Null Geodesics Around a Kerr Black Hole

Bo Milvang–Jensen

Astronomical Observatory
Niels Bohr Institute for Astronomy, Physics and Geophysics
University of Copenhagen

June, 1996

Abstract

This *document* is my exam project ("afløsningsopgave") for the course "Relativistic Computer Dynamics and Visualization[1]" by Åke Nordlund[2] and Axel Brandenburg[3]. The document is about three things: an aspect of the Theory of General Relativity, *viz.* null geodesics around a Kerr black hole, visualization, and Web authoring.

Please note, that you are right now reading the LaTeX ("paper") version of this document. As explained further down, this is only a secondary product, with the primary one being the HTML version (the *Web exhibition*), which can be found at `http://www.astro.ku.dk/~milvang/RelViz/`.

[1] http://www.astro.ku.dk/~aake/teaching/rel/
[2] http://www.astro.ku.dk/~aake/
[3] http://antares.ncl.ac.uk/~brandenb/

Contents

Chapter 1

Introduction

In this chapter, I will discuss what a Web exhibition is, what kind of contents is has, what the requirements are to view it, and what considerations I have made in choosing LaTeX2HTML as the document preparation system for the entire document.

1.1 What is a Web exhibition?

The web page for the *RelViz course*[1] says:

> The examination for this course will take the form of WWW exhibits (similar to the Exhibits[2] of the NCSA Relativity Group[3]; i.e., HTML structures with explanatory text, graphics, images, and MPEG movies, describing a visualization of relevance to the subject of the course.

The word *Web exhibition* immediately implies that the *document* in question will be available on *the Web*, but but not per se what the *content type* will be, and that it will be more than just plain text. A more descriptive phrase might be *Web multimedia report*. What content types my Web exhibition will encompass will be discussed in the next section.

1.2 Multimedia content types

A multimedia document, such as this Web exhibition, consists of a number of different content types, such as still images, movies, and 3D scenes. The definitions given below of these and related concept are my own, relevant to this context, and might not be universally accepted or applicable.

1.2.1 Still images

A *still image* (or just *image*) is a 2D array of pixels, which each has a color.

The color is usually specified in RGB (red, green, blue) colorspace as 3 numbers from 0 to 255, i.e. there are $2^{24} = 16.8$ million possible colors. Two important image classes exist:

24 bit images: Images which allow for any number of the 16.8 million (24 bit) colors to be present simultaneously in a given image

8 bit colormapped images: Images which allow only for a maximum of 256 (8 bit) colors to be present in a given image. These colors are drawn from the 24 bit palette, and the mapping between the up to 256 numbers and the 24 bit colors constitutes a *colormap*.

Some of the image file formats found on the Web are:

[1] http://www.astro.ku.dk/~aake/teaching/rel/
[2] http://jean-luc.ncsa.uiuc.edu/Exhibits/
[3] http://jean-luc.ncsa.uiuc.edu/

GIF (Graphics Interchange Format): GIF images are 8 bit colormapped, and are compressed using a loss free algorithm (LZW). The compression scheme is well suited for images with large area of the same color, such as the images found in this Web exhibition.

JPEG (Joint Photo Experts Group): JPEG images are 24 bit color, and are compressed using a lossy algorithm. The compression scheme is well suited for continuous tone natural images, such as scanned photographs. I have chosen not to use this image format at all in this project. Actually, JPEG is a encoding scheme, not a file format. The file format used on the Web is called JFIF (JPEG File Interchange Format). For more details on JPEG, see the JPEG FAQ[4].

SGI / IRIS / RGB: SGI is a native Silicon Graphics[5] image file format. Images can be of different types, including 24 bit color. They can be uncompressed or run-length encoded (RLE). Files have extension `.rgb`, hence one of the names. This format has been used a lot in this project as an intermediate image file format; this will be described in more detail below. On a Silicon Graphics computer, `man 4 rgb` provides more information about this image file format.

PNG (Portable Network Graphics): PNG is still in development, but is mentioned here due to it's promising specifications: it is lossless, well-compressed, and supports up to 48 bit color, thus being a likely replacer of GIF.

All of the above are described in further detail in the Graphics File Formats FAQ part 3[6] (see also part 1[7], 2[8], and 4[9].) Compression per se, including LZW and JPEG (but not RLE), is described in further detail in the Compression FAQ.

1.2.2 Movies

A *movie* is a set of images, which are displayed sequentially. This *video track* can have an accompanying *audio track*, but in this context I will not be dealing with the latter.

Some of the movie file formats found on the Web are:

MPEG (Moving Picture Expert Group): There are several types of MPEG: MPEG-1, MPEG-2, and MPEG-4. It seems to me, that the MPEG most often found on the Web is MPEG-1, and usually with only a video track. MPEG seems to be by far the most popular movie format on the Web, and therefore it is one of the formats I offer the movies of this project in. MPEG uses 3 different compression techniques at once, with one of them being similar to JPEG. The compression is lossy and efficient.
For more information about MPEG, see the MPEG FAQ[10], the Moving Picture Expert Group Information Page[11], and the the first two sections of the article by Patel, Smith, & Rowe (1993)[12].
For the record: MPEG is also known as ISO/IEC JTC1 SC29 WG11 ;-)

QuickTime: QuickTime seems to be an Apple product; they host a Web site[13] about it. However, viewers and encoders for other platforms are available. QuickTime can store among other

[4] http://www.cis.ohio-state.edu/hypertext/faq/usenet/jpeg-faq/top.html
[5] http://www.sgi.com/
[6] http://www.cis.ohio-state.edu/hypertext/faq/usenet/graphics/fileformats-faq/part3/faq.html
[7] http://www.cis.ohio-state.edu/hypertext/faq/usenet/graphics/fileformats-faq/part1/faq.html
[8] http://www.cis.ohio-state.edu/hypertext/faq/usenet/graphics/fileformats-faq/part2/faq.html
[9] http://www.cis.ohio-state.edu/hypertext/faq/usenet/graphics/fileformats-faq/part4/faq.html
[10] http://www.powerweb.de/mpeg/mpegfaq/
[11] http://www.vol.it/MPEG/
[12] ftp://mm-ftp.cs.berkeley.edu/pub/multimedia/papers/MM93.ps.Z
[13] http://quicktime.apple.com/

things movies, using different compression schemes. For this project, I have chosen "Animation", which is an RLE scheme. This compression I presume is loss free. For this reason, I also offer my movies in that format.
For more information about QuickTime, see the QuickTime FAQ[14].

