Special Relativistic Visualization by Local Ray Tracing

Article *in* IEEE Transactions on Visualization and Computer Graphics · January 2011

DOI: 10.1109/TVCG.2010.196 · Source: PubMed

3 authors, including:

Thomas Müller
Haus der Astronomie, Heidelberg, Germany
81 PUBLICATIONS **482** CITATIONS
SEE PROFILE

Sebastian Grottel
Technische Universität Dresden
29 PUBLICATIONS **428** CITATIONS
SEE PROFILE

Special Relativistic Visualization by Local Ray Tracing

Thomas Müller, Sebastian Grottel, and Daniel Weiskopf, *Member, IEEE Computer Society*

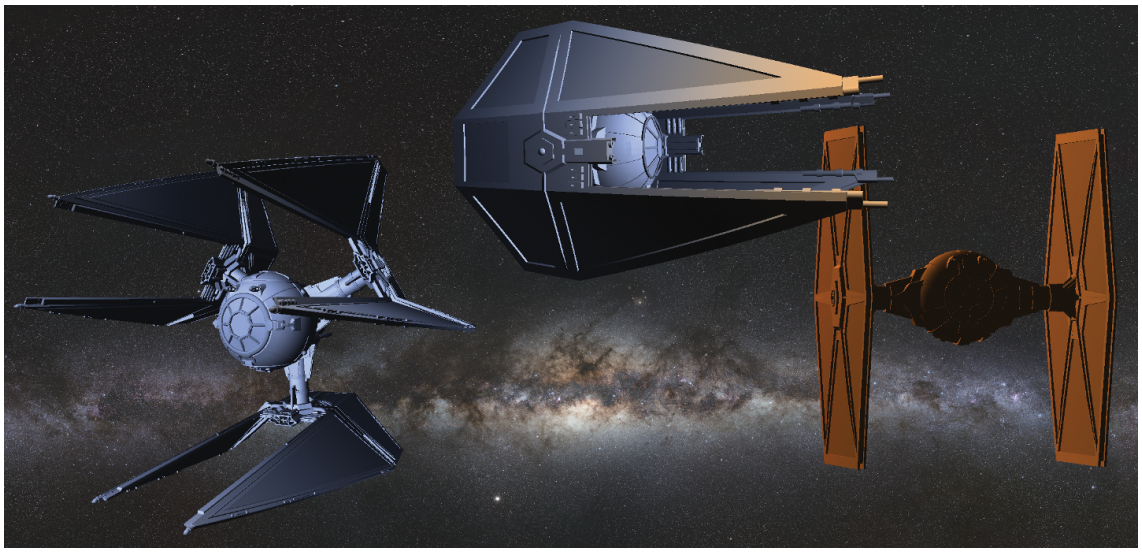

Fig. 1. Three vessels move in different directions at near light speed. Most dominant is the redshift and the magnification of the receding right vessel. Background Milky Way from ESO [1]

Abstract— Special relativistic visualization offers the possibility of experiencing the optical effects of traveling near the speed of light, including apparent geometric distortions as well as Doppler and searchlight effects. Early high-quality computer graphics images of relativistic scenes were created using offline, computationally expensive CPU-side 4D ray tracing. Alternate approaches such as image-based rendering and polygon-distortion methods are able to achieve interactivity, but exhibit inferior visual quality due to sampling artifacts. In this paper, we introduce a hybrid rendering technique based on polygon distortion and local ray tracing that facilitates interactive high-quality visualization of multiple objects moving at relativistic speeds in arbitrary directions. The method starts by calculating tight image-space footprints for the apparent triangles of the 3D scene objects. The final image is generated using a single image-space ray tracing step incorporating Doppler and searchlight effects. Our implementation uses GPU shader programming and hardware texture filtering to achieve high rendering speed.

Index Terms—Poincaré transformation, aberration of light, Doppler effect, illumination, searchlight effect, special relativity, GPU ray tracing

1 Introduction

Special Relativity describes space and time not as two separate qualities but as one entity: spacetime. In daily life, however, we do not see this unity, because the peculiarities of Special Relativity only occur for high relative velocities close to the speed of light. Unfortunately, even with today's most advanced technology, we are far from reaching relativistic speeds. Hence, the only way to experience relativistic effects is by computer simulation. The most puzzling effects predicted by Special Relativity are time dilation and length contraction. Roughly speaking, moving clocks appear to run more slowly and if we could measure a moving rod, it would be length-contracted in the direction of motion. Furthermore, due to aberration, an observer at high velocity would see an object under a different angle than it would appear if they were at rest. Additionally, light undergoes a Doppler shift of wavelengths. If an object and an observer approach each other, the searchlight effect causes the light intensity to increase in the common direction. With higher observation angles, however, light intensity decreases so much that the observer will see nearly nothing.

Visualizing relativistic effects has a long history. Probably, Lampa [14] in 1924 was the first who correctly described how a fast moving object would appear to an observer. But his work was not recognized at that time. In 1939, Gamow claimed in his book "Mr. Tompkins in Wonderland" [6] that length contraction could actually be observed. Even Einstein overlooked the subtle difference between measuring and seeing. But this difference is fundamental because *measuring* is done simultaneously at the object (with respect to the observer's reference frame), whereas *seeing* is an operation simultaneous in the eye. The latter one is what we call relativistic rendering. Correct descriptions of observational relativity appeared in 1959, when Penrose [16] showed that a sphere always has a circular silhouette irrespective of its relative velocity with respect to the observer. In the same year, Terrell [25] demonstrated that a moving rod, while it is length-contracted in the direction of motion, appears rotated. In the following years, several articles described relativistic effects at great length, e.g. [30, 23, 22]. However, all of them lack the possibility to interactively study the effects.

• *The authors are with Visualization Research Center (VISUS), University of Stuttgart, Allmandring 19, 70569 Stuttgart, Germany, E-mail: {Thomas.Mueller, Sebastian.Grottel, Daniel.Weiskopf}@visus.uni-stuttgart.de.*

Manuscript received 31 March 2010; accepted 1 August 2010; posted online 24 October 2010; mailed on 16 October 2010.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

In 1989 [2], Taylor [24] developed an early exploratory tool for educational purpose. He demonstrated that interactive computer applications help students to comprehend the peculiarities of Special Relativity. A more up-to-date, game-like simulation of Special Relativity was developed by Savage et al. [21], where the user can freely move in a static, artificial environment. They also conducted an extensive evaluation of their interactive simulation and showed the effectiveness for teaching [20]. Weiskopf et al. [27] detailed that relativistic visualization is specifically suited for popular science magazines, film contributions to TV shows, or interactive museum installations.

2 Previous work

Special relativistic visualization can be grouped into two categories. On the one hand, illustrative visualizations like Minkowski diagrams can be used to depict e.g. the causal structure of spacetime. On the other hand, rendering of nearly realistic scenarios offers the possibility to experience all visual effects from a first-person point of view, which is the objective of our work. To implement this, there are mainly three conceptually different methods: polygon rendering, ray tracing, and image-space methods.

Early publications on relativistic polygon rendering are due to Hsiung et al. [12] and Gekelman [8]. Hsiung et. al. [12] introduced the so called time-buffer for fast visualization, which was inspired by, and implemented through, the z-buffer. Relativistic polygon rendering is based on the apparent shapes of objects as seen by a relativistic observer.

Unfortunately, polygon rendering is only an approximation and visual artifacts may appear. Ray tracing is a method to guarantee optimal visual quality. Special relativistic ray tracing was extensively studied by Hsiung and Dunn [10] and Hsiung and Thibadeau [11]. They extended standard 3D ray tracing to the 4D spacetime respecting the finite speed of light and the Lorentz transformation of light rays. An improved spacetime ray tracing system that includes reflection and transmission phenomena was presented by Li et al. [15].

