Development of a Relativistic Raytracer

Research Project

by

Julius Kunze

Dresden · 2013-04-28

Contents

Introduction

Programs that simulate the **visual relativistic effects at high velocities** are often specialized on particular simulation types and do not have a user interface. The intention of this project was to develop a program that is able to simulate these counterintuitive effects as universal as possible and make it possible for any user to perform own simulations. This project is based on a previously developed classical raytracer where extensibility was a top priority.

The equations that describe these effects are derived from **special relativity**. Based on these equations a relativistic raytracer was developed. The gravitational visual effects of the general relativity were excluded due to the complexity of this theory. The used algorithm restricts the simulation to **constant velocities of the observed objects**. Nevertheless, it is possible to simulate observers performing any accelerated motion. Especially, an algorithm was found and implemented that enables simulation of material surfaces and light sources that move relative to each other.

A graphical user interface was added to enable input of custom scene descriptions and output of video and image files. An interactive compiler for a self-defined functional description language was implemented to enable and simplify writing of complex 3D scene descriptions with motion information. An object-oriented and for the most part functional programming approach was used.

By applying many clean code developer principles extensibility of the program and productivity during the development process was achieved. Especially test driven development was used to validate mathematical and physical correctness of the program. The occurring effects, consisting of aberration effect, Doppler effect, searchlight effect and light travel time effect, were visualized by example simulations.

The program offers visualization of fast relative motions with a new universality and third party usability. The results could be used for various educational purposes, virtual reality and in the video game and film industry.

Part I

Methodology

In this part the effects that occur when objects are observed that have a velocity near that of light are derived from special relativity.

1 Derivation of a model for visual perception

To derive the visual effects at high velocities a suitable model for visual perception has to be found at first which will be done step by step.

1.1 Radiant power

Any physical object that emits electromagnetic radiation emits so called *radiant power* Φ (compare [RadiantPower]).

Different point of the surface can emit distinct amounts of radiation. One point of the surface can emit distinct amounts of radiation in different directions and the radiation of one point in one direction can distribute distinct over different wavelengths of the emitted light. The model of radiant power can not describe this behavior completely. Therefore, the concept of *spectral radiance* is needed, whose derivation follows.

1.2 Radiant intensity

Let $d\Phi = M \cdot dA$ be the radiant power that is emitted by a small surface part dA around a point of an object surface. $M = \frac{d\Phi}{dA}$ is called *radiant intensity* at that point (compare [RadiantIntensity]). The total radiant power Φ of a surface A can now be expressed as

$$\Phi = \int_A M \, dA$$

in terms of M at each surface point.

1.3 Solid angle

For the next step of the definition it is important to understand the concept of solid angle. The solid angle Ω of any surface A relative to a point O is the area of the projection of A on a sphere with radius 1 and midpoint O. (compare [Tipler, S. 674]) The following properties are important:

- If A is a part of a sphere surface with radius r and midpoint O then it is $\Omega = \frac{A}{r^2}$. (The SI unit of solid angle is steradian: $[\Omega] = 1sr = 1\frac{m^2}{m^2} = 1$.)
- If O is enclosed by A completely, it is$\Omega = 4\pi \cdot sr$.
- If A transforms into A' during central extension around O and Ω' is the solid angle of A' relative to O, it is $\Omega = \Omega'$.

A direction can be described by spherical coordinates with azimuth angle $\varphi \in [-\pi; \pi]$ and polar angle $\theta \in [0; \pi]$ where the orientation of this coordinate system can be defined arbitrarily. The solid angle of a rectangular segment with edge lengths $\sin\theta \cdot d\varphi$ and $d\theta$ at the coordinates (φ, θ) is $d\Omega = \sin\theta \cdot d\theta \cdot d\varphi$.

1.4 Radiance

When the following sections refer to a direction in connection with a surface segment with normal vector $\overrightarrow{n}$, let a spherical coordinate system (φ, θ) be defined implicitly, where the direction $\overrightarrow{n}$ matches with$\theta = 0$. Related to a point on a surface and a specific direction with $\theta < \frac{\pi}{2}$ (»to the front«) let

$$d^2\Phi = L \cdot \cos\theta \cdot dA \cdot d\Omega$$

be the radiant power that is emitted by the perpendicular projection $\cos\theta \cdot dA$ of the surface dA into the given direction in the given solid angle $d\Omega$ around the given direction.

$$L = \frac{d^2\Phi}{\cos\theta \cdot dA \cdot d\Omega}$$

is called *radiance* (compare [Radiance]) at this point in this direction (see figure (1.1)).

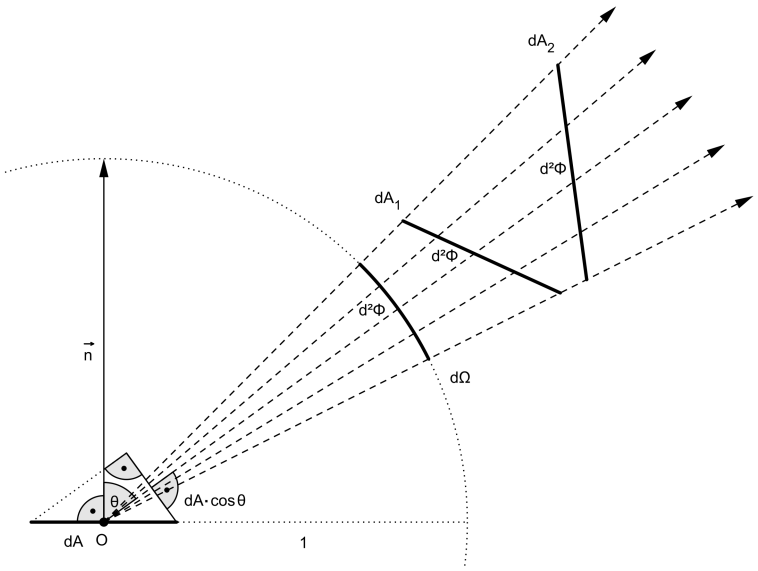

Figure 1.1: If the radiance of the shown direction is constantly L, all surfaces that have the solid angle $d\Omega$ relative to the point O (for example dA_1 and dA_2) are flooded by the same radiant power $d^2\Phi = L \cdot \cos\theta \cdot dA \cdot d\Omega$. (The reason is: If r is the distance from O, the photon density decreases with r^2 while the area of a surface increases with r^2 at a constant solid angle $d\Omega$.) (The figure was created by myself with [GeoGebra].)

Radiance is not directly based on dA but on the perpendicular projection $\cos\theta \cdot dA$ of the surface dA (which is significant for the radiant power $d^2\Phi$) to make L only dependent on physical surface properties of the emitting material.

The radiant intensity M of a surface point and the radiance $L(\varphi, \theta)$ of that point in the direction (φ, θ)

are connected by

$$M = \frac{d\Phi}{dA} = \int_{\Omega} L \cos\theta d\Omega = sr \cdot \int_{0}^{\frac{\pi}{2}} d\theta \int_{-\pi}^{\pi} d\varphi \, L(\varphi, \theta) \cos\theta$$

where Ω is the complete solid angle ($2\pi \cdot sr$) above the emitting surface segment.