Animated GIF: The GIF89 standard supports multiple images in the same file, and an animation of these. Netscape has a *built-in* viewer not only for GIFs but also for animated GIFs, so therefore animated GIF has a big advantage. Besides, Netscape will show the movie while it is being loaded. For the these reasons, I have also made my movies of this project as animated GIF. On the downside, it is not possible to single step the frames, play the movie again, etc.

1.2.3 3D Scenes

A *3D scene* is a collection of "material" objects, such as surfaces (constructed by a number of polygons), points and lines, living in 3D space. The two 3D scene formats I have used in this project are:

Inventor: Inventor is two things:

- An object-oriented toolkit for developing interactive, 3D graphics.
- A file format for exchanging 3D data among applications.

I will only be using Inventor in the latter sense, i.e. as a file format, or *programming language*.
A technical note: Inventor V1.0 is called *IRIS Inventor*, whereas Inventor V2.0 is called *Open Inventor*. I will only be using Inventor V1.0, and I will refer to that as Inventor.
What I call a *3D scene* is in Inventor terminology called a *scene*.
Information about Inventor can be found at the Open Inventor page[15].

VRML (Virtual Reality Modeling Language): VRML is *the* standard for 3D scenes on the Web. It is based on Inventor. It has a subset of the Inventor commands, and some additional commands which are not found in Inventor; this will be exemplified later.
What I call a *3D scene* is in VRML terminology called a *world*.
When I refer to VRML, I mean VRML V1.0; the specifications are available here[16].
For more information about VRML, see the VRML Forum[17].

Inventor and VRML are explained further in section 4.1.

1.3 Browser requirements

If I were to make a report on paper, I could send the actual paper print-out to the person I wanted to read it. I would not have to worry much about what it would look like for him/her, since it was the very paper I myself had had in my hand, that he/she would have gotten.

If I were to make a report on a video tape, it would still be fairly simple, but I would have to make sure, that that person were able to view the video tape format I had used, e.g. Super-VHS PAL. And of course, it would be absolutely necessary that the person had a VCR in the first place.

Now that I am making a report as a multimedia document on the Web, what then? As a first prerequisite, the person should have a Web browser (such as Netscape[18]), running on a machine with access to the Net. In more details, the browser requirements for this multimedia report are:

[14] http://www.QuickTimeFAQ.org/
[15] http://www.sgi.com/Technology/Inventor.html
[16] http://vrml.wired.com/vrml.tech/vrml10-3.html
[17] http://vrml.wired.com/
[18] http://www.netscape.com/

Ability to show the different occurring content types: As discussed in section 1.2, there are several different "classes" of content types, e.g. still images, movies, and 3D scenes, and within these classes there are several different "formats". These latter each have a distinct so-called *MIME type*. When a Web server sends some data (e.g. the contents of a GIF file), it also sends the MIME type, so that the Web browser (e.g. Netscape) knows what kind of data it is getting. The Web browser then has a table (the *mailcap*), that specifies what to do with a number of different MIME types; it could either handle it itself, or it could send it to an external viewer.

Here is an example of all the different MIME types that I use in this Web exhibition. Try to click on it and see what happens. If your browser just lets you save the file without displaying it, help might be found further down. Note, that I will explain in section 5.2 what these files show.

`text/plain`: Text, plain

`text/html`: Text, HTML (that is what you are reading right now ;-)

`image/gif`: Still image, GIF

`video/mpeg`: Movie, MPEG (here: MPEG-1 video bitstream)

`video/quicktime`: Movie, QuickTime (here: *Animation* coded)

`image/gif`: Movie, Animated GIF

`application/x-inventor`: 3D Scene, Inventor

`x-world/x-vrml`: 3D Scene, VRML

`application/postscript`: Page description language, PostScript

Ability to show the HTML markup used: I have used HTML version 2.2, that have *tags* for making tables. If you see a nicely formatted 3 by 3 table right below this, your browser understands these tags:

type	*style*	
smart	red	short
rather silly	puce	tall

(courtesy Lamport 1986 ;-)

Ability to make interactive forms: In this project I have also used interactive forms. For these, the browser needs to render the associated HTML tags correctly, accept input from the user, and send it in the right way to the Web server. An example of a form follows here. It will return an HTML document telling you what value you submitted.

> In the HTML version there is an interactive form at this place. *Please go see it there!*.

Ability to decompress compressed files: I have often compressed Inventor and VRML files using `gzip`, since their size is reduced a factor of ~ 5 that way. Here[19] is a gzip'ed Inventor file.

If all of the above worked for you, you do not have to read through the rest of this section!

So, what kind of setup fulfills all of the above requirements? To start with the three last ones (tables, forms, and gzip): just get Netscape version 2 or later. I have been running version 2.02 myself.

To be able to view all the above mentioned MIME types, you have to:

[19] http://www.astro.ku.dk/~milvang/RelViz/MIME/test.iv.gz

- Have an external viewer for those MIME types which your Web browser cannot handle itself (Netscape has built-in viewers for plain text, HTML, GIF, and animated GIF).
- Have your Web browser configured to actually use these external viewers. For Netscape, this is done through the mailcap file. There are actually two: a system-wide and a personal; follow the menu chain *Options* ⟶ *General Preferences...* ⟶ *Helpers* to see where they are located (typically `/usr/local/lib/netscape/mailcap` and `~/.mailcap`).