Image-based methods are a fast alternative to polygon rendering or ray tracing for restricted scenarios where a fast moving observer travels through a static environment. These methods, like in Weiskopf et. al. [28, 27] and Savage et. al. [21], however, are subject to visual artifacts due to limited image-space resolution.

All three methods can be extended to include illumination models as realized e.g. by Chang et al. [5], Betts [4], or Weiskopf et al. [29].

Our hybrid approach takes advantage of polygon rendering and ray tracing, combined for high image quality and speed. Based on a proxy geometry created by adopting the polygon rendering approach, we perform local ray tracing to interactively visualize multiple objects in arbitrary relative motion, thus being more flexible than image-based methods. The strategy of local GPU ray tracing has been successfully used in various applications of interactive computer graphics. For example, Gumhold [9] presented point-based ray tracing of depth-corrected quadratic surfaces (e.g. spheres, cones, cylinders, and ellipsoids). The approach was further extended to render complex glyphs built from implicit quadratic polynomial surfaces, specialized for different application domains, e.g. by Reina and Ertl [18]. Gascuel el. at. [7] used local ray tracing to calculate non-linear projections for shadow maps or environment maps on the GPU.

3 Physical fundamentals

The physical fundamentals of Special Relativity are discussed at great length in the standard literature, see e.g. Rindler [19]. To be self-consistent and to clarify our notation, we give a short introduction to the Poincaré and Lorentz transformations between two reference systems in relative motion. We also briefly review special relativistic polygon rendering and the spectrum-to-color conversion considering Doppler shift and searchlight effect (see Weiskopf [26] for details).

3.1 Poincaré transformation

A standard situation of Special Relativity is a reference system S' moving with velocity $\vec{v}$ relative to the reference system S as shown in Fig. 2.

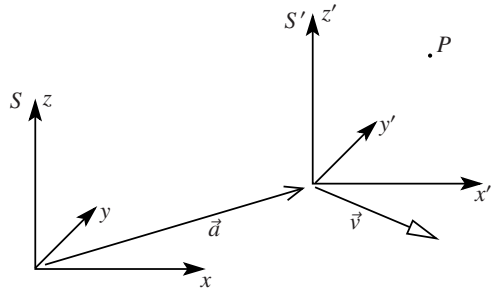

Fig. 2. System S' moves with velocity $\vec{v}$ relative to S. With respect to S, the origin of S' is located at spatial position $\vec{a}$ at $t = t' = 0$.

The axes of both systems are aligned parallel. Here and in the following, primed coordinates belong to S' and non-primed coordinates belong to S. With respect to S, the origin of S' is located at spatial position $\vec{a}$ at $t = t' = 0$. A point P in spacetime can now be described either with coordinates $\mathbf{x} = (ct, x, y, z) = (ct, \vec{x})$ of S or with coordinates $\mathbf{x}' = (ct', x', y', z') = (ct', \vec{x}')$ of S'. Time is multiplied by the speed of light, c, so that all coordinates have the same units. Physically speaking, P is called an event because it is described by spacetime coordinates. Note that we use boldface symbols for events as well as for four-vectors.

Both coordinate representations of P are related by the Poincaré transformation

$$\mathbf{x} = \Lambda_{\vec{\beta}}\mathbf{x}' + \mathbf{a}, \qquad \mathbf{x}' = \Lambda_{-\vec{\beta}}(\mathbf{x} - \mathbf{a}) \tag{1}$$

with $\mathbf{a} = (0, \vec{a})$ and the Lorentz matrix

$$\Lambda_{\vec{\beta}} = \begin{pmatrix} \gamma & \gamma\vec{\beta}^T \\ \gamma\vec{\beta} & \mathbb{1}_3 + \frac{\gamma^2}{\gamma+1}\vec{\beta}\vec{\beta}^T \end{pmatrix}. \tag{2}$$

Here, $\mathbb{1}_3$ describes the identity matrix in three dimensions, $\vec{\beta} = \vec{v}/c$ represents the velocity $\vec{v}$ relative to the speed of light, and $\gamma = 1/\sqrt{1 - \vec{\beta}\cdot\vec{\beta}}$ is the gamma factor.

To follow the path of light transport, which is the basis for ray tracing and local lighting algorithms, we need the description of light rays. Such a light ray is described by the four-vector $\mathbf{k} = \omega\left(1, k_x, k_y, k_z\right) = \omega(1, \vec{k})$ with the normalized ray direction $\|\vec{k}\| = 1$. The circular frequency is $\omega = 2\pi c/\lambda$, where λ is the wavelength of the light. Four-vectors are Lorentz-transformed via

$$\mathbf{k} = \Lambda_{\vec{\beta}}\mathbf{k}', \qquad \mathbf{k}' = \Lambda_{-\vec{\beta}}\mathbf{k}, \tag{3}$$

which can be split into a timelike and a spacelike part:

$$\omega = \omega'\gamma\left(1 + \vec{\beta}\cdot\vec{k}'\right), \quad \omega\vec{k} = \omega'\left[\gamma\vec{\beta} + \vec{k}' + \frac{\gamma^2}{\gamma+1}\left(\vec{\beta}\cdot\vec{k}'\right)\vec{\beta}\right]. \tag{4}$$

The timelike part determines the Doppler factor $D = \omega/\omega'$ and, thus, the frequency shift between both reference systems. The spacelike part, on the other hand, describes the aberration effect.

3.2 Special relativistic polygon rendering

The idea of polygon-based methods for special relativistic visualization is to transform the polygons of an object from its proper reference system into the reference frame of the observer. We use a similar method to approximate the image-space footprints as will be described in Sect. 4.1. Unfortunately, this transformation is non-linear. Effects like the distortion of the mesh, Doppler shift, and searchlight effect are possible with this approach but image quality depends on the resolution of the polygon mesh.

The object-space transformation: $\mathscr{R} : \mathbb{R}^3 \to \mathbb{R}^3, \vec{x}' \mapsto \vec{x}$ works as follows. First, a Poincaré transformation yields the position of the observer within the moving object system S' at observation time t_{obs}:

$$\mathbf{x}'_{\text{obs}} = \Lambda_{-\vec{\beta}}(\mathbf{x}_{\text{obs}} - \mathbf{a}). \tag{5}$$

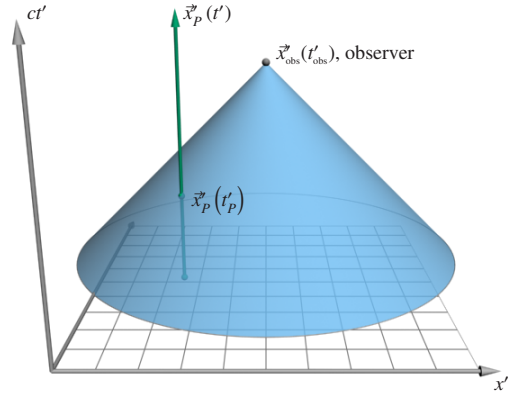

Fig. 3. The coordinate system represents space x' and time ct'. The worldline $\vec{x}'_P(t')$ is the position of the point P at rest. The intersection between the worldline $\vec{x}'_P(t')$ and the backward light cone of the observer with current position $\vec{x}'_{\text{obs}}$ yields the time t'_P, where light must be emitted by P to reach the observer at t'_{obs}.