1.5 Spectral radiance

In general the radiance M of a surface point in a specified direction is emitted by several wavelengths. In a small wavelength range from λ to $\lambda + d\lambda$ a radiance $dL = L_\lambda d\lambda$ is emitted. $L_\lambda = \frac{dL}{d\lambda}$ is called *spectral radiance* (compare [Radiance]) at this point in this direction at the wavelength λ. The radiance L and the spectral radiance $L_\lambda(\lambda)$ at the wavelength λ are connected by

$$L = \int_{0}^{\infty} L_\lambda(\lambda) d\lambda$$

It is not important for the definition of spectral radiance how the radiant power is produced. The radiant power that is emitted by a surface part can be produced by thermal emission, by transmission (in general in connection with refraction) or reflection of electromagnetic radiation or other processes.

1.6 The plenoptic function

The previously introduced quantities (radiation power, radiation intensity, radiance, spectral radiance) were always related to the *emission* of radiation. Completely analogous these sizes can express the *absorption* of radiation. To achieve that the light path of the above considerations must simply be reversed.

Assume that the absorbed spectral radiance

$$L_\lambda = \frac{d^3\Phi}{dA_\perp \cdot d\Omega \cdot d\lambda}$$

is known for each event $E(x, y, z, t)$ (= at the point of time t at the point in space (x, y, z)) in each direction with azimuth angle $\varphi \in [-\pi; \pi]$ and polar angle $\theta \in [0; \pi]$ and for each wavelength $\lambda \in (0; \infty)$ where $d^3\Phi$ is the radiant power that is absorbed by the surface $dA_\perp$ (perpendicular to the direction at E) from the solid angle $d\Omega$ around the direction (φ, θ) and wavelength range $d\lambda$ around λ.

$L_\lambda(\varphi, \theta, \lambda)$ can now be understood as a model of visual perception at a specified event; the function $L_\lambda(x, y, z, t, \varphi, \theta, \lambda)$ can be used as a model for every visual perception that an environment can cause. This function is called plenoptic function ([Adelson, S. 5]). Based on this model, visual relativistic effects can be derived.

2 Derivation of visual relativistic effects

In this part, the visually perceived picture of objects that move with a constant high velocity relative to an observer will be derived. The basis for this derivation is the theory of special relativity and previously introduced model as an abstraction for visual perception.

2.1 Consideration of a single photon

It is useful for the derivation to investigate the Lorentz transformation of a single photon at first. Consider a photon in vacuum (modeled as a mathematical ray with time parameter t which is a set of space time events) that passes through the events $E_0 = (x_0, y_0, z_0, t_0)$ and $E_1 = (x_1, y_1, z_1, t_1)$ with $t_1 > t_0$ (see figure (2.1)) in the observer reference frame S.

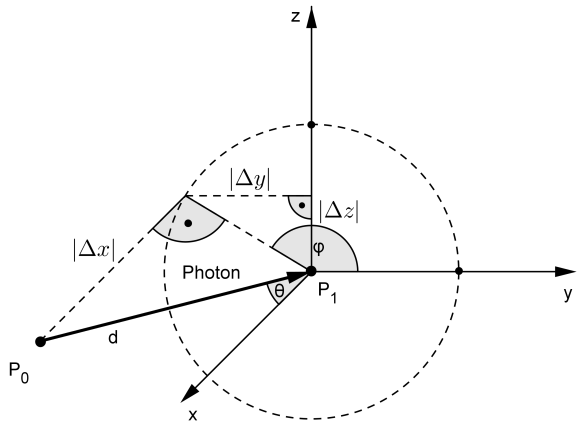

Figure 2.1: A photon passes through the points $P_0(x_0, y_0, z_0)$ and $P_1(x_1, y_1, z_1)$. (For simplicity P_1 is depicted in the coordinate origin of S but its location is arbitrary.) Let (φ, θ) be a spherical coordinate system with origin P_1 so that the angle between the x-axis and the photon path is represented by the polar angle $\theta \in [0; \pi]$ (with $\cos\theta = -\frac{\Delta x}{d}$) and the angle of the projection of the photon path on the yz-plane and the y-axis is represented by the azimuth angle $\varphi \in [-\pi; \pi]$. (The figure was created by myself with [GeoGebra].)

To move from P_0 to P_1 the photon needs the time

$$\Delta t = t_1 - t_0 = \frac{d}{c},$$

where

$$d = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2} = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (y_1 - y_0)^2}$$

is the distance between these point. From this it follows $t_1 = \frac{d}{c} + t_0$ (1).

Consider a second reference frame S' that moves with a constant velocity v in x-direction relative to S where the origin of S at $t = 0$ and the origin of S' at $t' = 0$ represent the same event. Let the coordinate systems of S be defined analogously in S'. Let $\beta := \frac{v}{c}$ and $\gamma := \frac{1}{\sqrt{1-\beta^2}}$. It is known that $-1 < \beta < 1$ and $\gamma > 1$. Because of the Lorentz transformation (compare [Tipler, S. 1159]) the same events of the photon in S' are:

$$E_0' = (x_0', y_0', z_0', t_0') = \left(\gamma(x_0 - vt_0),\, y_0,\, z_0,\, \gamma\left(t_0 - \frac{vx_0}{c^2}\right)\right)$$

and

$$E_1' = (x_1', y_1', z_1', t_1') = \left(\gamma(x_1 - vt_1),\, y_1,\, z_1,\, \gamma\left(t_1 - \frac{vx_1}{c^2}\right)\right).$$

With (1) it follows

$$x_1' = \gamma\left(x_1 - v\frac{d}{c} - vt_0\right) = \gamma\left(x_1 - \beta d - vt_0\right)$$

and from that

$$\Delta x' = x'_1 - x'_0 = \gamma\left(x_1 - \beta d - v t_0\right) - \gamma(x_0 - v t_0) = \gamma\left(\Delta x - \beta d\right).$$

From $\Delta y = \Delta y'$ and $\Delta z = \Delta z'$ it follows $\varphi = \varphi'$. Furthermore it is $\cos\theta = -\frac{\Delta x}{d}$ and $\cos\theta' = -\frac{\Delta x'}{d'}$ (see figure (2.1)). Now it is

$$\begin{aligned}\cos\theta' &= -\frac{\Delta x'}{d'} = -\frac{\gamma\left(\Delta x - \beta d\right)}{d'} \\ &= -\frac{\gamma\left(\Delta x - \beta d\right)}{\sqrt{\gamma^2\left(\Delta x - \beta d\right)^2 + \Delta y^2 + \Delta z^2}}.\end{aligned}$$

From $\gamma = \frac{1}{\sqrt{1-\beta^2}} > 0$ it follows

$$\begin{aligned}\cos\theta' &= -\frac{\Delta x - \beta d}{\sqrt{(\Delta x - \beta d)^2 + \frac{d^2 - \Delta x^2}{\gamma^2}}} \\ &= -\frac{-\cos\theta - \beta}{\sqrt{\frac{(\Delta x - \beta d)^2 + (1-\beta^2)(d^2 - \Delta x^2)}{d^2}}} \\ &= \frac{\cos\theta + \beta}{\sqrt{(-\cos\theta - \beta)^2 + (1-\beta^2)(1-\cos^2\theta)}} \\ &= \frac{\cos\theta + \beta}{\sqrt{\cos^2\theta + 2\beta\cos\theta + \beta^2 + 1 - \cos^2\theta - \beta^2 + \beta^2\cos^2\theta}} \\ &= \frac{\cos\theta + \beta}{\sqrt{1 + 2\beta\cos\theta + \beta^2\cos^2\theta}} \\ &= \frac{\cos\theta + \beta}{|1 + \beta\cos\theta|}.\end{aligned}$$

With $1 + \beta\cos\theta \geq 0$ it follows

$$\cos\theta' = \frac{\cos\theta + \beta}{1 + \beta\cos\theta}.$$

The transformation of the direction of movement of the photon is expressed by

$$\theta' = \arccos\left(\frac{\cos\theta + \beta}{1 + \beta\cos\theta}\right),\ \varphi' = \varphi \tag{2.1}$$

(This effect is visualized in figure (2.2)).