If you are running on a Silicon Graphics computer like I am, you could use the following mailcap:

```
video/mpeg; mpeg_play %s >/dev/null 2>&1; \
        description="MPEG Movie"; \
        x11-bitmap="/usr/lib/Zmail/bitmaps/movie.xbm"

video/quicktime; movieplayer -nofork %s >/dev/null 2>&1; \
        compose=moviemaker %s; \
        description="QuickTime Movie"; \
        x11-bitmap="/usr/lib/Zmail/bitmaps/movie.xbm"

application/x-inventor; ivview %s >/dev/null 2>&1

x-world/x-vrml;  /usr/local/bin/webspace -remote %s -URL %u ; \
        description="VRML document"

application/postscript; ghostview %s; \
        description="PostScript Language";
```

The syntax is actually quite simple, it is "*MIME_type*; *viewer* `%s`", where `%s` will be substituted by the name of the temporary file containing the data. Netscape spawns external viewers using the Bourne Shell (`/bin/sh`), and "`>/dev/null 2>&1`" is the syntax for throwing away standard output and standard error, so that one does not have to click "OK" on all the messages like "could not allocate *whatever* color", or "played movie at *foo* frames per second".

Note, that the MIME types `text/plain`, `text/html`, and `image/gif` are *not* mentioned, and therefore they will be handled by Netscape itself. This is important, since an external GIF viewer might not be able to view animated GIFs – this is the case for `xv` and `imgview`. Note, that *both* the system-wide *and* your personal mailcap file has not to map `image/gif` to anything for this to work.

About the viewers I am using, and possible alternatives:

- `mpeg_play`[20]: public domain MPEG player.
  Alternative for playing MPEG movies: Not XAnim, since it can only play the I frames, not the B and P frames (for what I, B and P frames are, see e.g. Patel, Smith, & Rowe (1993)[21]).
- `movieplayer`: A Silicon Graphics specific program that plays among other things QuickTime movies. QuickTime movies using the *Animation* coding does not require a license, while those using *Cinepak* coding does. *Please note*, that you have to chose *Play Every Frame* in the *Options* menu before you press *Play* to get the intended result.
  Alternative for playing QuickTime movies: XAnim[22].
- `ivview`: A Silicon Graphics specific program that views / renders Inventor files.
  Alternative for viewing Inventor files: Nothing I know of. But since in this context Inventor and VRML are virtually the same, as long as you can view the latter, you are OK.

[20] http://bmrc.berkeley.edu/projects/mpeg/mpeg_play.html
[21] ftp://mm-ftp.cs.berkeley.edu/pub/multimedia/papers/MM93.ps.Z
[22] http://www.portal.com/~podlipec/home.html

- `webspace`[23]: A Silicon Graphics specific program (I think) that views / renders VRML files. Alternatives for viewing VRML files: VRweb (here[24] or here[25]); also check out what Yahoo has to say on VRML browsers.

- `ghostview`[26]: A public domain PostScript viewer. Silicon Graphics users can also use `xpsview`.

A final comment is about speed, e.g. the speed with which you can play a movie. Here at the Observatory, we have a number of Silicon Graphics Indys with typically 32 Mbytes of RAM and a 100 MHz R4000 CPU. They run as X-terminals, and the typical user will via them log in to a computer with more RAM and a faster CPU (say, an R8000). However, for playing movies, I have found a speed factor of 4–25 in favor of the Indy! Therefore, I advise you to run Netscape and thus the movie viewers locally, if you have the same type of computer setup as we do, or even better, to make the comparison yourself.

1.4 On using LaTeX2HTML

The HTML versions currently supported by major Web browser, such as Netscape, does not include markups for mathematical symbols and equations. Therefore, the only way to include these in the text is to somehow generate an image of each of them, and then inline these in the HTML document. LaTeX2HTML[27] can do this automatically: it takes the LaTeX source code, generates the necessary images, and outputs one or more HTML documents with these images inlined. Therefore, for making an astronomical HTML document, using LaTeX2HTML is almost inevitable.

LaTeX2HTML can be used in two conceptually different ways:

- Using a LaTeX file that was not written specifically with the intend of ending up as hypertext.

- Using a LaTeX file that was written with the intend of ending up as hypertext, using the LaTeX *package* (or *style*, if you will) `html`.

I have used it in the latter way. I have written the entire document in LaTeX using the `html` package. This has a number of advantages:

- An automatically generated table of contents.

- An automatically generated *navigation panel* on each HTML page.

- The ability to use mathematical symbols anywhere in the document.

- A paper ("LaTeX") version as a byproduct, i.e. one PostScript file with the intire document. Hyperlinks can either not be shown in this, or shown as footnotes giving the URL.

- The possibility of making macros using the LaTeX command `\newcommand`.

The disadvantages are:

- One has to run the source code through LaTeX2HTML before it can be viewed as hypertext.

The LaTeX source code for this document has been split up in a number of files, which the main file `index.tex`[28] refers to using the `\input` command. All these files are in this directory[29].

I want 3 different versions of the document:

[23] http://webspace.sgi.com/
[24] http://www.iicm.tu-graz.ac.at/Cvrweb
[25] http://www.3dsite.com/cgi/arch-VRML/VRweb.html
[26] http://www.cs.wisc.edu/~ghost/
[27] http://cbl.leeds.ac.uk/nikos/tex2html/doc/latex2html/latex2html.html
[28] http://www.astro.ku.dk/~milvang/RelViz/tex/index.tex
[29] http://www.astro.ku.dk/~milvang/RelViz/tex/

- HTML[30]
- PostScript[31]
- gzip'ed PostScript[32]

To make sure that they are always consistent, I have used this[33] small C-shell script that calls `latex` twice, calls `dvips`, calls `latex2html` (which for some reason deletes the `.dvi` file), and calls `gzip`. Besides the command line options to 2html, it is customized through a `.latex2html-init`[34] file.

When working with LaTeX2HTML, I had to struggle with a number of bugs and peculiarities in the program. These I have listed in appendix A.

