

A Texture Mapping Approach for the Visualization of Special Relativity

Daniel Weiskopf*

Theoretical Astrophysics[†]
University of Tübingen

Abstract

This paper describes a novel rendering technique for the special relativistic visualization of the apparent geometry of fast moving objects. This method is based on the relativistic aberration of light and makes use of texture mapping. Interactive frame rates are possible, since texture mapping is supported by modern graphics hardware. Unlike the well-known relativistic polygon rendering approach, the texture-based relativistic rendering allows for a physically correct treatment of the geometric quantities that effect the calculation of illumination. Furthermore, no artifacts due to relativistic transformations are introduced.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture I.3.8 [Computer Graphics]: Applications—Special Relativity J.2 [Physical Sciences and Engineering]: Theoretical Astrophysics

Keywords: special relativity, visualization, texture mapping

1 Introduction

Einstein’s Theory of Special Relativity is widely regarded as a difficult and hardly comprehensible theory. One important reason for this is that the properties of space, time, and light in relativistic physics are totally different from those in classical, Newtonian physics. In many respects, they are contrary to human experience and everyday perception, which is based on low velocities.

Mankind is limited to very low velocities compared to the speed of light. Therefore, computer simulations are the only means of visually exploring the realm of special relativity and can thus help the intuition of physicists.

The visual appearance of rapidly moving objects shows intriguing effects of special relativity. Apart from a previously disregarded article by Lampa [8] in 1924 about the invisibility of the Lorentz contraction, the first solutions to this problem were given by Penrose [12] and Terrell [17] in 1959. Various aspects were discussed by Weiskopf [19], Boas [1], Scott and Viner [16], and Scott and van Driel [15].

Hsiung and Dunn [3] were the first to use advanced visualization techniques for image shading of fast moving objects. They propose an extension of normal three-dimensional ray tracing. Hsiung et al. [4, 5] add the visualization of the Doppler effect. Hsiung et al. [6] and Gekelman et al. [2] describe a polygon rendering approach which is based on the apparent shapes of objects as seen by a relativistic observer. Relativistic polygon rendering extends the normal rendering pipeline by an additional transformation of the vertices and makes use of modern computer graphics hardware. Polygon rendering is also used as a basis for a virtual environment for special relativity [13, 18].

In this paper, a novel technique for special relativistic rendering is presented. This technique is based on an effect of special relativ-

ity which is called relativistic aberration of light. Unlike the relativistic polygon rendering approach, this method does not transform the coordinates of the vertices, but transforms the images which are rendered in the normal non-relativistic way. Therefore, it leaves the rendering pipeline unaltered up to the very last step. The relativistic transformations are performed on the image plane by texture mapping, which allows interactive frame rates on modern computer graphics hardware.

The next section briefly describes the physical background needed for the rendering algorithm. In Section 3, the rendering algorithm is presented. Section 4 gives details of the implementation and shows results. Section 5 compares the new rendering technique of this paper with the well-known polygon rendering technique. The paper ends with a conclusion and an outlook on future work.

2 Lorentz Transformation and Aberration of Light

This section describes only those terms and equations of special relativity which are relevant for rendering. A detailed presentation of special relativity can be found in [10, 11, 14]. The key term used from relativistic physics is the so-called relativistic aberration of light. The relativistic aberration of light causes a rotation of the direction of light when one is changing from one inertial frame of reference to another. The aberration of light is sufficient to fully describe the apparent geometry seen by a fast moving camera.

Let us consider two inertial frames of reference, S and S' , with S' moving with velocity v along the z axis of S . The usual Lorentz transformation along the z axis connects frames S and S' .

In reference frame S , consider a light ray whose direction is described with spherical coordinates θ and ϕ , as shown in Fig. 1. In frame S' , the direction of the light ray is described by θ' and ϕ' . The expressions for the relativistic aberration of light connect these two representations, cf. [9, 11]:

$$\cos \theta' = \frac{\cos \theta - \beta}{1 - \beta \cos \theta}, \quad (1)$$

$$\phi' = \phi, \quad (2)$$

where $\beta = v/c$ and c is the speed of light.