Similarly it can be shown (compare [Radiance, S. 3]) that for the wavelengths λ and λ' of the photon in S and S' it is

$$\frac{\lambda'}{\lambda} = \gamma_\theta$$

where $\frac{1}{\gamma(1+\beta\cos\theta)}$ is written as γ_θ for a shorter notation.

From this it follows that the transformation of the wavelength of the photon is expressed by

$$\lambda' = \gamma_\theta \lambda \tag{2.2}$$

Special case example: If the photon has the relative movement direction of S' in S (in S' »it comes exactly from behind«) for v, $\beta > 0$ it is $\theta = \pi$ and $\frac{\lambda'}{\lambda} = \frac{1}{\gamma(1+\beta\cos\pi)} = \frac{\sqrt{1-\beta^2}}{1-\beta} = \sqrt{\frac{1+\beta}{1-\beta}}$ which means that the wavelength in S' is elongated relative to S. If the photon comes from the opposite direction (in S' »exactly from the front«) it is $\theta = 0$ and $\frac{\lambda'}{\lambda} = \frac{1}{\gamma(1+\beta\cos 0)} = \frac{\sqrt{1-\beta^2}}{1+\beta} = \sqrt{\frac{1-\beta}{1+\beta}}$ which means that the wavelength in S' is shortened relative to S.

2.2 Consequences for visual perception

Let $L_\lambda(\varphi, \theta, \lambda)$ be the plenoptic function in S at E_1 and $L'_\lambda(\varphi', \theta', \lambda')$ be the plenoptic function in S' at E'_1; let spherical coordinate systems be defined as above. It is the aim to express L'_λ in terms of L_λ because then it is possible to calculate the visual impression that would be observed at at a high velocity (modeled by L'_λ) from the visual impression at rest (modeled by L_λ). Eventually, this enables the simulation of the corresponding effects.

2.2.1 Aberration effect

Every photon that passes E_1 in S is transformed as discussed. Because of equation (2.1) for every direction with coordinates (φ, θ) it is:

$$\varphi' = \varphi,\ \theta' = \arccos\left(\frac{\cos\theta + \beta}{1 + \beta\cos\theta}\right) \tag{2.3}$$

Because of $\cos\theta' = \frac{\cos\theta+\beta}{1+\beta\cos\theta} \Leftrightarrow (1 + \beta\cos\theta)\cos\theta' = \cos\theta + \beta \Leftrightarrow \cos\theta' - \beta = \cos\theta(1 - \beta\cos\theta') \Leftrightarrow \cos\theta = \frac{\cos\theta'-\beta}{1-\beta\cos\theta'}$ it follows the inverse transformation

$$\varphi = \varphi',\ \theta = \arccos\left(\frac{\cos\theta' - \beta}{1 - \beta\cos\theta'}\right) \tag{2.4}$$

This transformation equation expresses the so-called *aberration effect* completely. It is visualized by figure (2.2): Speaking vividly, the perceived picture is compressed in the direction of movement and pulled apart opposite to the direction of movement compared to that of an observer at rest (see figure (9.1) in appendix, picture 2).

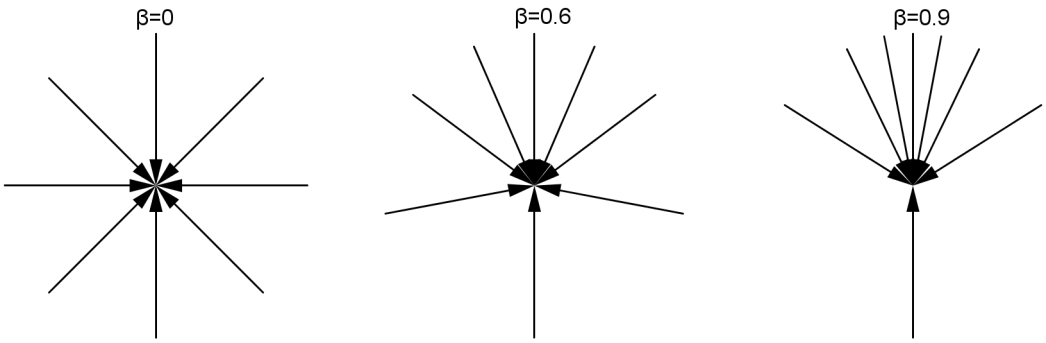

Figure 2.2: Photons arrive at an observer at rest from all sides (left figure). If the observer has a constant velocity upwards (mid and right figure) the observed direction of the same photons changes due to equation (2.3). (The figure was created by myself with [GeoGebra].)

2.2.2 Doppler effect

From equation (2.2) it follows for all wavelengths λ:

$$\lambda' = \gamma_\theta \lambda \tag{2.5}$$

From this it follows the inverse transformation

$$\lambda = \gamma_\theta^{-1} \lambda' \tag{2.6}$$

This transformation equation expresses the so-called *Doppler effect* ([Radiance, S. 1]) completely. Speaking vividly, the perceived picture is blue-shifted in the direction of movement and red-shifted opposite to the direction of movement compared to that of an observer at rest (see figure (9.1) in appendix, picture 3).

2.2.3 Searchlight effect

It is possible to show (compare [Weißkopf, S. 2f]) that for all directions (φ, θ) and wavelengths λ it is:

$$L'_\lambda(\varphi',\, \theta',\, \lambda') = \gamma_\theta^{-5} \cdot L_\lambda(\varphi,\, \theta,\, \lambda) \tag{2.7}$$

From this it follows the inverse transformation

$$L_\lambda(\varphi,\, \theta,\, \lambda) = \gamma_\theta^{5} \cdot L'_\lambda(\varphi',\, \theta',\, \lambda') \tag{2.8}$$

These transformation equations expresses the so-called *searchlight effect* ([Radiance, S. 1]) completely. Speaking vividly, the perceived picture is brighter in the direction of movement and darker opposite to the direction of movement compared to that of an observer at rest (see figure (9.1) in appendix, picture 4).

2.2.4 Complete transformation of the plenoptic function

Insertion of equations (2.4) and (2.6) into (2.7) returns the complete transformation equation of the plenoptic function:

$$L'_\lambda(\varphi',\, \theta',\, \lambda') = \gamma_\theta^{-5} \cdot L_\lambda\left(\varphi',\, \arccos\left(\frac{\cos\theta' - \beta'}{1 - \beta\cos\theta'}\right),\, \gamma_\theta^{-1}\lambda'\right) \tag{2.9}$$

Insertion of equations(2.3) and (2.5) into (2.8) returns the complete inverse transformation:

$$L_\lambda(\varphi,\, \theta,\, \lambda) = \gamma_\theta^{5} \cdot L'_\lambda\left(\varphi,\, \arccos\left(\frac{\cos\theta + \beta}{1 + \beta\cos\theta}\right),\, \gamma_\theta\lambda\right) \tag{2.10}$$

The transformation equation (2.9) contains all visual effects that can be observed at extremely high velocities: Aberration, Doppler and searchlight effect. Based on this transformation equation, a simulation algorithm can be developed.