[30] http://www.astro.ku.dk/~milvang/RelViz/
[31] http://www.astro.ku.dk/~milvang/RelViz/index.ps
[32] http://www.astro.ku.dk/~milvang/RelViz/index.ps.gz
[33] http://www.astro.ku.dk/~milvang/RelViz/do_index
[34] http://www.astro.ku.dk/~milvang/RelViz/.latex2html-init

Chapter 2

The Theory of General Relativity

2.1 The basic concepts

In the theory of general relativity, gravity is treated as a curvature of space-time instead of a force. A point in the 4-dimensional space-time is called an *event*. The trajectory of a body in space-time is called a *world line*.

The separation ds between to infinitely close events with coordinates (x^0, x^1, x^2, x^3) and $(x^0 + dx^0, x^1 + dx^1, x^2 + dx^2, x^3 + dx^3)$, respectively, is given by

$$ds^2 = g_{\mu\nu} dx^\mu dx^\nu, \tag{2.1}$$

where $g_{\mu\nu}$ is the *metric tensor*. We use the Einstein summation convention: repeated indices are to be summed over. If we wrote the above equation with explicit summation, it would be

$$ds^2 = \sum_{\mu=0}^{3} \sum_{\nu=0}^{3} g_{\mu\nu} dx^\mu dx^\nu.$$

We can chose any coordinates we like. For black holes, which a spherical symmetrical, time + spherical polar coordinates, (t, r, θ, φ), are well suited.

2.2 The Kerr metric

The *Schwarzschild metric* describes space-time around a spherical mass M which does not have any angular momentum J. The Schwarzschild metric is:

$$g_{\mu\nu} = \begin{pmatrix} -\alpha^2 & 0 & 0 & 0 \\ 0 & \alpha^{-2} & 0 & 0 \\ 0 & 0 & r^2 & 0 \\ 0 & 0 & 0 & r^2 \sin^2\theta \end{pmatrix}, \tag{2.2}$$

where the lapse function / gravitational redshift factor is:

$$\alpha = \left(1 - \frac{2M}{r}\right)^{1/2} \tag{2.3}$$

The more general *Kerr metric* describes space-time around a spherical mass M which *does* have an angular momentum J, such as a rotating black hole. The Kerr metric is:

$$g_{\mu\nu} = \begin{pmatrix} -\alpha^2 + \omega^2\varpi^2 & 0 & 0 & -\omega\varpi^2 \\ 0 & \rho^2/\Delta & 0 & 0 \\ 0 & 0 & \rho^2 & 0 \\ -\omega\varpi^2 & 0 & 0 & \varpi^2 \end{pmatrix}, \tag{2.4}$$

where the Boyer-Lindquist coordinates are:

$$\Delta = r^2 + a^2 - 2Mr \tag{2.5}$$
$$\rho^2 = r^2 + a^2 \cos^2\theta \tag{2.6}$$
$$\Sigma^2 = (r^2 + a^2)^2 - a^2\Delta\sin^2\theta \tag{2.7}$$
$$\varpi = (\Sigma/\rho)\sin\theta \tag{2.8}$$
$$\omega = \frac{2aMr}{\Sigma^2} \tag{2.9}$$

and where the rotation parameter a is defined by

$$a = \frac{J}{M}. \tag{2.10}$$

One has $0 \leq a \leq M$; an object with higher angular momentum than corresponding to $a = M$ cannot stick together.

The *horizon* is the surface which one when using the coordinates of a distant observer in flat space will never see anything pass. It is not a real singularity, but merely a coordinate singularity. It is given by

$$r_H = M^2 + \left(M^2 - a^2\right)^{1/2}. \tag{2.11}$$

Another interesting surface is that given by the *static limit* r_{stat}, within which is is compulsory to move along the direction of rotation, since space-time is rotating with so high speed, that even when one is moving with the speed of light in the opposite direction, one will still be moving in the direction of rotation. r_{stat} is given by

$$r_{stat} = M^2 + \left(M^2 - a^2\cos^2\theta\right)^{1/2}. \tag{2.12}$$

The area within the static limit but outside the horizon is called the *ergosphere*.

Note, that we in all of the above have used "natural units", *viz.* $c = G = 1$. It might, however, be useful to know what absolute sizes I will be talking about when I say that r is 2, or M is 1. This can be derived by writing the equation for r_H with c and G included:

$$r_H = \frac{GM}{c^2} + \left(\left(\frac{GM}{c^2}\right)^2 - \left(\frac{J}{Mc}\right)^2\right)^{1/2} \tag{2.13}$$

(Berry 1976). We see that length over mass is

$$\frac{G}{c^2} = \frac{6.67 \cdot 10^{-11}\mathrm{m}^3/(\mathrm{kg\,s}^2)}{(3.00 \cdot 10^8 \mathrm{m/s})^2} = 7.4 \cdot 10^{-28}\mathrm{m/kg} = 1.5\,\mathrm{km}/M_\odot \tag{2.14}$$

2.3 Geodesics

We want to be able to calculate the trajectory of a freely falling object (massive or mass less). In special relativity, and in a local intertial frame, a freely falling object moves in a straight line / has a constant velocity. From there, we can go to general coodinates by using the "trick" of changing from an ordinary derivative to a covariant derivative. This gives the following equation of motion (also known as the geodesics equation):

$$\frac{d^2x^\mu}{d\lambda^2} + \Gamma^\mu_{\nu\sigma}\frac{dx^\nu}{d\lambda}\frac{dx^\sigma}{d\lambda} = 0 \tag{2.15}$$

λ, the *affine parameter*, is a parameter along the curve, which is unique up to a transformation of the form $\lambda \mapsto a\lambda + b$ (Misner, Thorne, & Wheeler 1973).