3 Texture-Based Relativistic Rendering

The physical basis for texture-based relativistic rendering is the Lorentz transformation of properties of light. Suppose that the photon field is known at one point in spacetime, i.e. at a point in three-dimensional space and at an arbitrary but fixed time. This photon field is measured in one frame of reference which is denoted S_{obj} . Now consider an observer, i.e. a camera, being at this point in spacetime. The observer is not at rest relative to S_{obj} , but is moving at arbitrary speed relative to S_{obj} . However, the observer is at rest in another frame of reference, S_{observer} . The photon field can then be calculated with respect to S_{observer} by the Lorentz transformation

*weiskopf@tat.physik.uni-tuebingen.de

[†]Theoretical Astrophysics, University of Tübingen, Auf der Morgenstelle 10, D-72076 Tübingen, Germany

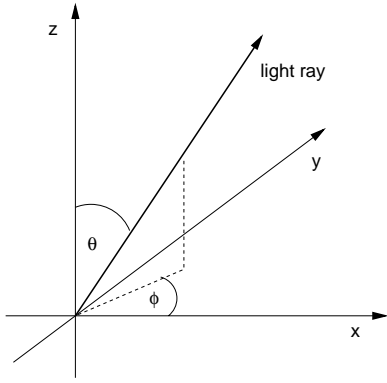


Figure 1: A light ray in spherical coordinates.

from S_{obj} to $S_{observer}$. Finally, the transformed photon field is used to generate the picture taken by the observer's camera.

Now we restrict ourselves to static scenes, i.e. all scene objects and light sources are at rest relative to each other and relative to S_{obj} . Here, all relevant information about the photon field—namely direction, color, and intensity of the incoming photons—can be determined by standard computer graphics algorithms, since the finite speed of light can be neglected in all calculations for the static situation. With this information and with the use of the equations for the relativistic aberration of light, the picture seen by the relativistic observer can be generated. In this paper, only the relativistic effects on the apparent geometry are taken into account. Changes in color and brightness due to the Doppler and searchlight effects are ignored.

How can this idea of the Lorentz transformation of the photon field be efficiently implemented? First, consider the situation for another observer that is at rest in S_{obj} . The information about the photon field is stored in the form of images projected onto a unit sphere with the observer being at the origin of the sphere. The projection of one image is illustrated in Fig. 2. The covering of the whole sphere is accomplished by projecting several images which are taken with differing orientations of the camera. A similar method is used for reflection and environment mapping. The direction of an incoming light ray is determined by the coordinates (θ, ϕ) of the corresponding point on the sphere.

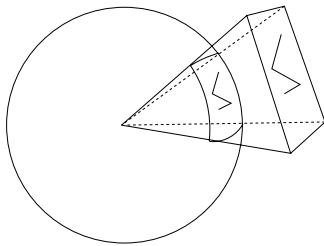


Figure 2: Projection of a rendered image onto a sphere.

Now consider the same situation with the moving observer. The photon field has to be changed by the Lorentz transformation from S_{obj} to $S_{observer}$. This yields a transformation of (θ, ϕ) to (θ', ϕ') according to Eqs. (1) and (2) for the relativistic aberration of light. The resulting distortion of the image mapped onto the sphere is illustrated in Fig. 3.

Usually, the direction of motion is not identical to the z axis. Therefore, additional rotations of the coordinate system have to be considered. The complete mapping of the coordinates of a point in the original image to the spherical coordinates of the corresponding

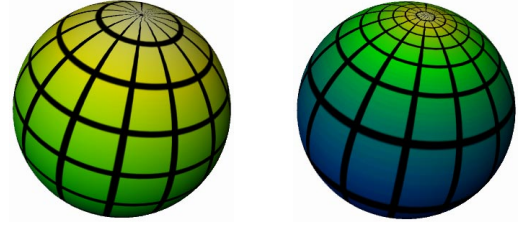


Figure 3: Effect of the relativistic aberration of light on the texture mapped onto the unit sphere. The left sphere shows the mapping without distortion, the right sphere illustrates the distortion for 90 percent of the speed of light. The direction of motion shows towards the northpole of the sphere.

point seen by the moving observer is

$$T_{\text{mapping}} = T_{\text{rot, b}} \circ T_{\text{aberration}} \circ T_{\text{rot, a}} \circ T_{\text{proj}},$$

with

$$\begin{aligned} T_{\text{proj}} &: [0, 1] \times [0, 1] \longrightarrow S^2, & (u, v) &\longmapsto (\theta, \phi), \\ T_{\text{rot, a}} &: S^2 \longrightarrow S^2, & (\theta, \phi) &\longmapsto (\theta_{\parallel}, \phi_{\parallel}), \\ T_{\text{aberration}} &: S^2 \longrightarrow S^2, & (\theta_{\parallel}, \phi_{\parallel}) &\longmapsto (\theta'_{\parallel}, \phi'_{\parallel}), \\ T_{\text{rot, b}} &: S^2 \longrightarrow S^2, & (\theta'_{\parallel}, \phi'_{\parallel}) &\longmapsto (\theta', \phi'). \end{aligned}$$