3 Design of a simulation algorithm for visual relativistic effects

3.1 Raytracing

Raytracing aims to render a 2D picture from a 3D scene consisting of the location of an observer, location and material information of observed 3D objects and light sources (for example triangles, spheres...) in a virtual 3D coordinate system (see figure (3.1)).

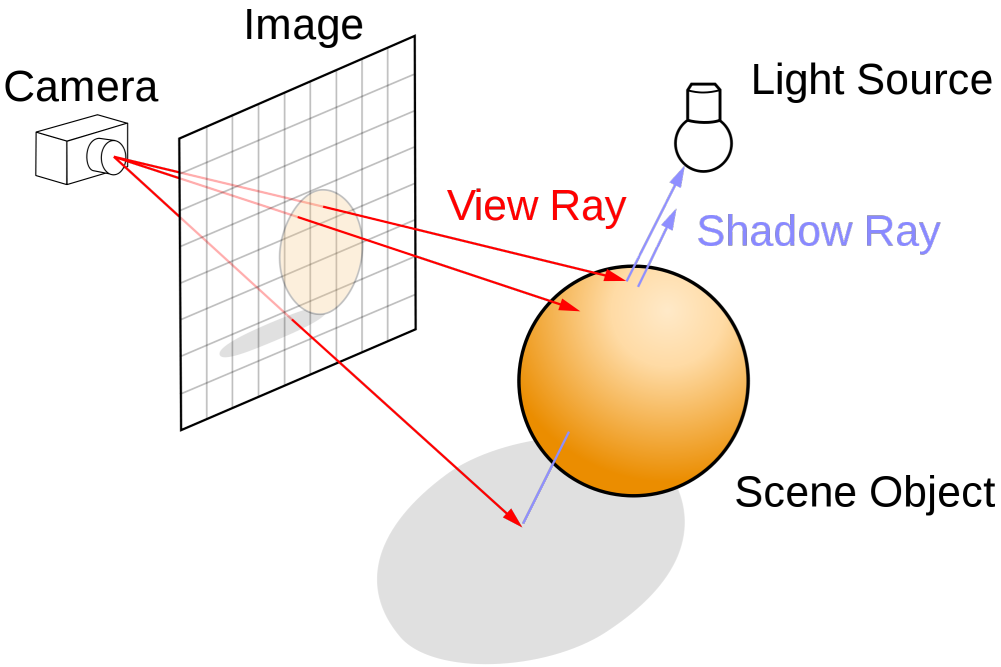

Figure 3.1: Basic principle of raytracing: A *sight ray* (or »view ray«) is calculated for each pixel that starts at the observer (»camera«) location and points into the observed 3D scene. The pixel is colored depending on the material of the objects that are hit by the sight ray first. The intersection calculations are possible with arbitrary shaped surfaces like planes, triangles, spheres, cylinders or cuboids. However, more complex shaped surfaces tend to cause more computational effort. Source: `http://en.wikipedia.org/wiki/File:Ray_trace_diagram.svg` (Accessed December 19, 2011)

Raytracing is based on the principle of reversibility of light which allows to track *light rays* moving towards the observer by calculating the path that reverse *sight rays* take that start at the observer and follow the same physical laws as light rays. Therefore various extensions of this algorithm can be implemented easily to enable shadows and direct illumination, reflection and refraction (*recursive raytracing*) or scattering of light (*path tracing*) for example. This good extensibility is a big advantage compared to other rendering algorithms (especially *polygon rendering*) and the reason why the classical raytracer developed the previous project can be extended to a relativistic raytracer naturally. The required steps of extension are explained in the following subsections.

3.2 Spectral raytracing

Common computer screens use the *RGB* format to display colors where color is generated by three color component numbers (for red, green and blue) between 0 and 255.

This representation of color does not contain the full information of a physical light spectrum. If the spectrum is needed for a calculation (for example for the relativistic Doppler effect), an other representation

has to be used. For this, a function of spectral radiance $L_\lambda(\lambda)$ depending on wavelength λ is suitable. This is exactly the distinctive mark of *spectral raytracing* (compare [Raytracing]): A sight ray is not send out to return an RGB color but a light spectrum of the form $L_\lambda(\lambda)$. Before such a spectrum can be displayed on a screen, it has to be converted into RGB format by a converter function. In the project, a light spectrum to RGB converter was implemented based on the picture data of figure (3.2).

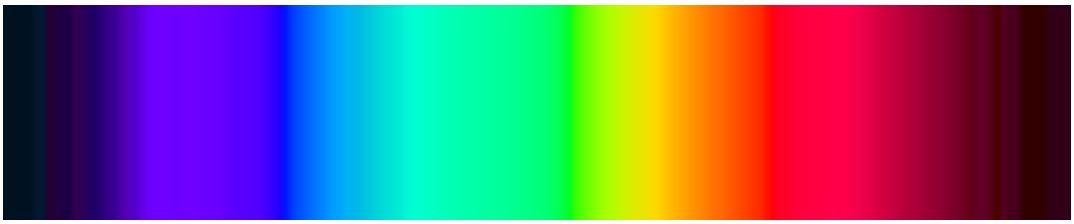

Figure 3.2: RGB representation of spectral colors in the visible light spectrum (wavelength $380nm$ to $710nm$) assuming sRGB 1.10 device characteristics. Source: `http://en.wikipedia.org/wiki/File:Ray_trace_diagram.svg` (Accessed December 19, 2011)

In principle, any raytracing algorithm (for example a recursive raytracing or path tracing) that is based on RGB colors can be converted so that it is based on light spectra $L_\lambda(\lambda)$. That means that every raytracer can be converted into a spectral raytracer.

3.3 Relativistic raytracing

Relativistic Raytracing (compare [Raytracing]) is an extension of the raytracing algorithm that allows to simulate the visual effects of the theory of relativity. As reasoned in the introduction, the following considerations are limited to special relativity.

3.3.1 Relativistic raytracing with a single object reference frame

The simplest relativistic raytracing algorithm is needed for the case when the visual effects are simulated occurring when objects - modeled as material surfaces and light sources - (at rest in inertial reference frame S') are observed by an extremely fast observer (at rest in reference frame S) and these objects do not move relative to each other. In connection with relativistic raytracing it was very helpful to replace the geometric model of a sight ray (= set of space locations, as used in classical raytracing algorithms) by a time-dependent model (= set of space time events). This time-dependent model of sight rays was used in the following algorithms.

For each pixel of the image the following procedure is used:

- Depending on the observer location and orientation and the pixel position, the corresponding sight ray (in the observer reference frame S) is calculated.
- The sight ray is transformed from S to S' using the equation (2.4) (= taking into account aberration effect).
- Classical spectral raytracing is used to determine the corresponding light spectrum $L_\lambda(\lambda)$.
- The spectrum is inverse transformed using the equations (2.5) and (2.7) (= taking into account the Doppler and searchlight effect).
- To enable display the spectrum on a screen the spectrum has to be converted into an RGB color by a conversion function.