Because of the finite speed of light, the observer can see only objects that lie on the backward light cone defined by the equation (see also Fig. 3)

$$-\left(ct'-ct'_{\text{obs}}\right)^2+\left(x'-x'_{\text{obs}}\right)^2+\left(y'-y'_{\text{obs}}\right)^2+\left(z'-z'_{\text{obs}}\right)^2=0. \quad (6)$$

Now, the intersection between the worldline $\vec{x}'_P(t')$ of the point P and the backward light cone of the observer yields the time

$$ct'_P = ct'_{\text{obs}} - \|\vec{x}'_P - \vec{x}'_{\text{obs}}\|, \quad (7)$$

where light must be emitted by the point P to reach the observer at observation time and thus yields $\mathbf{x}'_P$. The apparent position of P with respect to the observer's reference frame thus reads

$$\mathbf{x}_P = \Lambda_{\vec{\beta}}\mathbf{x}'_P + \mathbf{a}. \quad (8)$$

In principle, this transformation is done for each single point of an object. The resulting object is called photo-object as it describes the apparent shape of the object seen from the observer. As an example, consider a wireframe model of a cube located in S' moving with $\vec{\beta}$. The observer is located in S at $\vec{x}_{\text{obs}}$, which corresponds to a shift vector $\vec{a} = -\vec{x}_{\text{obs}}$. The twelve edges of the cube are described by straight lines in S', where each point of a line has to be transformed. Figure 4 shows three snapshots of the cube at three different observation times.

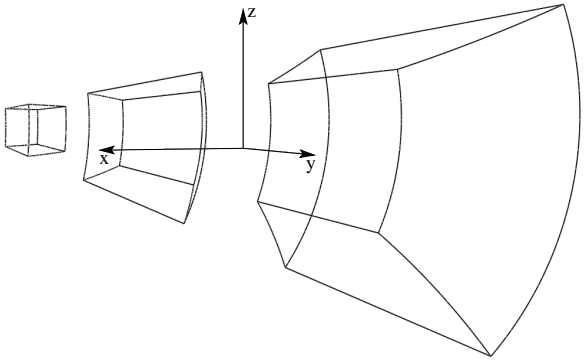

Fig. 4. Photo-objects of a cube with edge length $l = 1$ and velocity $\vec{\beta} = (0, 0.95, 0)^T$ as seen by an observer located at $\vec{x}_{\text{obs}} = (2, 5, 0.2)^T$ with a pinhole camera and 35° field of view. The observation times are $ct_{\text{obs}} = \{4.4, 5.12, 5.66\}$ (from left to right). The center of the cube rests at the origin of S'. The cube appears distorted and rotated due to the finite speed of light and the different times the light needs to reach the observer at the same instants of time from the various points of emission.

3.3 Color

To implement special relativistic effects concerning color, we need a *complete* spectrum $L(\lambda)$, $\lambda \in \mathbb{R}^+$, for each object material. The completeness is crucial because the Doppler shift with Doppler factor $D = \omega/\omega' = \lambda'/\lambda$, as discussed in Sect. 3.1, is able to map the infrared as well as the ultraviolet part of the spectrum into the visual regime. Beside the Doppler shift, however, we also have to take the searchlight effect into account. The transformation of wavelength-dependent radiance between S and S' by the fifth power of the Doppler factor reads [29]

$$L(\lambda, T) = D^5 L'\left(\lambda', T'\right). \quad (9)$$

In general, it is nearly impossible to measure all spectra of an object for all wavelengths. Hence, we use the most natural spectrum in nature generated by an ideal black body. The corresponding Planck spectrum with respect to wavelength λ reads

$$L(\lambda, T) = \frac{2hc^2}{\lambda^5}\left[\exp\left(\frac{hc}{kT\lambda}\right) - 1\right]^{-1}, \quad (10)$$

where h is Planck's constant, k is Boltzmann's constant, and T is the temperature of the black body in degrees Kelvin. The wavelength of maximum intensity follows from Wien's displacement law: $\lambda_{\max} = b/T$ with $b \approx 2.8978 \cdot 10^{-3}\ Km$. The advantage of using this Planck spectrum is that when applying Doppler shift and searchlight effect, we obtain again a Planck spectrum but of different temperature. This immediately follows from the transformation Equation (9) together with $D = \lambda'/\lambda$. Hence, the relation between apparent and actual temperature is simply given by the Doppler factor: $T' = T/D$.

In Fig. 5, the Planck spectrum of a black body at temperature $T = 6000\ K$ is shown. If this black body would directly approach the observer with 20 percent the speed of light, which corresponds to a Doppler factor $D = \sqrt{3/2}$, the observer would recognize a temperature of 7348 K. On the other hand, if the black body would fly in the opposite direction, resulting in $D = 1/\sqrt{3/2}$, it appears to have a temperature of only 4899 K.

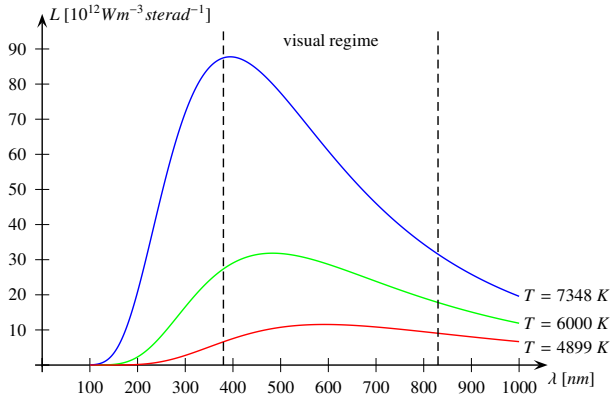

Fig. 5. The Planck spectrum of temperature $T = 6000\ K$ is shifted by Doppler factors $D = \sqrt{3/2}$ and $D = 1/\sqrt{3/2}$.

Now, to convert the resulting Planck spectrum to RGB color, we first calculate the CIE XYZ tristimulus values by a convolution of the Planck spectrum with the CIE color matching functions,

$$\Sigma_j(T) \propto \int L(\lambda, T)\,\sigma_j(\lambda)d\lambda, \quad (11)$$

where $\Sigma_j = (X, Y, Z)$ and $\sigma_j(\lambda) = \{\bar{x}(\lambda), \bar{y}(\lambda), \bar{z}(\lambda)\}$ (see e.g. Wyszecki [31]). Then, we convert the CIE XYZ tristimulus values to linear RGB color space using the *ITU-R Recommendation BT.709* with transformation matrix

$$\begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix} = \begin{bmatrix} 3.2405 & -1.5372 & -0.4985 \\ -0.9693 & 1.8760 & 0.0416 \\ 0.0556 & -0.2040 & 1.0573 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \quad (12)$$

see e.g. Poynton [17] for a detailed discussion of color transformation. The final transformation to sRGB space reads

$$C_{\text{sRGB}} = \begin{cases} 12.92 C_{\text{linear}}, & C_{\text{linear}} \le 0.0031308 \\ (1+a) C_{\text{linear}}^{1/2.4} - a, & C_{\text{linear}} > 0.0031308 \end{cases}, \qquad (13)$$

where $a = 0.055$ and $C_{\text{linear}} = \{R_{709}, G_{709}, B_{709}\}$. Unfortunately, without any modifications, this method would map all temperatures below 1900 K to yellow and the others to white due to gamut restrictions. Hence, we have to scale the tristimulus values either individually or by a constant factor before mapping them to the sRGB color space. One local tone-mapping method is to scale the tristimulus values according to their luminosity: $\Sigma_j \to \Sigma_j / Y$, which results in the color table as shown in Fig. 6 (left). Here, the vertical axis scales the C_{sRGB} values by a material dependend reflection factor $f \in [0,1]$ as we will do in our local ray tracing method, see Sect. 4.2. This might be a