A light ray, which I want to study for this project, moves with the speed of light, which is the same as it has a null separation, i.e. $ds^2 = 0$. If we have a metric like the Kerr one and solve for $dt \equiv dx^0$, we get a 2nd order equation:

$$\begin{aligned} 0 &= g_{00}dx^0dx^0 + g_{11}dx^1dx^1 + g_{22}dx^2dx^2 + g_{33}dx^3dx^3 + g_{03}dx^0dx^3 + g_{30}dx^3dx^0 \\ &= g_{00}\,(dx^0)^2 + 2g_{03}\,dx^0 + \left[g_{11}(dx^1)^2 + g_{22}(dx^2)^2 + g_{33}(dx^3)^2\right], \end{aligned} \tag{2.16}$$

where we have used the symmetry of the metric tensor, i.e. $g_{03} = g_{30}$. This equation can be readily solved. Notationwise, we could just as well have used $(dt, dr, d\theta, d\varphi)$ instead of (dx^0, dx^1, dx^2, dx^3).

Chapter 3

The code

In this chapter, the IDL programs (procedures), that do the null geodesic calculation, are described.

Note, that all the IDL programs written for this project are in this directory[1].

3.1 kerr

The IDL procedure `kerr`[2] was provided by Nordlund/Brandenburg. It calculates the Kerr metric and the connection coefficients.

3.2 pde

The IDL procedure `pde`[3] was provided by Nordlund/Brandenburg. It solves the geodesics equation (equation (2.15).

3.3 hyman

The IDL procedure `hyman`[4] was provided by Nordlund/Brandenburg. It does the time advance using the method of Hyman.

3.4 script1

The IDL procedure `script1`[5] is written by me, and is the cornerstone of the project (while still, it wouldn't be any use without the beforementioned lower layer procedures). It integrates the geodesics equation by calling `hyman` a number of times.

[1] http://www.astro.ku.dk/~milvang/RelViz/idl/
[2] http://www.astro.ku.dk/~milvang/RelViz/idl/kerr.pro
[3] http://www.astro.ku.dk/~milvang/RelViz/idl/pde.pro
[4] http://www.astro.ku.dk/~milvang/RelViz/idl/hyman.pro
[5] http://www.astro.ku.dk/~milvang/RelViz/idl/script1.pro

Chapter 4

Visualization setup

4.1 Inventor and VRML

Let me give an example of what an Inventor file looks like:

```
#Inventor V1.0 ascii

Separator {
    Separator {
        Material {
            diffuseColor 0.0 0.0 1.0 # Blue
            transparency 0.25        # 25% transparent
        }
        Sphere {                     # A sphere
            radius 1.5               # of radius 1.5
        }
    }
    Separator {
        Transform {
            translation 3.0 0.0 0.0  # We move a bit
        }
        Material {
            diffuseColor 0.0 1.0 0.0 # Green
        }
        Cube {                       # A Cube (of default dimensions)
        }
    }
}
```

As you can see, it is a plain text (ASCII) file, with nice commments in it. The *Separator* command acts as logical paranthesis. You can view the file here[1].

Now, if we in the above Inventor 3D scene replaces "Inventor" with "VRML" in the first line, we have a VRML 3D scene! You can view it here[2].

[1] http://www.astro.ku.dk/~milvang/RelViz/Intro/intro.iv
[2] http://www.astro.ku.dk/~milvang/RelViz/Intro/intro.wrl

4.2 The visualization code

4.2.1 `script1` again

4.2.2 `writegeometry`

4.2.3 `write_r_stat`

4.3 A raw geodesic

4.4 The 3D widgets

4.4.1 The coordinate system widget

foo

If nothing else is stated, the length of the axis shown in the following are 5 (in the units discussed above). This length is measured from where the 3 axis cylinders join to the end of the cone – therefore, if we place a small sphere at the point (5,0,0), it will look like this:

foo text

4.4.2 The horizon widget

foo

4.4.3 The static limit / ergosphere widget

foo

4.5 A final 3D scene

4.6 Making a movie

Let's take the 2nd movie below as an example of how the movies and the related data products come about. Note, that I will explain in section 5.2 what these files show.

1. The IDL procedure `mf_02`[3] calls `script1` 101 times, each time changing one or more parameters; in this case a/M is changed from 0.00 to 1.00. It writes 101 Inventor files, called `frame000.iv`, ..., `frame100.iv`.

2. Explorer[4] is used to make 101 RGB images from the 101 Inventor 3D scenes. In turn, the Inventor files are read using the module `ReadGeom`, rendered using the module `Render`, and written as an RGB image using the module `WriteImg`. A `For` module and two `AnimFile` modules are used to automate the process for all 101 frames. Before the run is started, the user interactively positiones the first (or last) 3D scene in `Render`, finding a nice view / camera setting. This is used for all 101 files, then. The `Render` window size is chosen to be 448 pixels by 448 pixels, i.e. multiple of 16. The technical details are described further in appendix B.

3. The IDL procedure `mf_02_an`[5] is used to annotate the images – in this case, the text "a/M=0.00", ..., "a/M=1.00" is overlayed in the lower right corner of the images. It calls `annotation_image` 101 times, and every time the temporary annotation image is added to the "real" image (which Explorer produced) using the Silicon Graphics UNIX command `add`. `add` has the side effect of

[3] http://www.astro.ku.dk/~milvang/RelViz/idl/mf_02.pro
[4] http://www.nag.co.uk/Welcome_IEC.html
[5] http://www.astro.ku.dk/~milvang/RelViz/idl/mf_02_an.pro

transforming the uncompressed RGB files that Explorer produced to RLE compressed RGB files, reducing the file size by a factor of ~ 4, from 600 kbyte ($448^2 \cdot 3$ byte) to 110–160 kbyte.

4. The Silicon Graphics UNIX command `dmconvert` is used to make an Animation coded QuickTime movie, `movie02.qt`[6], and an MPEG movie, `movie02.mpeg`[7]. This is done by this[8] C-shell script.