S^2 is the unit sphere. The map T_{proj} projects the rendered image onto the sphere and determines the corresponding coordinates on the sphere. The coordinates of the pixels are in the interval $[0, 1]$. The rotation $T_{\text{rot, a}}$ takes into account that the direction of motion differs from the z axis of the sphere. The actual relativistic transformation is accomplished by $T_{\text{aberration}}$ with the use of Eqs. (1) and (2). Finally, $T_{\text{rot, b}}$ describes the rotation of the observer's coordinate system relative to the direction of motion.

The relativistic rendering process has to generate the texture coordinates for the mapping of the non-relativistic images onto the unit sphere which surrounds the moving observer. With the inverse map T_{mapping}^{-1} , the texture coordinates can be calculated from the spherical coordinates θ' and ϕ' in the coordinate system of the observer.

Fig. 4 shows the structure of the relativistic rendering process. This process is very similar to the one used for the implementation of reflection or environment mapping onto a sphere. The main difference lies in the generation of the texture coordinates.

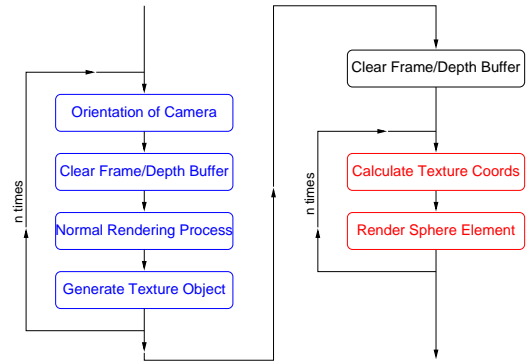


Figure 4: Structure of the relativistic rendering process.

In the first part marked blue, the textures for a spherical mapping are generated. Here, the normal non-relativistic rendering process is

called n times; n is the number of textures mapped onto the sphere and depends on the viewing angle and the orientation that are used for the rendering of the texture images.

In the second part marked red, the texture coordinates are calculated with the use of T_{mapping}^{-1} . Then the textured sphere is actually drawn. The relativistic transformation is completely absorbed into the calculation of the texture coordinates. For the generation of the final image a picture is taken from inside the sphere. Note that the viewpoint has to be at the midpoint of the sphere, whereas the orientation is not restricted and allows for viewing into arbitrary directions.

4 Implementation and Results

The relativistic rendering algorithm is implemented in C++ and runs on top of a normal OpenGL renderer. OpenGL version 1.1 [20, 7] is used.

The structure of the relativistic rendering process is illustrated in Fig. 4. The generation of the non-relativistic texture images (blue boxes in the figure) runs in the back buffer in order to hide it from the user. The OpenGL command `glCopyTexImage2D` transfers the results of non-relativistic rendering from the frame buffer to texture memory. Texture objects (`glBindTexture`) allow fast access to the stored textures. In the current implementation, a constant number of six textures is used to cover the whole sphere. The texture coordinates (red box) are only recalculated if the velocity has changed.

An example of relativistic rendering can be found in Fig. 6. Fig. 5 shows the same scene in the non-relativistic limit. The position of the camera is identical in both pictures.



Figure 5: Non-relativistic view of the box-shaped test scene.

Performance measurements can be found in Table 1. They show that two factors play a dominant role for the rendering speed. The first is the time for the rendering of the normal non-relativistic images which are used as textures. This process usually is fill rate limited. The second is the time for transferring the non-relativistic images from the frame buffer to texture memory.

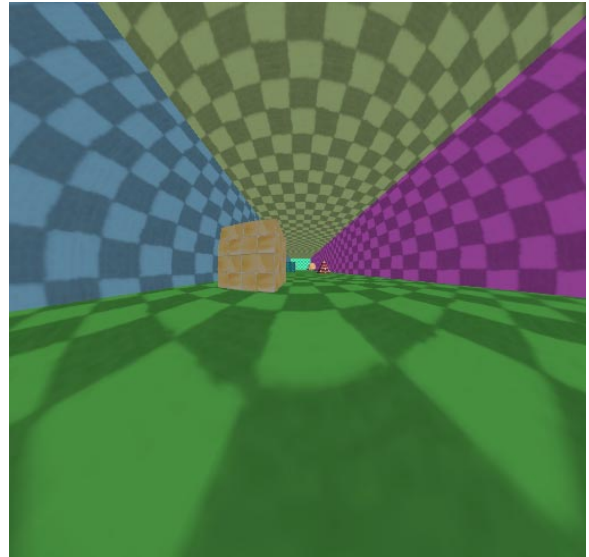


Figure 6: Special relativistic rendering. The observer is rushing into the scene with 90 percent of the speed of light.