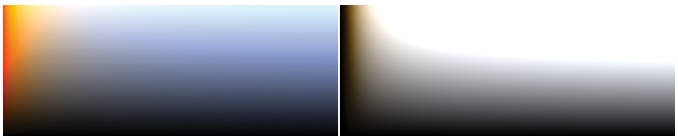

Fig. 6. Color tables for Planck spectra from $T_{\text{min}} = 1000\ K$ to $T_{\text{max}} = 26000\ K$ (horizontal axis) scaled by the luminance Y for each temperature (left) or by the luminance for $T = 1500\ K$ (right). The vertical axis scales C_{sRGB} by a factor $f \in [0,1]$, where f will be f_{ar} as in Eq. (18).

quite appealing color mapping, but we lose the luminosity information and, thus, the searchlight effect. An alternative tone-mapping method scales all tristimulus values by a definite luminance value. Here, we use the luminance of a black body with temperature $T = 1500\ K$. Additionally, we apply a global logarithmic tone-mapping function like e.g. $C_{\text{sRGB}} \to 0.6 \cdot \log_{10}(C_{\text{sRGB}} + 1)$. A temperature of about 1500 K is now mapped to a dark red color, whereas temperatures above 3600 K are mapped to white, see Fig. 6 (right) with $f = 1$. The resulting images can be seen in Fig. 7. For all other figures, however, we use the color table of Fig. 6 (left).

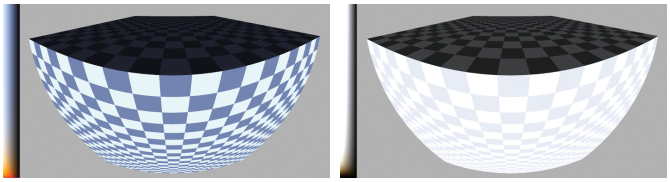

Fig. 7. The different color mappings for Planck spectra from Fig. 6 applied to a cube moving at 95 percent the speed of light toward the observer at position $\vec{x}_{\text{obs}} = (0.03, 4.46, 0.6)^T$. The center of the cube rests at the origin of S'. The light source is at $\vec{x}_{\text{light}} = (90, 0, 9.8)^T$. The right image retains the searchlight effect, while the mapping in the left image has higher contrast on the textured faces allowing for better perception of the distortion of the cube.

4 Algorithm and Implementation

Our approach follows the idea of ray tracing for optimal image quality. In principle, for each fragment (point) of the image a viewing ray of light is traced back from the observer's position in the corresponding direction defined by the perspective projection and the viewing angle. This viewing ray is intersected with the geometry of the meshes to be rendered and at each intersection point the resulting color is evaluated. This implies that for each viewing ray an intersection test must be performed with each triangle of the scene geometry, which is in general not possible in interactive computer graphics. The method of local ray tracing of point-based glyph data, see e.g. Reina and Ertl [18], provides a solution by not searching the intersection of the nearest triangle for a specific image fragment's viewing ray but correctly projecting a triangle to image-space, determining the fragments for whose viewing rays the triangle is possibly visible, and only testing this triangle with these fragments. By performing this calculation for each triangle of the mesh, the whole image is constructed and expensive object-space intersection search is avoided.

In our case, this projection of the triangles of the mesh to image-space is the calculation of the apparent triangles (photo-objects) with respect to the observer's frame S. Obviously, no polygonal proxy geometry can be the precise image-space silhouette of the apparent triangle. So we create a tight-fitting rectangular proxy geometry as conservative approximation.

For simplification reasons the scenario investigated in the rest of this article defines one constraint: the observer and the light position are at rest in the common reference system S. An object of interest is also at rest but with respect to the reference system S'. The system S' moves with constant velocity with respect to S as detailed in Sect. 3.1. When multiple objects are visualized simultaneously, each one can have its own reference system.

Algorithm 1 shows a short sketch of our local ray tracing method. The details are described in the following sections. The code in green is implemented either in the vertex or in the geometry shader whereas the red text is processed in the fragment shader.

Algorithm 1 Local ray tracing

```
for all triangles Δ(P'_1, P'_2, P'_3) do
  Calculate apparent points P_i and Q_ij via R. {Sect. 3.2}
  Transform P_i, Q_ij into image-space and find intersections G̃_ij. {Sect. 4.1}
  Determine window-aligned proxy geometry M from P̃_i,G̃_ij. {Sect. 4.1}
  for all fragments f ∈ M do
    Generate view direction d⃗ from x⃗_obs and x⃗_f with respect to S. {Sect. 4.2}
    Transform d⃗ and x⃗_obs into S'. {Eqn. (3), (5)}
    Find intersection x'_is between d⃗' and Δ(P'_1, P'_2, P'_3). {Sect. 4.2}
    Transform x'_is into S. {Eq. (8)}
    Calculate combined Doppler factor at x_is. {Eq. (17)}
    Do Phong-illumination with Planck spectrum. {Fig. 9}
  end for
end for
```

4.1 Defining the proxy geometry

The proxy geometry as silhouette approximation for the apparent triangle is generated in the vertex-shader or the geometry-shader stage, depending on the graphical primitive used. The classical approach uses a point primitive, which generates a single image-space point-sprite for each vertex uploaded. On current graphics cards, the geometry shader provides an alternative allowing for a tighter fitting primitive. In both cases, we upload each triangle using a single vertex with the two additional positions, normal vectors, and texture coordinates as vertex attributes. The configuration of the moving object system S', like shift vector $\vec{a}$, speed $\vec{\beta}$, and Lorentz matrices, is constant for each object, and therefore, these parameters are transferred as uniforms. The triangle mesh is stored in the graphics card's memory using vertex buffer objects.

Preliminary calculations showed that the apparent shape of a straight line is a hyperbola in the plane spanned by the line and the moving direction $\vec{\beta}$. This also applies to the edges of a triangle. Any segment of a hyperbola between two points is enclosed by the triangle formed by these points and the intersection point of their tangents. To obtain a polygon that is guaranteed to enclose the apparent triangle, we compute the convex hull of the polygons enclosing the three hyperbolae for the three edges of the triangle, which can also be understood as union of the triangles enclosing the edge hyperbolae and the triangle of the corner points of the apparent triangle itself, see Fig. 8.

To do this, we first apply the object-space transformation $\mathscr{R}$, see Sect. 3.2, to the three triangle vertices P'_i and the six additional vertices Q'_{ij} defined by

$$\vec{x}'_{Q_{ij}} = \vec{x}'_{P_i} + \varepsilon \left(\vec{x}'_{P_j} - \vec{x}'_{P_i} \right), \quad i \ne j \in \{1,2,3\}, \quad \varepsilon \ll 1. \qquad (14)$$

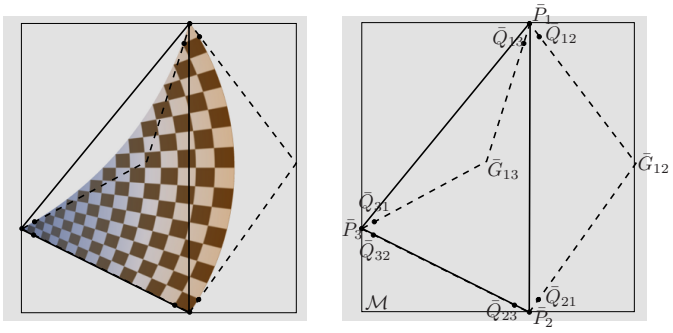

Fig. 8. The convex hull of the apparent curvilinear triangle, constructed from the corner points $\bar{P}_i$ and the intersection points $\bar{G}_{ij}$, defines the most favored estimate of the footprint. A more conservative approximation follows from a min-max calculation of all six points $(\bar{P}_i, \bar{G}_{ij})$ resulting in the window-aligned rectangle $\mathscr{M}$. In this example, $\bar{P}_2$, $\bar{Q}_{23}$, $\bar{Q}_{32}$, and $\bar{P}_3$ lie all on the same line, so $\bar{G}_{23}$ is just the midpoint between $\bar{P}_2$ and $\bar{P}_3$.