5. The Silicon Graphics UNIX command `togif` is used to make 101 GIF images, `frame000.gif`, ..., `frame100.gif` from the RGB images. The size of the GIF files is only 23 kbyte. The public domain UNIX program GIFMerge[9] is used to make an animated GIF, `movie02_anim.gif`[10]. This is done by this[11] C-shell script.

6. The UNIX program `xv` is used to make a PostScript file, `frame000.ps`[12] from the first RGB file. The black background color is changed to white, and the yellow annotation text color changed to black, well suited for printing.

7. All the RGB files are deleted.

8. The Silicon Graphics UNIX command `ivToVRML` (see here[13]) is used to make 101 VRML files, `frame000.wrl`, ..., `frame100.wrl` from the 101 Inventor files. This is done by this[14] C-shell script.

9. The Inventor and VRML files are compressed using `gzip`, reducing the file size by a factor of ~ 5.

10. All the visualization files in the directory in question[15] take up about 12 Mbyte of disk space.

[6] http://www.astro.ku.dk/~milvang/RelViz/Anim/02/movie02.qt
[7] http://www.astro.ku.dk/~milvang/RelViz/Anim/02/movie02.mpeg
[8] http://www.astro.ku.dk/~milvang/RelViz/Anim/02/Make_movies1
[9] http://www.iis.ee.ethz.ch/~kiwi/GIFMerge/
[10] http://www.astro.ku.dk/~milvang/RelViz/Anim/02/movie02_anim.gif
[11] http://www.astro.ku.dk/~milvang/RelViz/Anim/02/Make_movies2
[12] http://www.astro.ku.dk/~milvang/RelViz/Anim/02/frame000.ps
[13] http://webspace.sgi.com/tools/
[14] http://www.astro.ku.dk/~milvang/RelViz/Anim/02/Make_VRML
[15] http://www.astro.ku.dk/~milvang/RelViz/Anim/02/

Chapter 5

Visualization results

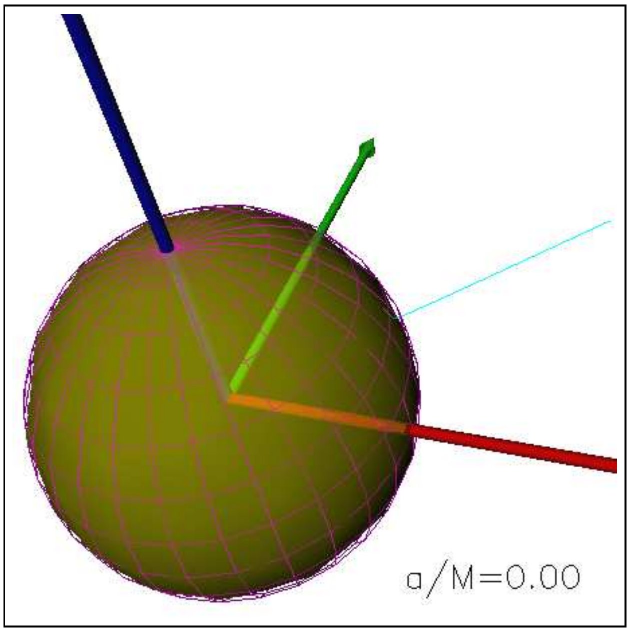
5.1 a/M varies

We start a photon in the equatorial plane, $(r, \theta, \varphi) = (5, \pi/2, \pi/4)$, and with direction radially inwards, $(dr, d\theta, d\varphi) = (-0.5, 0, 0)$. Therefore, it will stay in the equaorial plane.

We integrate the geodesic equation in 300 steps, using a step size $d\lambda = 0.20$

In this movie, we vary the rotation parameter a/M from 0.00 (no rotation) to 1.00 (the most extreme rotation possible). In every frame of the movie, we show a null godesic, that is, the trajectory of a photon, so it's proper time axis is along the geodesic, so to speak, and has nothing to do with the time axis of the movie, which shows a change in a/M.

The IDL script is `mf_01`[1].



Movie: QuickTime[2] (3.9 Mbyte), MPEG[3] (0.5 Mbyte), or animated GIF[4] (2.2 Mbyte)
First frame: Inventor[5], VRML[6], or GIF[7]

[1] http://www.astro.ku.dk/~milvang/RelViz/idl/mf_01.pro
[2] http://www.astro.ku.dk/~milvang/RelViz/Anim/01/a_over_M.qt
[3] http://www.astro.ku.dk/~milvang/RelViz/Anim/01/a_over_M.mpeg
[4] http://www.astro.ku.dk/~milvang/RelViz/Anim/01/a_over_M_anim.gif
[5] http://www.astro.ku.dk/~milvang/RelViz/Anim/01/a000.iv.gz
[6] http://www.astro.ku.dk/~milvang/RelViz/Anim/01/a000.wrl.gz
[7] http://www.astro.ku.dk/~milvang/RelViz/Anim/01/a000.gif

Last frame: Inventor[8], VRML[9], or GIF[10]
All files: this directory[11]

Note: this movie has a minor problem: frame 000 and 001 are identical, except for the annotation text. This is because something went wrong when running Explorer. The Inventor (and VRML) files *are* different.

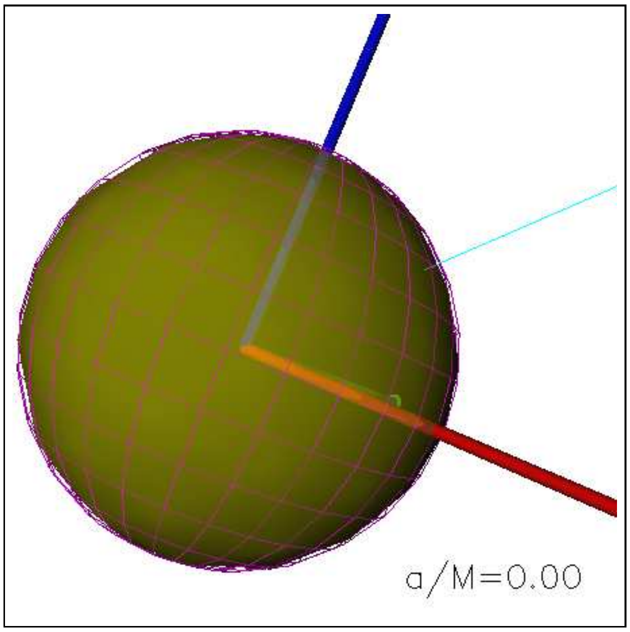
5.2 a/M varies (2nd example)

Now we start the photon 45 degress off the equatorial plane, $(r, \theta, \varphi) = (5, \pi/4, \pi/4)$, still with direction radially inwards, $(dr, d\theta, d\varphi) = (-0.5, 0, 0)$.