5 Discussion

In this section, texture-based relativistic rendering is compared to relativistic polygon rendering.

Relativistic polygon rendering extends the normal rendering pipeline by a relativistic transformation of the coordinates of the vertices. Since this transformation is non-linear, artifacts are introduced by the linear connection between the transformed vertices through straight edges. These artifacts can be reduced by a fine remeshing of the original meshes in a preprocessing step or by an adaptive subdivision scheme during runtime. All this is not needed for the texture approach, since the relativistic transformation is rather performed at the end of the rendering pipeline and effects every pixel in the image plane. Therefore, the texture-based relativistic rendering does not need any modifications of the scene objects or the core rendering method. It does not increase the number of triangles to be rendered, it has no extra computational cost per vertex, and it does not introduce any relativistic artifacts.

Rendering performance depends on different factors for relativistic polygon rendering and for texture-based relativistic rendering. The frame rate for relativistic polygon rendering is usually limited by the floating point performance of the processor due to the transformation of the vertices and by the polygon rate of the graphics board due to a higher number of polygons. The frame rate for the texture approach is limited by the pixel fill rate and by the bandwidth between frame buffer and texture memory. Since increased pixel fill rate is important for other computer graphics applications such as volume rendering, there will be a continuous effort of hardware developers to achieve higher pixel fill rates.

Another important issue is a physically correct handling of illumination. The reflection on a surface is usually described in terms of the angle between the light vector and the surface normal and of the angle between the viewing vector and the surface normal. Relativistic polygon rendering transforms the coordinates of all vertices. Therefore, both angles are changed, which, in general, results in a wrong calculation of illumination. However, relativistic polygon rendering can handle two special cases: ambient lighting, and diffuse (Lambertian) materials illuminated by directional lights (infinitely far light sources). The first case is trivial, since there is no angle dependency at all. For the second case, the illumination is

image size	Onyx2		Visual WS	
	512 ²	1024 ²	512 ²	1024 ²
frame rate	7.1 fps	4.9 fps	10.0 fps	5.9 fps
time portion for:				
normal rendering process	50%	38%	59%	52%
transfer frame/tex buffer	21%	41%	11%	26%
others	29%	21%	30%	22%

Table 1: Rendering performance on an SGI Onyx2 system with InfiniteReality2 graphics board and MIPS R10000/195MHz processor, and on an SGI Visual Workstation 320 with Cobalt graphics board and Intel Pentium III/500 MHz processor. The time portions for the non-relativistic rendering process and for the data transfer from frame buffer to texture memory are given. The image and texture sizes are 512*512 or 1024*1024 pixels, respectively. The test scene is shown in Figs. 5 and 6.

independent of the angle between the viewing vector and the surface normal, and the angle between the light vector and the surface normal can be correctly calculated if the relativistic transformation changes only the coordinates of the vertex, but not the surface normal.

The correct handling of the geometric quantities needed for the computation of illumination is an important advantage of texture-based relativistic rendering. Let us assume that the non-relativistic image is correctly generated by a camera which is at rest relative to the scene objects and the light sources. Then, apart from the influence of the Doppler and searchlight effects, the relativistic image will be correct as well because the Lorentz transformation in the form of the aberration formula allows for the complete transformation of the relevant physical properties. Therefore, more realistic illumination algorithms can be used, such as Phong shading or mirror effects. In future work, changes in color and brightness due to the Doppler and searchlight effects will also be included.

One problem with texture-based relativistic rendering arises due to the properties of the aberration of light. The aberration equation does not conserve the element of solid angle. Therefore, the relativistic mapping does not conserve the area of an element on the sphere. The image is compressed towards the direction of motion, whereas the image gets magnified towards the back. This magnification can reveal a lower resolution of the texture. This issue could be solved by adapting the texture resolution to the relativistic distortion.

6 Conclusion and Future Work

In this paper a texture-based approach to special relativistic rendering has been introduced. The physical basis is the relativistic aberration of light. The algorithm uses texture mapping in order to transform the non-relativistic image into the frame of reference of a fast moving observer. Texture-based relativistic rendering allows for the fast visualization of the apparent geometry of objects which are at rest relative to each other.