In our implementation, we use $\varepsilon = 0.01$. The resulting apparent vertices P_i and Q_{ij} are transformed into image-space (more precisely: into device coordinates) by means of the model-view-projection matrix, $\mathscr{I} : P_i \to \bar{P}_i$, $Q_{ij} \to \bar{Q}_{ij}$, and scaling with the viewport size. $\bar{P}_i$ and $\bar{Q}_{ij}$ form a tangent vector at point $\bar{P}_i$. The intersection $\bar{G}_{ij}$ of the two tangents $\bar{P}_i\bar{Q}_{ij}$ and $\bar{P}_j\bar{Q}_{ji}$ of each edge (ij) is then calculated, resulting in the enclosing triangle for that edge.

Putting all together, the convex hull for the apparent curvilinear triangle would require the geometry shader to output a variable number of vertices (3 to 6). Although the geometry shader is capable of doing this, the performance then decreases dramatically; that is a well known limitation of the geometry shader stage. As a compromise, we use a window-aligned rectangle that follows from a min-max calculation of all six points $(\bar{P}_i, \bar{G}_{ij})$. Figure 8 shows an example where the convex hull of the apparent curvilinear triangle is specified by the polygon $(\bar{P}_1, \bar{P}_3, \bar{P}_2, \bar{G}_{12})$.

The intersections G_{ij} may be behind the camera's position in S, resulting in a wrong intersection in image-space $\bar{G}_{ij}$ due to the perspective projection. Fortunately, this only happens for very distorted triangles, i.e. very fast triangles, very close to the observer and thus resulting in large image-space proxy geometries, meaning that there can only be very few such triangles visible. Using a shader program capable of managing the silhouette approximation both for the common case and this special case slows down the whole system extremely (by more than one order of magnitude). However, since this special case is rare, the geometry shader generates a screen-filling proxy geometry for the very distorted triangles, ensuring artifact free images, while only introducing an acceptable loss of performance.

4.2 Local ray tracing

The rasterization stage of the graphics pipeline automatically generates image-space fragments covering the whole proxy geometry $\mathscr{M}$. For each fragment $f \in \mathscr{M}$ we determine a local light ray $\vec{d} = (\vec{x}_{\text{obs}} - \vec{x}_{\text{f}})/\|\vec{x}_{\text{obs}} - \vec{x}_{\text{f}}\|$ from the observer's position $\vec{x}_{\text{obs}}$ and the fragment position $\vec{x}_{\text{f}}$ that follows from the transformation of the fragment coordinates by means of the inverse model-view-projection matrix. We then transform the corresponding four-vector $\mathbf{d} = (1, \vec{d})$ and the observer's position into the triangle's rest frame S'. In S', the triangle is flat and is represented by

$$\vec{x}'(\zeta, \xi) = \vec{x}'_1 + \zeta\vec{\sigma} + \xi\vec{\tau}, \tag{15}$$

where $\vec{\sigma}$ and $\vec{\tau}$ are the normalized span vectors from $\vec{x}'_1$ to $\vec{x}'_2$ and $\vec{x}'_3$, respectively. The intersection follows from solving the linear equation $\mathbf{Ay} = \mathbf{b}$, where $\mathbf{b} = \vec{x}'_{\text{obs}} - \vec{x}'_1$ and

$$\mathbf{A} = \begin{pmatrix} \sigma_x & \tau_x & -d'_x \\ \sigma_y & \tau_y & -d'_y \\ \sigma_z & \tau_z & -d'_z \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} \zeta \\ \xi \\ \psi \end{pmatrix}. \tag{16}$$

For a valid intersection $\vec{x}'_{\text{is}}$, Equation (7) delivers the time t'_{is} when light must be emitted in order to reach the observer at observation time t'_{obs}. The back transformation of the point $\mathbf{x}_{\text{is}} = (t'_{\text{is}}, \vec{x}'_{\text{is}})$ by means of Equation (8) yields the apparent position $\vec{x}_{\text{is}}$ of the intersection with respect to S. The distance between the apparent position and the observer $\|\vec{x}_{\text{is}} - \vec{x}_{\text{obs}}\|$ is written as fragment depth. This enables our implementation to handle depth-sorting, occlusion, and even intersection of the apparent objects correctly by simply utilizing the depth test of the graphics hardware.

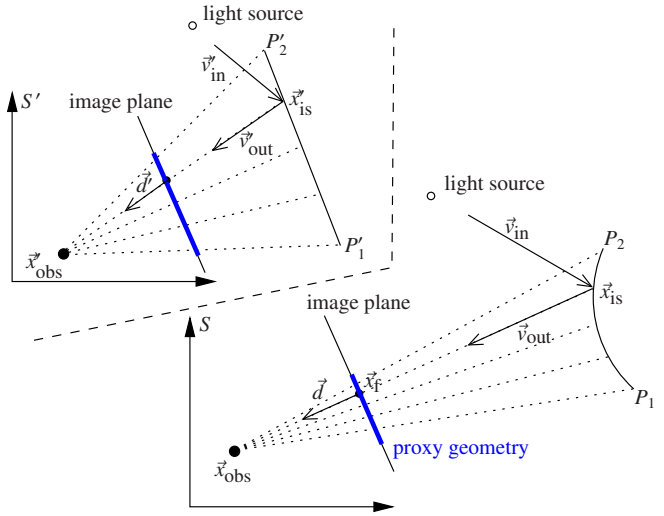

Fig. 9. 2D sketch of the local ray tracing method. For each fragment f of the proxy geometry, the intersection $\vec{x}_{\text{is}}$ of the light ray $\vec{d}$ with the photo-object P_1P_2 (curved solid line) must be determined. Note that $\vec{d}$ represents an incident light ray. However, the actual intersection calculation is done in S', where $P'_1P'_2$ is a straight line. The angles between the light rays (dashed lines) are smaller in S compared to S', since object and observer approach each other, resulting in aberration.

Figure 9 shows a 2D sketch of our local ray tracing method. After the proxy geometry has been constructed by means of the apparent positions P_1 and P_2 of an edge as described in the previous section, the intersection of the light ray $\vec{d}'$ with the flat edge $P'_1P'_2$ in S' delivers the apparent point $\vec{x}_{\text{is}}$.

To aid the perception of the geometry of the object several visual cues are required, like texturing and lighting. Texturing is possible by interpolation of the texture coordinates of the vertices based on ζ and ξ, including texture filtering through mip-mapping. However, lighting the triangle is slightly more complex. The light source, which rests at $\vec{x}_{\text{light}}$ with respect to S, illuminates the object with a Planck spectrum of temperature T_{light}. For Phong-like illumination, we first determine the incident and outgoing light directions at the apparent position $\vec{x}_{\text{is}}$ with respect to S, see Fig. 9. The corresponding four-vectors read $\mathbf{v}_{\text{in}} = \omega_{\text{in}}(1, \vec{v}_{\text{in}})$ and $\mathbf{v}_{\text{out}} = \omega_{\text{out}}(1, \vec{v}_{\text{out}})$. Both have to be transformed to S' by means of Equation (3) to determine the combined Doppler factor

$$D = \frac{1 - \vec{\beta} \cdot \vec{v}_{\text{out}}}{1 - \vec{\beta} \cdot \vec{v}_{\text{in}}}. \tag{17}$$

Fortunately, the Planck temperature T_{light} of the light source only has to be divided by the Doppler factor D to obtain the apparent temperature T_{app}. The corresponding sRGB color, C_{sRGB}, see Sect. 3.3, can be read from the precalculated one-dimensional color lookup table.