3.3.2 Relativistic raytracing with multiple object reference frames

The previously used algorithm does not allow simulation of objects that move relative to each other. To enable this, the following algorithm was developed by myself (simplified description):

- Depending on the observer location and orientation and the pixel position, the corresponding sight ray s of the observer frame S is calculated.
- The corresponding sight rays s'_i in all object reference frames O_i (with index i from $\{1, ..., n\}$) are calculated by transformation analogously to the previously introduced algorithm.
- For each object reference frame O_i the event E'_i of the first intersection of s'_i with the material surface at rest in O_i is calculated.
- The corresponding events E_i in S are calculated by inverse transformation.
- Only the event $E*$ with the maximum time component was actually hit by the sight ray. Because of that the other events can be discarded.
- If the material of the hit surface scatters in $E*$, the following procedure is used:
  - For each object reference frame O_i the corresponding transformed event E_i*' is calculated.
  - For each object reference frame O_i a secondary sight ray (»shadow ray«) from E_i*' to every point light source at rest in O_i is calculated.
  - Each secondary sight ray is transformed into each other object reference frame to ensure that no other surface is located between the light source and the surface intersection point (at E_i*') and therefore surface is illuminated by the point light source. If this is the case, the accordingly transformed spectrum of the light source is added to the observed spectrum.
- If the material reflects or refracts the light at $E*$, the algorithm is used recursively for the reflected or refracted sight rays.
- The inverse transformation of the obtained spectrum and the display procedure is analogous to the previously introduced algorithm.

The algorithm enables general simulation of direct illumination and hard shadows of arbitrary material surfaces and point light sources that move relative to each other with constant velocities. It is the relativistic analogue to the classical recursive raytracing. This algorithm enables the simulation of the *light travel time effect*, which is caused by the fact that light does not propagate immediately but has a limited velocity. This causes effects like delayed shadows and asymmetrical appearance of symmetrical objects.

3.3.3 Relativistic raytracing with accelerated observers

Contrary to accelerated *observed objects*, accelerated *observers* can be implemented within this approach. The key step is to assume a relative velocity at each point of time which does not affect the visual perception. This enables the usage of Lorentz transformation and therefore the usage of the above algorithms. The simulation of an observer that would measure a constant acceleration and the simulation of a constantly rotating observer were implemented as examples of this possibility.

4 Implementation process and principles

The following elements and procedures were used to develop the simulation program:

- Microsoft Visual Basic .NET 4.0 as object-oriented and statically typed programming language were used because object-orientation allows to write testable, easily understandable and extensible source code even for big program architectures and static typing exposed many possible faults early so that they can be fixed fast.
- The development environment Microsoft Visual Studio 2010 Ultimate [1] to simplify writing and organization of source code.
- The development environment extension JetBrains ReSharper 6.0 [2] that provides many useful tools for source code navigation, refactoring, unit testing and quality management and increases programming productivity dramatically.
- The library avilib [3] was embedded to slice picture files into video files.
- To implement examples for accelerated observers, the equations in [Gibbs] and [Murray] were used.

As many clean code developer principles (that are described in [Martin] and [Westphal]) as possible were applied to achieve, among other things, extensibility of the program and productivity during the development process:

- **Test driven development** (»write automated tests before implementation«) was used to validate correctness of the program and detect errors as early as possible and fix them efficiently. NUnit 2.5.5 [4] was used as testing library. Furthermore, whenever a bug was supposed in the raytracing library, a matching automated unit test was written. If the test failed, it was fixed immediately. These practices caused that over 200 unit test were written to validate the correctness of the program's algorithms.
- Very few decisions were made before the implementation of the code. An **iterative process** (»try, fail, correct, ...«) was used to allow a flexible development of the application and a fast reaction on design problems.
- Git [5] together with Git Extensions 2.28 [6] as graphical user interface were used as **source control system** to keep all earlier program versions comfortably and avoid loss of code.
- **Programming principles** like the "Don't repeat yourself" principle, the single responsibility principle, the composition over inheritance principle and the principle of least astonishment were applied where possible.
- Common naming and code style **conventions** were used to improve readability of source code.

[1] `http://www.microsoft.com/visualstudio/en-us/products/2010-editions/ultimate/overview` (Accessed December 19, 2011)
[2] `http://www.jetbrains.com/resharper/` (Accessed December 19, 2011)
[3] `http://www.vbforums.com/showthread.php?t=395812` (Accessed December 19, 2011)
[4] `http://www.nunit.org/?p=download` (Accessed December 19, 2011)
[5] `http://git-scm.com/` (Accessed December 19, 2011)
[6] `http://code.google.com/p/gitextensions/` (Accessed December 19, 2011)

Part II

Results

5 Sources and program

An interactive simulation program for all special relativistic visual effects with an intuitive user interface was developed. The application consists of the following components:

- An (either recursive or diffuse) spectral relativistic raytracer. This is the core of the program that performs the actual simulation. It has the following capabilities:
  - It is possible to include any amount of geometric objects (including planes, triangles, spheres, cylinders, cuboids and any unions and intersections of these 3D objects that can consist of any surface material) and point light sources (that can send out any intensity spectrum) into the relativistic simulations.
  - These objects can move relative to each other with arbitrary constant velocities.
  - All objects properties (materials of surfaces, illumination) can be time-dependent.
  - The observer can perform an arbitrary (in general accelerated) motion.
- A functional[7] description language that enables user-defined 3D scenes.
- An interactive compiler that simplifies the input of 3D scenes by automatically updated error output, completion suggestions and documentation. This compiler converts the textual description of a scene into an internal format which is used by the raytracer to render the picture or video (»it compiles the scene description«).
- A graphical user interface that links all program components (see figure (9.4) in appendix). This contains:
  - A tab to create, load, change and save scene definition files which contains the interactive compiler.
  - A tab to estimate the needed render time for pictures and videos to start and cancel the rendering process.
  - A tab to save the rendered pictures and videos files (.bmp, .png and .avi in various codecs are supported formats).
  - A help window that explains the basics of the scene description language and makes the user interface mostly self-explanatory.

The current source code, the executable application, all example simulations listed below and more materials are publicly available on the internet under `https://github.com/JuliusKunze/Fusion`.

[7] In contrast to imperative programming languages, a functional programming language solely defines objects (for example numbers, vectors, spheres, functions...) that can not be changed after the definition. (Example: If a number a is defined by $a := 0$ in a functional programming language, it can not 5 can not be assigned to a anymore. a is 0 in the whole time then.) This design causes the language to be time-independent which simplifies the process of definition.

6 Example simulations

Various pictures and videos were rendered as example simulations. (All videos were rendered with a duration of 10 seconds and 24 frames per second. They were converted from .avi to .mp4 to save disk space and to make the files portable.) The most important are:

GATE [0 TO 3].PNG (See figure (9.1) in appendix.) The four pictures show the flight through a simple geometric scene at $0.8c$. The scene consists of ideally diffuse white surfaces that are illuminated by sunlight (modeled as a black body spectrum with an effective temperature of $5778K$). The first picture was simulated neglecting all relativistic effects. Aberration effect (picture 2), Doppler effect (picture 3) and searchlight effect (picture 4) are taken into account one by one. (4000×4000 pixels)

GATE.MP4 The video shows the flight through a simple geometric environment at $0.8c$ (scene from the previous simulation). Searchlight effect is neglected for clarity. (2000×2000 pixels)

MOVED MONOCHROMATIC LIGHT SOURCE.MP4 The video shows a cube and a yellow monochromatic point light source flying above a plane at $0.3c$. The light source is fixed in a constant distance above the center of the cube. All surfaces are ideally diffuse and white. Searchlight effect is neglected for clarity. The observer is at rest relative to the plane. (800×800 pixels)

MOVED MONOCHROMATIC LIGHT SOURCE.PNG (See figure (9.3) in appendix.) The picture is frame of MOVED MONOCHROMATIC LIGHT SOURCE.MP4 in high resolution. (4000×4000 pixels)