There are several advantages compared to relativistic polygon rendering. There are no artifacts due to straight lines between vertices. There is no need for additional polygons and for the computation of the transformation of vertices. Most importantly, a physically correct model for the calculation of the geometric quantities effecting illumination is implemented.

In future work, the Doppler and searchlight effects will be considered. Furthermore, the issue of rendering performance and image quality will be addressed. A limiting part in the rendering process is the pixel fill rate for the generation of the images for the textures. The number of textures can be reduced if only that part of the sphere which is actually viewed by the relativistic observer is covered by textures. In addition, the resolution of the textures could be adapted to the magnification by the aberration of light. This will increase rendering speed and enhance image quality.

Acknowledgements

Special thanks to Bettina Salzer for proof-reading. This work was supported by the Deutsche Forschungsgemeinschaft (DFG) and is part of the project D4 within the Sonderforschungsbereich 382.

References

- [1] M. L. Boas. Apparent shape of large objects at relativistic speeds. *American Journal of Physics*, 29(5):283–286, May 1961.
- [2] W. Gekelman, J. Maggs, and L. Xu. Real-time relativity. *Computers in Physics*, 5(4):372–385, July/Aug. 1991.
- [3] P.-K. Hsiung and R. H. P. Dunn. Visualizing relativistic effects in spacetime. In *Proceedings of Supercomputing '89 Conference*, pages 597–606, 1989.
- [4] P.-K. Hsiung and R. H. Thibadeau. Spacetime visualization of relativistic effects. In *Proceedings of the 1990 ACM Eighteenth Annual Computer Science Conference*, pages 236–43, Washington, DC, Feb. 1990.
- [5] P.-K. Hsiung, R. H. Thibadeau, C. B. Cox, R. H. P. Dunn, M. Wu, and P. A. Olbrich. Wide-band relativistic doppler effect visualization. In *Proceedings of the Visualization 90 Conference*, pages 83–92, Oct. 1990.
- [6] P.-K. Hsiung, R. H. Thibadeau, and M. Wu. T-buffer: Fast visualization of relativistic effects in spacetime. *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, 24(2):83–88, Mar. 1990.
- [7] R. Kempf, C. Frazier, and OpenGL Architecture Review Board. *OpenGL reference manual: the official reference document to OpenGL, version 1.1*. Addison-Wesley Developers Press, Reading, MA, USA, second edition, 1997.
- [8] A. Lampa. Wie erscheint nach der Relativitätstheorie ein bewegter Stab einem ruhenden Beobachter? *Zeitschrift für Physik*, 27:138–148, 1924. In German.
- [9] J. M. McKinley. Relativistic transformations of light power. *American Journal of Physics*, 47(7):602–605, July 1979.
- [10] C. W. Misner, K. S. Thorne, and J. A. Wheeler. *Gravitation*. Freeman and Company, New York, 1973.
- [11] C. Møller. *The Theory of Relativity*. Clarendon Press, Oxford, second edition, 1972.
- [12] R. Penrose. The apparent shape of a relativistically moving sphere. *Proceedings of the Cambridge Philosophical Society*, 55:137–139, 1959.
- [13] R. T. Rau, D. Weiskopf, and H. Ruder. Special relativity in virtual reality. In H.-C. Hege and K. Polthier, editors, *Mathematical Visualization*, pages 269–279. Springer Verlag, Heidelberg, 1998.
- [14] W. Rindler. *Introduction to Special Relativity*. Clarendon Press, Oxford, second edition, 1991.
- [15] G. D. Scott and H. J. van Driel. Geometrical appearances at relativistic speeds. *American Journal of Physics*, 38(8):971–977, Aug. 1970.
- [16] G. D. Scott and R. R. Viner. The geometrical appearance of large objects moving at relativistic speeds. *American Journal of Physics*, 33(7):534–536, July 1965.
- [17] J. Terrell. Invisibility of the Lorentz contraction. *Physical Review*, 116(4):1041–1045, Nov. 1959.
- [18] D. Weiskopf. An immersive virtual environment for special relativity. Report Nr. 108, SFB 382, University of Tübingen, Germany, Jan. 1999. http://www.uni-tuebingen.de/uni/opx/reports/weiskopf_108.ps.gz.
- [19] V. F. Weisskopf. The visual appearance of rapidly moving objects. *Physics Today*, 13(9):24–27, 1960.
- [20] M. Woo, J. Neider, T. Davis, and OpenGL Architecture Review Board. *OpenGL programming guide: the official guide to learning OpenGL, version 1.1*. Addison-Wesley, Reading, MA, USA, 1997.