We integrate the geodesic equation in 300 steps, using a step size $d\lambda = 0.20$

In this movie, we again vary the rotation parameter a/M from 0.00 (no rotation) to 1.00 (the most extreme rotation possible).

The IDL script is `mf_02`[12].



Movie: QuickTime[13] (3.6 Mbyte), MPEG[14] (0.5 Mbyte), or animated GIF[15] (2.2 Mbyte)
First frame: Inventor[16], VRML[17], or GIF[18]
Last frame: Inventor[19], VRML[20], or GIF[21]
All files: this directory[22]

[8] http://www.astro.ku.dk/~milvang/RelViz/Anim/01/a100.iv.gz
[9] http://www.astro.ku.dk/~milvang/RelViz/Anim/01/a100.wrl.gz
[10] http://www.astro.ku.dk/~milvang/RelViz/Anim/01/a100.gif
[11] http://www.astro.ku.dk/~milvang/RelViz/Anim/01/
[12] http://www.astro.ku.dk/~milvang/RelViz/idl/mf_02.pro
[13] http://www.astro.ku.dk/~milvang/RelViz/Anim/02/movie02.qt
[14] http://www.astro.ku.dk/~milvang/RelViz/Anim/02/movie02.mpeg
[15] http://www.astro.ku.dk/~milvang/RelViz/Anim/02/movie02_anim.gif
[16] http://www.astro.ku.dk/~milvang/RelViz/Anim/02/frame000.iv.gz
[17] http://www.astro.ku.dk/~milvang/RelViz/Anim/02/frame000.wrl.gz
[18] http://www.astro.ku.dk/~milvang/RelViz/Anim/02/frame000.gif
[19] http://www.astro.ku.dk/~milvang/RelViz/Anim/02/frame100.iv.gz
[20] http://www.astro.ku.dk/~milvang/RelViz/Anim/02/frame100.wrl.gz
[21] http://www.astro.ku.dk/~milvang/RelViz/Anim/02/frame100.gif
[22] http://www.astro.ku.dk/~milvang/RelViz/Anim/02/

5.3 a/M varies (3rd example)

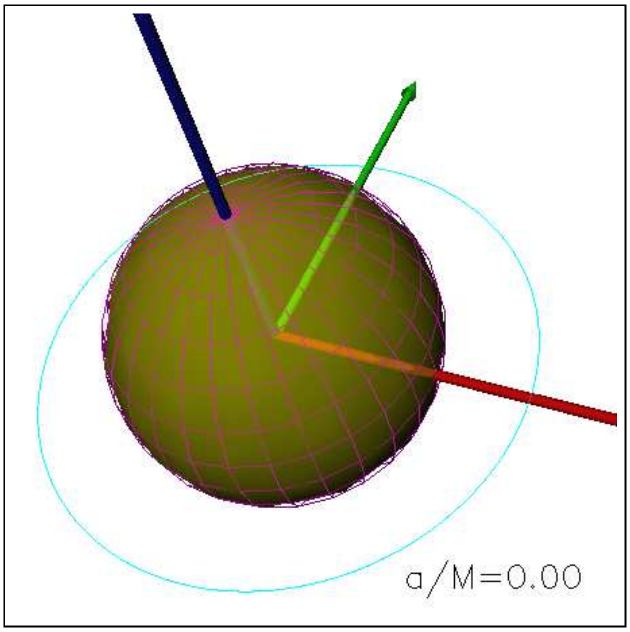
Now we start the photon in the equatorial plane somewhat closer to the center, $(r, \theta, \varphi) = (3, \pi/2, 0)$, but now with with initial direction perpendicular to the radial direction, and against the direction of the black hole, $(dr, d\theta, d\varphi) = (0, 0, -0.01)$.

We integrate the geodesic equation in 300 steps, using a step size $d\lambda = 0.20$

In this movie, we again vary the rotation parameter a/M from 0.00 (no rotation) to 1.00 (the most extreme rotation possible).

One sees, how the geodesics is dragged around with space-time.

The IDL script is `mf_03`[23].



Movie: QuickTime[24] (3.6 Mbyte), MPEG[25] (0.5 Mbyte), or animated GIF[26] (2.2 Mbyte)
First frame: Inventor[27], VRML[28], or GIF[29]
Last frame: Inventor[30], VRML[31], or GIF[32]
All files: this directory[33]

[23] http://www.astro.ku.dk/~milvang/RelViz/idl/mf_03.pro
[24] http://www.astro.ku.dk/~milvang/RelViz/Anim/03/movie03.qt
[25] http://www.astro.ku.dk/~milvang/RelViz/Anim/03/movie03.mpeg
[26] http://www.astro.ku.dk/~milvang/RelViz/Anim/03/movie03_anim.gif
[27] http://www.astro.ku.dk/~milvang/RelViz/Anim/03/frame000.iv.gz
[28] http://www.astro.ku.dk/~milvang/RelViz/Anim/03/frame000.wrl.gz
[29] http://www.astro.ku.dk/~milvang/RelViz/Anim/03/frame000.gif
[30] http://www.astro.ku.dk/~milvang/RelViz/Anim/03/frame100.iv.gz
[31] http://www.astro.ku.dk/~milvang/RelViz/Anim/03/frame100.wrl.gz
[32] http://www.astro.ku.dk/~milvang/RelViz/Anim/03/frame100.gif
[33] http://www.astro.ku.dk/~milvang/RelViz/Anim/03/

Chapter 6

Interactivity

6.1 Why interactivity?

Since the *genre* for this report is "*Web exhibition*", one can not only present movies and 3D scenes, but also the possibility of *interactively* work with the underlying code I have been using myself.