SHADOW.MP4 The video shows a cube flying above a plane with $0.3c$. All surfaces are ideally diffuse and white. Searchlight effect is neglected for clarity. The observer is at rest relative to the plane. At his position, a point light source emits black body radiation with an effective temperature of $5000K$. (750×750 pixels)

SUN [0 TO 3].MP4 The four videos simulates a flight passing the sun (modeled as black body with an effective temperature of $5778K$) at $0.5c$. The video is in time lapse. Grid lines on the sun are added for better orientation. The observer's view is fixed on the center of the sun. The first video was simulated neglecting all relativistic effects. Aberration effect (video 2), Doppler effect (video 3) and searchlight effect (video 4) are taken into account one by one. (1000×1000 pixels)

TWO CUBES [1, 2].PNG (See figure (9.2) in appendix.) The pictures show the same scene at the same time from two different perspectives: Two cubes on a checkerboard that are at rest relative to the observer while a point light source is running symmetrically through the cubes with a speed of $0.9c$. Doppler and searchlight effect were neglected to simplify the scene. The shadows of the cubes appear curved and asymmetrical due to the light travel time effect and the shape of the shadows depends on the perspective. (4000×4000 pixels)

TWO CUBES.MP4 The video shows two cubes on a checkerboard that are at rest relative to the observer while a point light source is running symmetrically through the cubes

with a speed of $0.9c$. Doppler and searchlight effect were are neglected for clarity. The shadows of the cubes appears curved and asymmetrical due to the light travel time effect. (1000×1000 pixels)

TWO CUBES BLINKING.MP4 Two cubes on a checkerboard are at rest relative to the observer while a blinking point light source is running symmetrically through the cubes with a speed of $0.9c$. Doppler and searchlight effect are neglected for clarity. The stripes in front of blinking light source are very thin, which is the exact geometrical analogue to the relativistic Doppler effect (that would shorten the light wavelengths in this area). (1000×1000 pixels)

ROCKET ACCELERATION.MP4 The video shows the view of an accelerated observer in space. The observer would measure an acceleration of $0.1c/1s$. 1000 randomly placed stars (modeled with an equal radius) are modeled with an infinite distance to the observer (very far away). A panel below the observer is added for orientation. All relativistic effects are taken into account. (700×700 pixels)

ROTATION.MP4 The video shows the view of an observer that rotates around one of two cubes at $0.7c$. Two blinking point light sources above the cubes emit monochromatic light (red and blue). Doppler effect and searchlight effect are neglected for clarity. (1000×1000 pixels)

All listed simulations are publicly available on the internet under `https://github.com/JuliusKunze/Fusion-Test/tree/master/Simulationexamples`.

Part III

Discussion

7 Errors

This section will discuss how simulation results could differ from reality due to errors in the used physical model and the program implementation.

Errors in the theory of special relativity as the basis of this project are extremely improbable due to outstanding conformity with results of physical experiments.

There are several known differences from the reality that are caused by the design of the simulation algorithm (that has render results on a computer in a finite time):

- For the raytracing algorithm, a picture has to be divided into pixels. This causes a rasterization effect that does not appear in reality.
- The color display has multiple errors: Monitors, that are based on a tristimulus color format (like RGB), principally can not display all visible colors. In addition, there brightness is limited by a constant bound. Whenever monitors do not meet sRGB 1.10 device characteristics exactly the color will not be exactly equal to the realistic visual impression even in the displayable color range. Additionally, the conversion from a light spectrum to an RGB color uses rasterization which causes color deviations. The size of the deviation depends on resolution of this rasterization.

Errors in the implemented algorithm are improbable due to the process of test driven development. While this technique lowers the probability of errors dramatically, it does not ensure that all cases are covered and correct.

8 Comparison to existing projects

Raytracers based on special relativity existed before this project already:

The raytracer from [Hakenberg] has several disadvantages compared to my project: The quality (resolution, frames per second, colors etc.) of the presented rendering results is relatively poor and there are obvious errors (for example randomly black pixels on surfaces). It does not support any geometric objects except triangles and has no graphical user interface. The source code is hardly extensible. An advantage over my project is the possibility of texture import from pictures.

The raytracer from [Searl] does not support multiple object reference frames that move relative to each other. Again, it has no user interface and poorly extensible code which makes it hard for third party users to use the program.

There are various other relativistic raytracer programs. These programs have similar properties to the two above or they are not able to simulate observed objects moving relative to each other.

The outstanding properties of simulation program of this project are:

- The **correctness** of the algorithms is validated by over 200 unit tests. Quality and realism of the simulation results had a high priority.

- The simulation possibilities are universal within special relativity. The program makes it possible to visualize **surfaces and light sources moving relative to each other** and allows accelerating observers.
- The program is **easily extensible** due to clean code principles.
- The interaction with the program is easy to learn and **user-friendly**.

9 Limitations and extension options

The program could be extended in many directions. Here are the most important opportunities.

- The used algorithm (which is based on the Lorentz transformation of sight rays) disallows simulations of **accelerated observed objects**. Another approach had to be found to enable such simulations.
- The **render time** of pictures and videos could be minimized by optimizing the using algorithm. The program uses raytracing but other rendering algorithms are imaginable, for example polygon rendering, where whole polygons are rendered at once instead of each pixel separately. Those algorithms often render much faster than raytracing but they do not have the universal extensibility that raytracers have.
- There is no possibility to load textures and the program supports simple geometric bodies only. Furthermore, only rendering of diffuse (that means direction-independent) surface materials is possible. The program does not make it possible to simulate dispersion (that means different degrees of refraction depending on the wavelength of light) and slowed light (for example in glass or water). The algorithm could be extended easily to enable relativistic path tracing (for example for soft shadows and extensive light sources). An **extension of the raytracer algorithm** is possible in many directions.
- Human beings have **two eyes** to capture the distance of observed objects. This aspect is not taken into account by the chosen model but could be implemented by doubly rendering videos at slightly different observer positions and merging the results.
- The user interface of the program is in English. **Translations** to other languages would simplify the usage of the application for many users.
- The 3D description language does not support different **physical units** - all physical quantities are expressed in SI base units (meter, second, kelvin, ...) implicitly. This is a source of error because quantities of different dimensions (for example length and time) can be mixed up in equations that do not have a physical sense. Forcing the explicit specification of units would eliminate this type of error.
- The derivation of the effects is based on special relativity. This disallows the simulation of **gravitational visual effects** at high velocities that could be simulated based on the much more complex general relativity only.