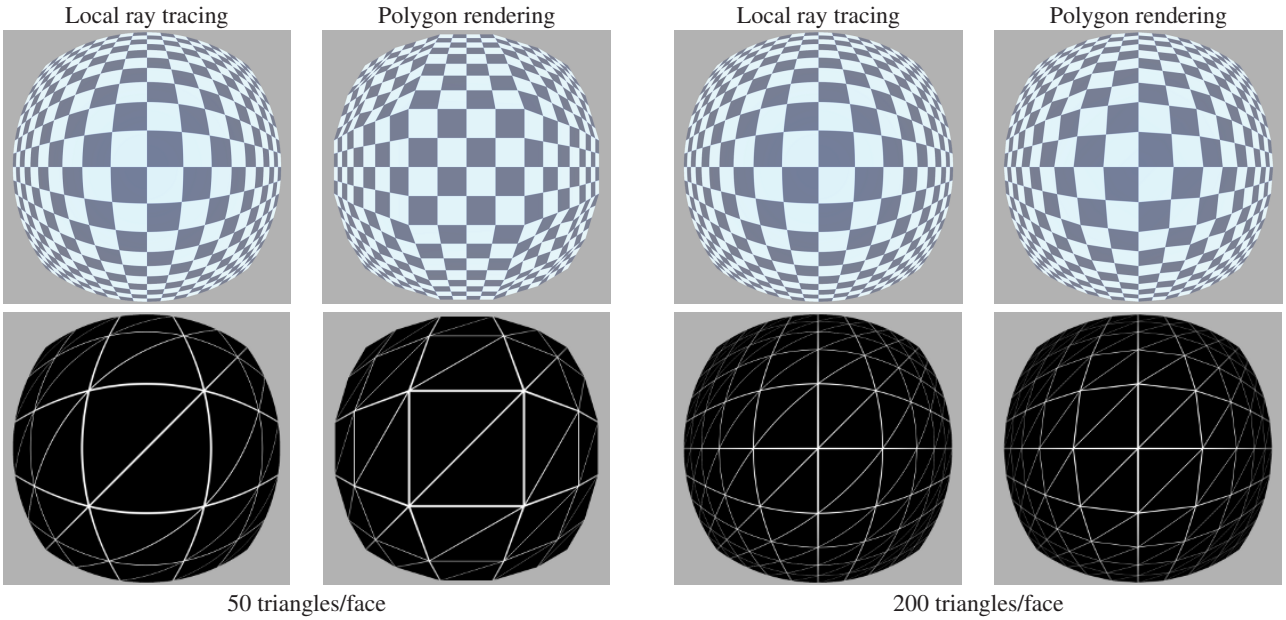

Fig. 10. A cube of size $l = 5$ approaches the observer with 95 percent the speed of light. **Top:** Checker texture with Doppler effect. **Bottom:** The wireframe texture shows the triangle edges.

The diffuse and specular reflection factors f_{diff} and f_{spec} are determined by standard methods using the above derived local light directions $\vec{v}'_{\text{in}}$ and $\vec{v}'_{\text{out}}$. The surface normal at $\vec{x}_{\text{is}}$ is interpolated according to $\vec{n}_{\vec{x}_{\text{is}}} = (1-\zeta_{\text{is}}-\xi_{\text{is}})\vec{n}_1 + \zeta_{\text{is}}\vec{n}_2 + \xi_{\text{is}}\vec{n}_3$ with normals $\vec{n}_i$ at vertices $\vec{x}_i$.

Since wavelength-dependent reflection data is not available from typical RGB-based material descriptions for scene objects, we use the RGB textures of the object to define an artificial reflectance factor

$$f_{\text{ar}} = \frac{1}{2}(\mathscr{G}+1)\left(f_{\text{amb}} + f_{\text{diff}} + f_{\text{spec}}\right), \tag{18}$$

where $\mathscr{G} = 0.3R + 0.59G + 0.11B$ is the gray value for the respective texture colors and f_{amb} is an arbitrary ambient factor. The color displayed at the end now follows from the multiplication $C_{\text{display}} = f_{ar} \cdot C_{\text{sRGB}}$.

5 Results

Using the method of local ray tracing we are able to calculate the correct viewing ray for each image fragment and thus the resulting images are optimal in image quality (not considering aliasing artifacts due to under-sampling). As stated earlier, the polygon-based method of creating a photo-object in S would require a 3D model with very small triangles, in the worst case triangles covering a single fragment each, to achieve equal results, which is not practical.

Figure 10 shows a comparison of our method with the polygon based method for a simple test case: a cube approaching the observer with 95 percent the speed of light. The left two columns of images show a cube with 50 triangles per face whereas the right two columns show a cube with 200 triangles per face. For the left images, the differences are clearly visible both in the wireframe rendering of the mesh and on the textured cube. In the right images, the differences in the wireframe rendering are less visible, however, the error in the distortion of the textured cubes is still present.

Our method is clearly superior to the polygon-based approach in cases where large triangles are close to the observer or traveling fast and are thus highly distorted. In cases when the triangles' footprints in image-space remain rather small, the quality differences of our method and polygon rendering are negligible. This can be seen in Fig. 11, where we use a typical 3D model like the Tie-Fighter [3]. The model was created for conventional computer graphics and is not especially prepared for our scenario. The solar panels are modeled with only a few rather large triangles and the differences between our method and polygon rendering are extreme. On the other hand, the cockpit sphere in the center is modeled by many rather small triangles and both methods result in roughly the same images for this part.

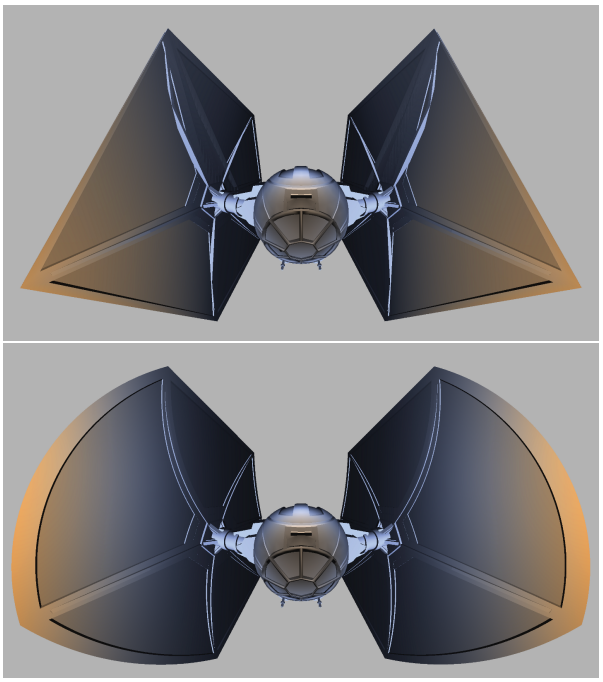

Fig. 11. Tie-Fighter model [3] approaches the observer with 90 percent the speed of light. The position of the observer is $\vec{x}_{\text{obs}} = (0, 15.2, 0.46)^T$ and the light source is located at $\vec{x}_{\text{light}} = (0, 12.5, 0.62)^T$ **Top:** polygon rendering results in a wrong shape of the solar panels, since they are modeled with few long triangles. **Bottom:** local ray tracing correctly creates a distorted image of the solar panels, independent of its modeling.

