel, Am. J. Phys. 38 (1970) 971–977.
[5] P.-K. Hsiung, R.H.P. Dunn, Proc. 1989 ACM/IEEE conf. Supercomp., 1989, pp. 597–606.
[6] C.T. Cunningham, J.M. Bardeen, Astrophys. J. 173 (1972) L137–L142.
[7] C.T. Cunningham, J.M. Bardeen, Astrophys. J. 183 (1973) 237–264.
[8] T. Ertl, et al., Proc. Eurographics'89, 1989, pp. 149–158.
[9] H.-P. Nollert, H. Ruder, H. Herold, U. Kraus, Astronom. Astrophys. 208 (1989) 153–156.
[10] R.J. Nemiroff, Am. J. Phys. 61 (1993) 619–632.
[11] D. Weiskopf, Visualization of four-dimensional spacetimes, Ph.D. Thesis, Eberhard-Karls-Universität Tübingen, 2001.
[12] F.H. Vincent, T. Paumard, E. Gourgoulhon, G. Perrin, Classical Quantum Gravity 28 (2011) 225011.
[13] D. Kuchelmeister, T. Müller, M. Ament, G. Wunner, D. Weiskopf, Comput. Phys. Commun. 183 (2012) 2282–2290.
[14] J. Dexter, E. Agol, Astrophys. J. 696 (2009) 1616–1629.
[15] F. Grave, The Gödel universe: physical aspects and egocentric visualizations, Ph.D. Thesis, University of Stuttgart, 2010.
[16] T. Müller, J. Frauendiener, Eur. J. Phys. 33 (2012) 955–963.
[17] T. Müller, D. Weiskopf, Am. J. Phys. 78 (2010) 204–214.
[18] T. Müller, S. Grottel, D. Weiskopf, IEEE Trans. Vis. Comput. Graphics 16 (2010) 1243–1250.
[19] T. Müller, F. Grave, Comput. Phys. Commun. 180 (2010) 2355–2360.
[20] T. Müller, F. Grave, Comput. Phys. Commun. 181 (2010) 413–419.
[21] OpenMPI is an open source MPI-2 implementation, http://www.open-mpi.org.
[22] T. Müller, F. Grave, arXiv:0904.4184 [gr-qc].
[23] T. Müller, Comput. Phys. Commun. 182 (2011) 1386–1388.
[24] D. Souflis, et al., http://tinyscheme.sourceforge.net.
[25] GNU Scientific Library (GSL), http://www.gnu.org/software/gsl.
[26] T. Müller, Gen. Relativity Gravitation 41 (2009) 541–558.
[27] J.M. Bardeen, W.H. Press, S.A. Teukolsky, Astrophys. J. 178 (1972) 347–369.
[28] M.S. Morris, K.S. Thorne, Am. J. Phys. 56 (1988) 395–412.
[29] T. Müller, Am. J. Phys. 72 (2004) 1045–1050.
[30] T. Müller, Gen. Relativity Gravitation 40 (2008) 2185–2199.
[31] U. Kraus, H. Ruder, D. Weiskopf, C. Zahn, Phys. J. 7/8 (2002) 77–82 (in German).
[32] Open graphics library (OpenGL), http://www.opengl.org.
[33] AntTweakBar, http://anttweakbar.sourceforge.net/.
[34] J.P. Luminet, Astronom. Astrophys. 75 (1979) 228–235.
[35] D.N. Page, K.S. Thorne, Astrophys. J. 191 (1974) 499–506.