Appendix

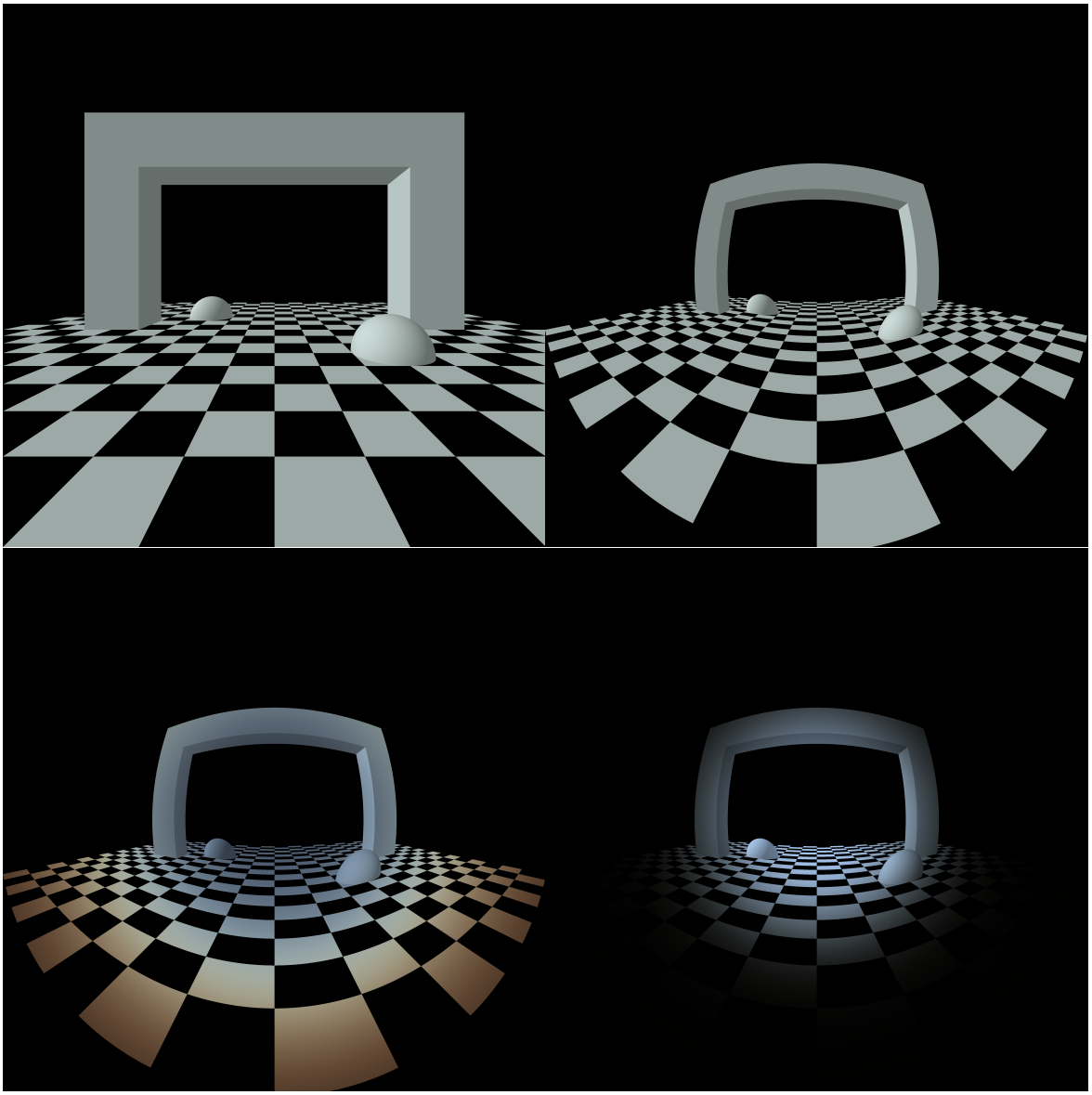

Figure 9.1: The four pictures show the flight through a simple geometric scene at $0.8c$. The scene consists of ideally diffuse white surfaces that are illuminated by sunlight (modeled as a black body spectrum with an effective temperature of $5778K$). The first picture was simulated neglecting all relativistic effects. Aberration effect (picture 2, top right), Doppler effect (picture 3) and searchlight effect (picture 4) are taken into account one by one. (4000×4000 pixels) (All pictures were rendered with the program created by myself.)

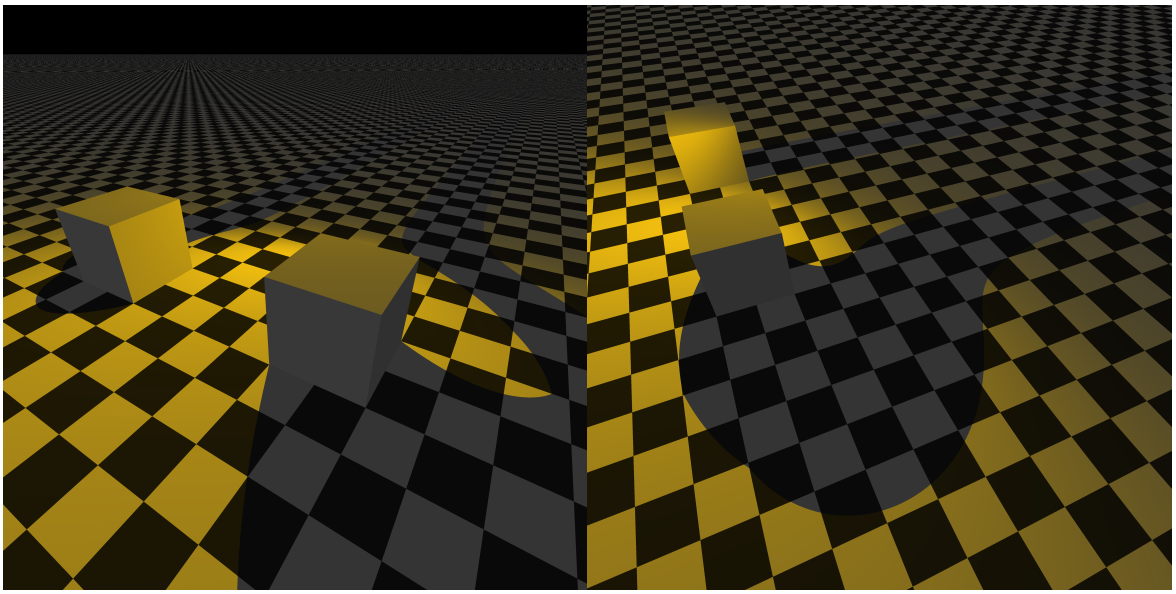

Figure 9.2: The pictures show the same scene at the same time from two different perspectives: Two cubes on a checkerboard that are at rest relative to the observer while a point light source is running symmetrically through the cubes with a speed of $0.9c$. Doppler and searchlight effect were neglected to simplify the scene. The shadows of the cubes appears curved and asymmetrical due to the light travel time effect and the shape of the shadows depends on the perspective. (4000×4000 pixels) (The pictures were rendered with the program created by myself.)

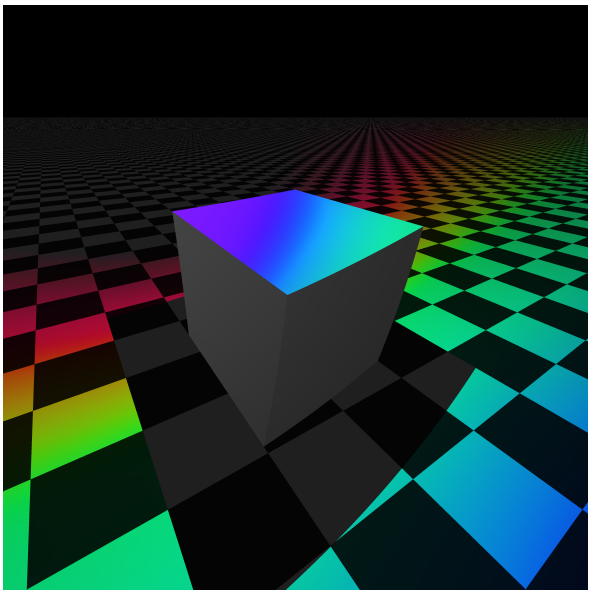

Figure 9.3: The picture shows a cube and a yellow monochromatic point light source flying above a plane at $0.3c$ (towards the bottom left corner of the picture). The light source is fixed in a constant distance above the center of the cube. All surfaces are ideally diffuse and white. Searchlight effect is neglected for clarity. The observer is at rest relative to the plane. (4000×4000 pixels) (The picture was rendered with the program created by myself.)