When rendering 3D models not specifically prepared for the distortion of special relativistic visualization but for generic computer graphics, additional problems may arise other than just inaccurate shape representations. It is common practice in 3D modeling to add

details to polygonal models by adding small parts intersecting with the base model. These parts are not connected on the vertex-level but result in the desired shapes when rendered. However, if these parts are distorted through aberration and the triangles of the base mesh become large in image-space, these parts can get visually disconnected (see Fig. 12) by polygon-based rendering, whereas our approach of local ray tracing correctly shows the contiguous model. This issue is related to the "T-vertices" problem, which would result in similar visual artifacts.

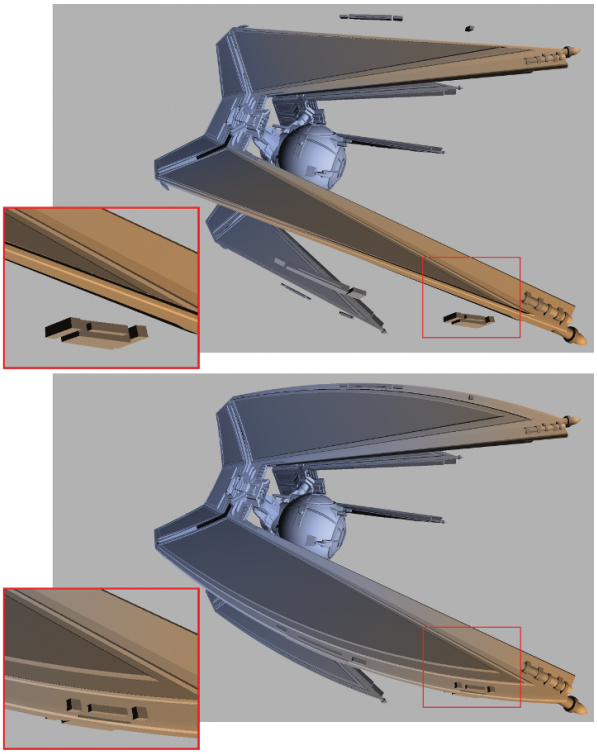

Fig. 12. Tie-Defender like 3D model [3] flies with 90 percent the speed of light in y-direction. The observer is located at $\vec{x}_{\text{obs}} = (0.84, 1.93, 0.31)^T$ and the light source is at $\vec{x}_{\text{light}} = (7.3, -3.3, 3.1)^T$. Top image shows polygon rendering, where parts of the model are disconnected since they are only modeled via intersection. Bottom image shows the result of local ray tracing. The model remains intact.

Figure 1 shows a scene with three objects moving in different directions with different velocities. They all expose different effects in lighting and (slight) distortions. Such a setup cannot easily be rendered with image-space approaches in general. Each object, more precisely each differently moving reference system, would require its own image-space pass. A final compositing pass would generate the final image from the intermediate results, requiring depth information and additional geometric information, like normal vectors, if deferred shading is to be applied for lighting with a light source outside the objects frame of reference. While being faster than our method, image-space approaches always have the problem of the finite resolution of the image-buffers, while for local ray tracing such resolution issues only appear for the textures placed on objects, which is an inherent problem of texturing (except for procedural textures).

Figure 13 shows a direct comparison between the image-based method, polygon rendering, and our local ray tracing method. Although the observer looks diagonally to the right, the Tie-Interceptor like 3D model, which moves with 99 percent the speed of light, is already behind the observer, which results in a strong magnification of the right wing. Here, it is apparent that the image-based method forfeits quality due to the limited resolution of the cube map. Polygon rendering, on the other side, does not reproduce the correct shape of the model. Our local ray tracing method is able to produce the correct shape with high quality.

Our approach scales well with the size of the 3D models rendered, as well as with the number of differently moving reference systems. Each triangle cloud has its own reference system without significant impact on the rendering performance. Since our method heavily utilizes the features of modern programmable graphics hardware, we achieve interactive rendering performance on commodity desktop computers. Our test system consists of an Intel Core 2 6600 CPU at 2.40 GHz, 2 GB RAM, and an NVidia GeForce GTX 280 graphics card. All performance measurements were conducted with a 1280×720 viewport. Our algorithm is implemented in OpenGL using GLSL shaders. As Table 1 shows, our system reaches interactive frame rates, while the result images in this article and the supplemental video prove our high image quality. The cube meshes are quite

Table 1. Rendering performance values in frames per second of our local ray tracing (LRT) method in contrast to the image-based (IB) and polygon methods for 3D scenes with different number of triangles. Interactivity is reached for all data sets on commodity graphics hardware. For the image-based method, we use a cube map with resolution of 2048^2 pixels per face.

Scene	# of Δ	IB	Polygon	LRT
cube 50 (Fig. 10)	300	360	3170	640
cube 200 (Fig. 10)	1200	350	3170	560
Tie-Fighter (Fig. 11)	27538	190	1900	75
Tie-Defender (Fig. 12)	61188	66	430	40
teaser (Fig. 1)	135994	–	–	62

small and thus the rendering performance is very high. Tie-Fighter and Tie-Defender achieve interactive frame rates. They are of medium size and the viewing parameters used when measuring the rendering performance are the same as those for the referenced figures. High distortions are visible and the differences from polygon rendering are clear. The teaser scene renders faster than the individual objects. This is due to the fact that although it contains more triangles, the image-space footprints of the meshes—and thus of the triangles—are much smaller, resulting in less workload for the fragment processing stage.

6 Conclusion and Future Work

We have presented an approach for interactive special relativistic visualization of arbitrary 3D polygon meshes. Our method creates per-pixel accurate images through local ray tracing on the GPU and results therefore in higher image quality compared to polygon-based and image-space rendering. By utilizing programmable graphics hardware, we reach interactive frame rates for mid-size polygon meshes. Our method is able to handle many different moving reference systems, i.e. multiple objects moving with different directions and speeds. Many visual effects, like aberration of light, Doppler shift, as well as the searchlight effect, can be observed with our software.

While uniform motion is, in principle, feasible at arbitrary relative speed, rapid rotation or deformation of a solid object is more challenging and is not handled in our system. In particular, acceleration should not be handled by just kinematic description, but should also include the consideration of dynamics and effects of forces. If, for example, a wheel is accelerated toward relativistic rotational speeds, its perimeter shrinks, which leads to enormous stresses that would finally disrupt the wheel. A step toward visualizing wheels rotating at relativistic speed was taken by Kraus et al. [13]. However, their work does not consider the acceleration phase from static to relativistic rotation. A comprehensive model to handle the relativistic physics for general scenarios of acceleration and deformation is still an open problem for future work. If the physical problems are neglected, four-dimensional special relativistic ray tracing is the rendering technique of choice for scene objects with arbitrary acceleration [26, Ch. 5.2]. Local ray tracing, as of this paper, assumes a uniform frame of reference across a geometric primitive (i.e., triangle) and, thus, cannot be applied to accelerating objects.