6.2 CGI scripts

CGI (Common Gateway Interface) scripts are the way one has to interface a Web *form* and some script, say, an IDL one. It is not trivial to write such a CGI script from scratch. I have been using the `cgi-lib.pl`[1] Perl library. You can view the source code for my major CGI script `script1.cgi`[2].

6.3 Do it yourself null geodesic

By using the interactive form below, you can create a 3D scene with your own null geodesics. This is a very good way of getting a feeling of how light rays move around a rotating black hole.

> In the HTML version there is an interactive form at this place. *Please go see it there!*.
> Summary of the form: one specifies
>
> - The black hole properties: M and a/M
> - The initial position of the photon (r, θ, φ)
> - The initial direction of the photon $(dr, d\theta, d\varphi)$
> - The numerical integration parameters
> - The desired 3D scene format
> - The drawing style of the geodesic
> - The color of the geodesic
> - The coordinate system widget parameters
> - The horizon widget parameters
> - The static limit / ergosphere widget parameters
>
> and then gets the output 3D scene.

[1] http://www.bio.cam.ac.uk/cgi-lib/
[2] http://www.astro.ku.dk/~milvang/cgi-bin/script1.cgi.txt

Chapter 7

Visualization results part two

Time's running out ... I did some experiments with multiple null geodesics in one 3D scene. They are in this directory[1] *Please check them out!* The IDL procedure used is `multi`[2].

[1] http://www.astro.ku.dk/~milvang/RelViz/test_more_rays/
[2] http://www.astro.ku.dk/~milvang/RelViz/idl/multi.pro

Chapter 8

Epilogue

My conclusion from doing this project is, that visualization helps troumendously in grasping the rich physics of black holes and the theory of general relativity. The only but severe downside is, that visualization is *extremely* time consuming, since it seems that one has to invent a number of wheels. Therefore, unfortunately, this *Web exhibition* is not as complete as I had wanted it to be.

Chapter 9

References and Bibiography

Berry, M. V. 1976, "Principles of Cosmology and Gravtitation", Cambridge: Cambridge University Press

Brandenburg, A., & Nordlund, Å. 1996, "Relativistic Computer Dynamics and Visualization", lecture notes, Copenhagen: NORDITA/NBIfAFG; available at `http://www.astro.ku.dk/~aake/teaching/rel/rel.ps.gz`

Lamport, L. 1986, "LaTeX: A Document Preparation System", Reading: Addison-Wesley

Misner, C. W., Thorne, K. S., & Wheeler, J. A. 1973, "Gravitation", San Francisco: W. H. Freeman & Company

Patel, K., Smith, B. C., & Rowe, L. A. 1993, "Performance of a Software MPEG Video Decoder", Proc. ACM Multimedia 93, Anaheim, CA, August 1993; available at `ftp://mm-ftp.cs.berkeley.edu/pub/multimedia/papers/MM93.ps.Z`

Thorne, K. S., Price, R. H., & MacDonald, D. A. 1986, "Black Holes: The Membrane Paradigm", New Haven and London: Yale University Press

Weinberg, S. 1972, "Gravitation and Cosmology", New York, London, Sydney, Toronto: John Wiley & Sons, Inc.

Appendix A

LaTeX2HTML bugs and peculiarities

During my work with LaTeX2HTML, I have found some things that causes errors in different ways, and which can be very time consuming to solve. For the interested reader, they are listed here, together with solutions / workarounds:

Tilde in URL: To make a hyperlink to a URL that has a tilde in it, one should write "~" as "\~{}", e.g.

```
\htmladdnormallinkfoot{my Web page}
    {http://www.astro.ku.dk/\~{}milvang/}
```

Tilde in URL for \htmladdimg: The above only works for `\htmladdnormallink` and `\htmladdnormallinkfoot`, not for `\htmladdimg`. I have not found any way to make `\htmladdimg` use a URL with a tilde in it. A workaround is to use the following instead:

```
\begin{rawhtml}
<img src="http://www.astro.ku.dk/~milvang/RelViz/Anim/01/a000.gif">
\end{rawhtml}
```

Note, that when using the `rawhtml` environment, a tilde should (of course) just be written as a tilde.

Underscore in URL: To make a hyperlink to a URL that has an underscore in it, one should write "_" as "_", e.g.

```
\htmladdnormallinkfoot{QuickTime}
    {http://www.astro.ku.dk/\~{}milvang/RelViz/Anim/01/a\_over\_M.qt}
```

Comments not allowed certain places: The following will give the error "Runaway argument?":

```
\begin{rawhtml} % bar
foo
\end{rawhtml} % bar
```

The workaround is not to place comments (i.e. "`% bar`" in the example) right there.

Getting text centered also in the HTML version: For some reason, LaTeX2HTML seems to ignore the `center` command when making the HTML version. A workaround is:

```
\begin{center}
\begin{rawhtml}
<center>
\end{rawhtml}
Here goes the text to be centered.
```

```
\begin{rawhtml}
</center>
\end{rawhtml}
\end{center}
```

Appendix B

Making a movie using explorer

This appendix will only be written if time allows ...

Just one note: If one wants to control the size of the Render window, and thus the images that comes out of WriteImg, one should look in the map file, with the place shown below, and know that the two numbers are 64 larger than the image size! In this case, we get 448 by 448. Know, that dmconvert requires images to be a multiple of 16 to make MPEGs.

```
(@5 =cxNRPanel [1]
    {   (title "Render")
        (maxiSize =cxWidgetSize
            {   (width 512)
                (height 512)
            }
        )
```