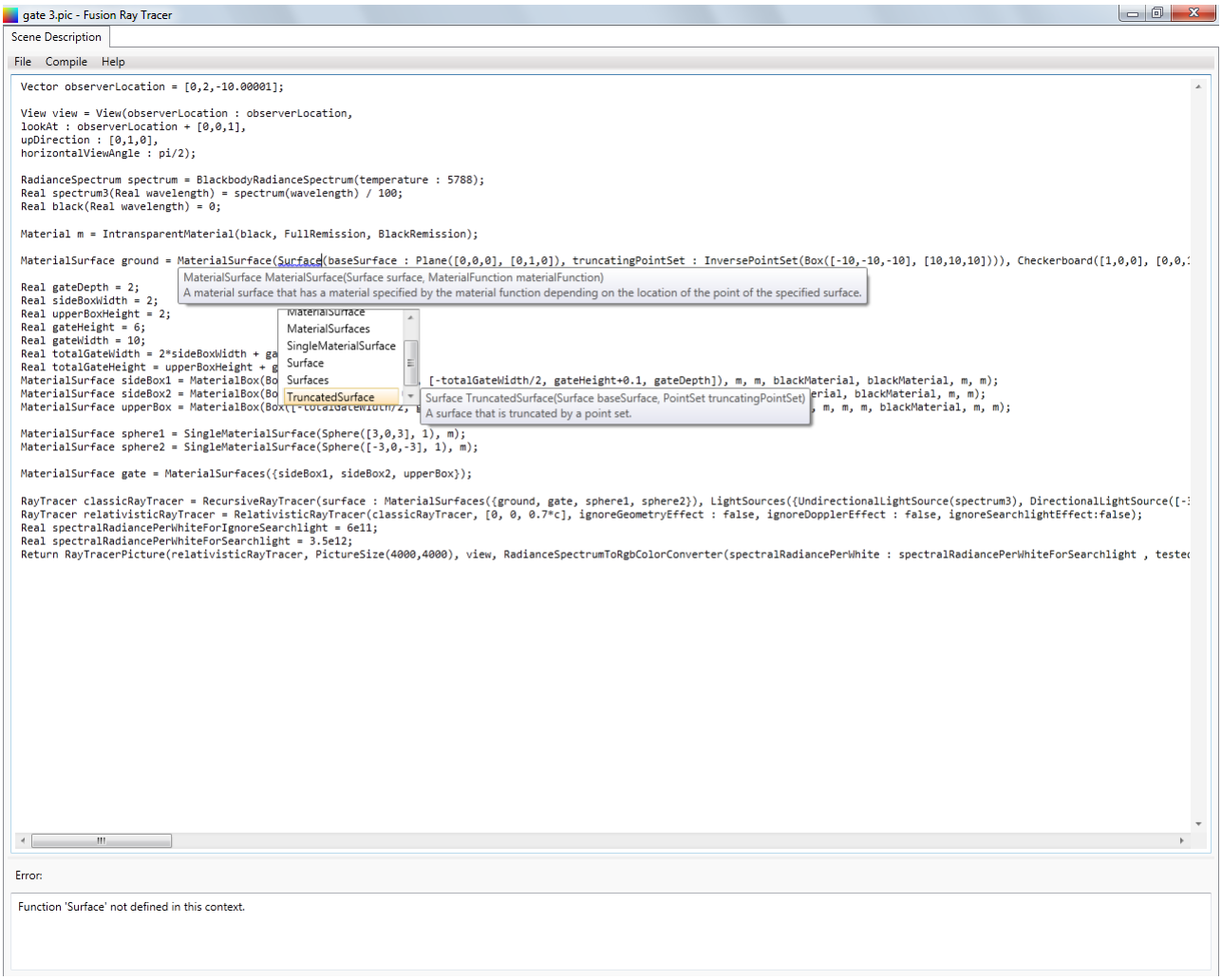

Figure 9.4: The developed raytracing application during the execution. The largest text box in the picture enables input of a scene definition. The program tries to compiler the scene after every change. If an error occurs, a corresponding message is displayed in the error text box below and the error is underlined. Furthermore, up-to-date guidance is implemented (the three small popup boxes in the center of the screenshot). For example, these show the needed parameters for the currently called function, the currently usable (predefined and own) constants, functions and types and corresponding descriptions texts. This assistance is enabled by the interactive compiler and eases the input of the definition dramatically. (The screenshot was created by myself.)

References

[Adelson] Adelson, Edward H. and James R. Bergen. *»The Plenoptic Function and the Elements of Early Vision.«* 1991. Accessed December 22, 2011. `http://web.mit.edu/persci/people/adelson/pub_pdfs/elements91.pdf#search=%22adelson%20plenoptic%20function%20elements%22`.

[Martin] Martin, Robert C. *Clean Code (German Edition)*. Heidelberg, München, Landsberg, Frechen, Hamburg: mitp, 2009.

[Westphal] Westphal, Ralph, and Stefan Lieser. *»Clean Code Developer.«* Accessed December 22, 2012. `http://www.clean-code-developer.de`.

[GeoGebra] Hohenwarter, Markus et al. *»GeoGebra. Free mathematics software for learning and teaching.«* Accessed July 16, 2011. `http://www.geogebra.org`.

[Gibbs] Gibbs, Philip et. al. *»The Relativistic Rocket.«* 1996. Accessed April 9, 2012. `http://math.ucr.edu/home/baez/physics/Relativity/SR/rocket.html`.

[Hakenberg] Hakenberg, Jan P. *»Raytracing Special Relativity.«* Accessed April 15, 2012. `http://www.hakenberg.de/diffgeo/special_relativity.htm`.

[Murray] Murray, Glenn. *»Rotation About an Arbitrary Axis in 3 Dimensions.«* 2011. Accessed April 11, 2012. `http://inside.mines.edu/~gmurray/ArbitraryAxisRotation`.

[Radiance] Wikipedia contributors. *»Radiance.«* Accessed December 19, 2011. `http://en.wikipedia.org/wiki/Radiance`.

[RadiantIntensity] Wikipedia contributors. *»Radiant intensity.«* Accessed December 19, 2011. `http://en.wikipedia.org/wiki/Radiant_intensity`.

[RadiantPower] Wikipedia contributors. *»Radiant flux.«* Accessed December 19, 2011. `http://en.wikipedia.org/wiki/Radiant_flux`.

[Raytracing] Wikipedia contributors. *»Raytracer.«* Accessed December 19, 2011. `http://de.wikipedia.org/wiki/Raytracer`.

[Searl] Searl, Antony. *»The Relativistic Raytracer.«* Accessed April 15, 2012. `http://www.anu.edu.au/physics/Searle/Obsolete/Raytracer.html`.

[Tipler] Tipler, Paul A. *Physik*. Heidelberg, Berlin, Oxford: Spektrum Akademischer Verlag, 1995.

[Weißkopf] Weißkopf, Daniel, Ute Kraus, and Hanns Ruder. *»Searchlight and Doppler Effects in the Visualization of Special Relativity: A Corrected Derivation of the Transformation of Radiance.«* 1995. Accessed December 19, 2011. `http://dl.acm.org/citation.cfm?id=336459`.