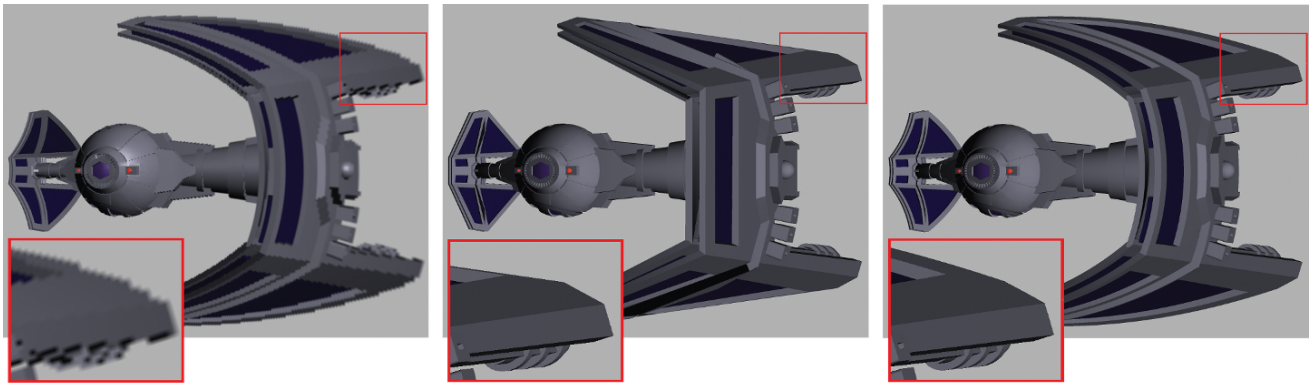

Fig. 13. Comparison between image-based (left), polygon (center), and local ray tracing (right) method. The Tie-Interceptor like 3D model passes the observer with 99 percent the speed of light. The strong magnification of the right wing results in poor visual quality of the image-based approach.

As future work, we plan to extend our software to a freely available tool usable for teaching in the context of Special Relativity. We want to allow the user to interactively explore relativistic effects by supporting import of arbitrary 3D models from common file formats and graphical interaction with the relevant visualization parameters, e.g., observer's position, directions of motion, speed, and the different visual effects shown (geometric only, Doppler shift, and searchlight effect).

Acknowledgments

This work is partially funded by Deutsche Forschungsgemeinschaft (DFG) as part of Collaborative Research Centre SFB 716 and DFG project "Astrographik". We wish to thank Guido Reina for his help making the supplemental video and Manfred Bauer for the Tie-Fighter, Tie-Defender, and Tie-Interceptor 3D models.

References

[1] The Milky Way panorama is by ESO/S. Brunier. http://www.eso.org/public/images/eso0932a.

[2] According to private communications with A. Hanson, there is even earlier unarchived and unpublished software for special relativistic rendering.

[3] Website providing the Tie-Fighter, Tie-Defender, and Tie-Interceptor like 3D meshes, http://mitglied.multimania.de/Mesh_Factor_Y/meshes/meshes.htm.

[4] C. Betts. Fast rendering of relativistic objects. *The Journal of Visualization and Computer Animation*, 9(1):17–31, 1998.

[5] M.-C. Chang, F. Lai, and W.-C. Chen. Image shading taking into account relativistic effects. *ACM Transactions on Graphics*, 15(4):265–300, Oct. 1996.

[6] G. Gamow. *Mr. Tompkins in Wonderland*. University Press, Cambridge, 1939.

[7] J.-D. Gascuel, N. Holzschuch, G. Fournier, and B. Péroche. Fast non-linear projections using graphics hardware. In *I3D '08: Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games*, pages 107–114, 2008.

[8] W. Gekelman, J. Maggs, and L. Xu. Real-time relativity. *Computers in Physics*, 5(4):372–385, July/Aug. 1991.

[9] S. Gumhold. Splatting illuminated ellipsoids with depth correction. In *Proceedings of 8th International Fall Workshop on Vision, Modelling and Visualization*, pages 245–252, 2003.

[10] P.-K. Hsiung and R. H. P. Dunn. Visualizing relativistic effects in spacetime. In *Proceedings of Supercomputing '89*, pages 597–606, 1989.

[11] 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, 1990.

[12] P.-K. Hsiung, R. H. Thibadeau, and M. Wu. T-buffer: Fast visualization of relativistic effects in spacetime. *Computer Graphics*, 24(2):83–88, Mar. 1990.

[13] U. Kraus, H. Ruder, D. Weiskopf, and C. Zahn. Was Einstein noch nicht sehen konnte. Schnelle Computer visualisieren relativistische Effekte. *Physik Journal*, 1(7/8):77–83, Jul. 2002.

[14] A. Lampa. Wie erscheint nach der Relativitätstheorie ein bewegter Stab einem ruhenden Beobachter? *Zeitschrift für Physik*, 27:138–148, 1924.

[15] J. Li, H.-Y. Shum, and Q. Peng. An improved spacetime ray tracing system for the visualization of relativistic effects. In *Eurographics 2001 Short Presentations*, 2001.

[16] R. Penrose. The apparent shape of a relativistically moving sphere. *Mathematical Proceedings of the Cambridge Philosophical Society*, 55:137–139, 1959.

[17] C. Poynton. *Digital Video and HDTV Algorithms and Interfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003. The primary chromaticities of the Rec. 709 standard read: $x_R = 0.64$, $y_R = 0.33$, $x_G = 0.3$, $y_X = 0.6$, $x_B = 0.15$, $y_B = 0.06$, $x_{w,D65} = 0.3127$, $y_{w,D65} = 0.3290$.

[18] G. Reina and T. Ertl. Hardware-accelerated glyphs for mono- and dipoles in molecular dynamics visualization. In *Proceedings of EUROGRAPHICS - IEEE VGTC Symposium on Visualization Eurovis '05*, pages 177–182, 2005.

[19] W. Rindler. *Relativity – Special, General and Cosmology*. Oxford University Press, 2001.

[20] C. Savage, D. McGrath, T. McIntyre, M. Wegener, and M. Williamson. Teaching physics using virtual reality. arXiv: 0910.5776v1 [physics.ed-ph], 2009.

[21] C. M. Savage, A. Searle, and L. McCalman. Real time relativity: exploratory learning of special relativity. *American Journal of Physics*, 75:791–798, 2007.

[22] G. D. Scott and H. J. van Driel. Geometrical appearances at relativistic speeds. *American Journal of Physics*, 38:971–977, 1970.

[23] G. D. Scott and M. R. Viner. The geometrical appearance of large objects moving at relativistic speeds. *American Journal of Physics*, 33:534–536, 1965.

[24] E. F. Taylor. Space-time software: Computer graphics utilities in special relativity. *American Journal of Physics*, 57:508–514, 1989.

[25] J. Terrell. Invisibility of the Lorentz contraction. *Physical Review*, 116(4):1041–1045, Nov. 1959.

[26] D. Weiskopf. *Visualization of Four-Dimensional Spacetimes*. PhD thesis, Eberhard-Karls-Universität Tübingen, 2001. http://nbn-resolving.de/urn:nbn:de:bsz:21-opus-2400.

[27] D. Weiskopf, M. Borchers, T. Ertl, M. Falk, O. Fechtig, R. Frank, F. Grave, P. Jezler, A. King, U. Kraus, T. Müller, H.-P. Nollert, I. Rica Mendez, H. Ruder, T. Schafhitzel, C. Zahn, and M. Zatloukal. Explanatory and illustrative visualization of special and general relativity. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):522–534, 2006.

[28] D. Weiskopf, D. Kobras, and H. Ruder. Real-world relativity: Image-based special relativistic visualization. In *Proceedings of the IEEE Visualization 2000 Conference*, pages 303–310, 2000.

[29] D. Weiskopf, U. Kraus, and H. Ruder. Searchlight and Doppler effects in the visualization of special relativity: A corrected derivation of the transformation of radiance. *ACM Transactions on Graphics*, 17(3):278–292, Jul. 1999.

[30] V. F. Weisskopf. The visual appearance of rapidly moving objects. *Physics Today*, 13(9):24–27, Sept. 1960.

[31] G. Wyszecki and W. Stiles. *Color Science – Concepts and Methods, Quantitative Data and Formulae*. Wiley Classics Library, 2000.