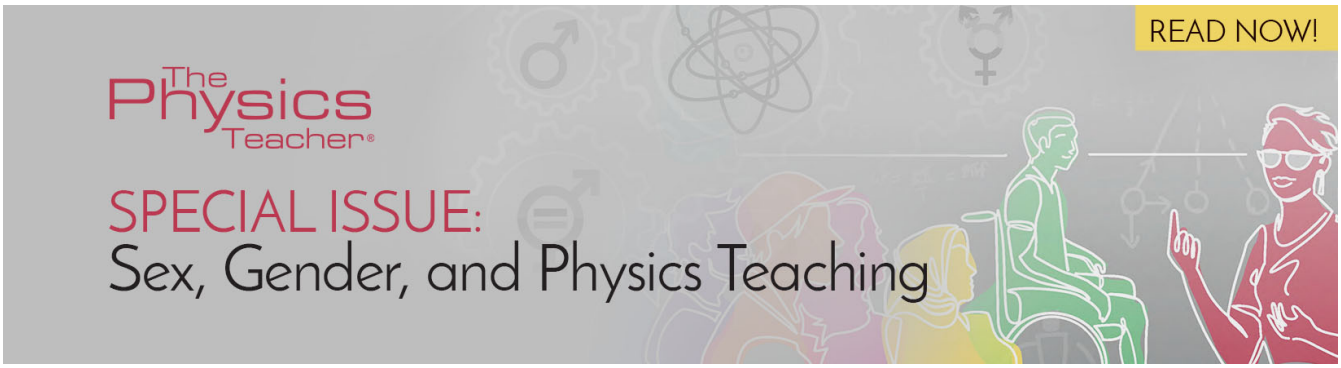
Visualizing relativity: The OpenRelativity project

Zachary W. Sherin, Ryan Cheu, Philip Tan, and Gerd Kortemeyer

Citation: American Journal of Physics **84**, 369 (2016); doi: 10.1119/1.4938057
View online: https://doi.org/10.1119/1.4938057
View Table of Contents: https://aapt.scitation.org/toc/ajp/84/5
Published by the American Association of Physics Teachers



Visualizing relativity: The OpenRelativity project

Zachary W. Sherin, Ryan Cheu,[a] and Philip Tan
Game Lab, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

Gerd Kortemeyer[b]
Lyman Briggs College and Department of Physics and Astronomy, Michigan State University, East Lansing, Michigan 48825

(Received 1 August 2015; accepted 17 November 2015)

We present OpenRelativity, an open-source toolkit to simulate effects of special relativity within the popular *Unity* game engine. Intended for game developers, educators, and anyone interested in physics, OpenRelativity can help people create, test, and share experiments to explore the effects of special relativity. We describe the underlying physics and some of the implementation details of this toolset with the hope that engaging games and interactive relativistic "laboratory" experiments might be implemented. [http://dx.doi.org/10.1119/1.4938057]

I. INTRODUCTION

"What would you see if you were riding a beam of light?" Einstein is reported[1] to have asked this question at the age of 16, only to find out a decade later that you can fundamentally never travel at that speed.[2,3] You can, however, travel near the speed of light, at least as a thought experiment, but what you would actually see was not correctly answered until two decades later.

There have been multiple attempts to visualize a relativistic world, perhaps most famously in 1940 by Gamow in his whimsical *Mr. Tompkins in Wonderland*.[4] In this story, Mr. Tompkins is riding his bicycle through a dreamworld city where the speed of light is lower (in fact only a little higher than the speed of a bicycle), and since all of special relativity scales with v/c, effects of relativity thus became visible at bicycle speeds. This book thus brought special relativity to a human scale. Unfortunately, though, Gamow apparently had not read Lampa,[5] who found in 1924 that due to the finite time that it takes light to run from the source to the observer, length contraction does not necessarily make objects look shorter: what you would *measure* is not what you would *see*. You would measure the length of a stick by determining the positions of its front and back end at the same instant in your reference frame, but you would see those two ends of the stick at different times in the past. In an educational context, Scherr later described these perspectives—what you measure and what you see—as ultimate reality versus visual reality.[6] In fact Lampa's work remained virtually forgotten for another two decades, until Terrell[7] and Penrose[8] again took into account that light emitted at a larger distance actually comes from further in the past. In addition to the already known fact that length contraction is not always visible as a contraction, they also found that spheres always appear spherical, regardless of relative speed (however, surface textures appear to stretch around the spherical shape).

To get the full picture, two more effects need to be taken into account, namely, the Doppler shift and relativistic aberration. The Doppler shift changes the apparent color of approaching and receding objects (including the famous "red shift" of spectral lines from stars in an expanding Universe). Relativistic aberration is due to the fact that more photons are directed at you from the direction into which you are traveling, leading to increased apparent brightness of objects that are in the direction of travel (often described as the "searchlight effect").

Incorporating all of these effects requires computation. The foundations for computer-generated visualizations of special relativity were laid four decades later by Weiskopf,[9] which led to a number of computer-generated movies,[10] including (in reference to Gamow and just in time for the 100th anniversary of the 1905 papers) a bicycle ride through the German city of Tübingen.[11] These visualizations "slow down" light, providing a mechanism to experience special relativity outside of non-intuitive environments such as space travel.

The first interactive first-person visualization of relativity was provided by Savage *et al*. as *Real Time Relativity*,[12] which has been used for teaching purposes at the Australian National University and elsewhere.[13] The project does not slow down light, but instead moves the viewer into space, where he or she actually travels fast. As opposed to the earlier movies, the movement of the first-person viewer is completely controllable. The *Real Time Relativity* engine does not include third-party movement, in that nothing but the viewer moves. While this may appear like an arbitrary omission and unfounded restriction, it is in fact due to the significant challenges encountered when trying to trace the history of an object to find out when it would have emitted photons that are momentarily visible to the viewer.

The third-party motion restriction was overcome by Doat *et al*. in a virtual billiards game that tracked the motion of a limited number of third-party objects. An advantage of this approach is that the same scenario can asynchronously be viewed from different frames of reference.[14] The current version of this billiards game, however, does not take into account Doppler shift and the searchlight effect.

In this paper, we describe the development of a code library for games that take place in relativistic environments. Education can be assisted through the use of games and other interactive media, especially for topics that are frequently difficult to understand and visualize. Special relativity is a physics topic for which it is challenging to develop intuition, and instructors need to be prepared to confront misconceptions and be strategic in refining intuition.[6] The toolset can help educators create new demonstrations to provide an intuitive, useful understanding of this complex topic. The hope is that through interactive visualization, students gain a "feel" for relativity and over time develop an intuition about the effects. To that end, similar to Kraus and Borchers,[11] we

also allow game scenarios within human-scale environments, and introduce special relativity by "slowing down" light.

II. OVERVIEW OF OpenRelativity

Most video games are written within game engines, which provide libraries and tools for the basic geometry, object management, and physics to be implemented. As the underlying game engine for OpenRelativity we chose the popular *Unity* engine,[15] which is available both in a free and a paid version. Game engines typically take care of perspective, hidden lines, movement, and rotation, and are optimized to interact with modern graphics cards. The challenge in adapting engines to a relativistic game environment is that those functions need to be replaced, and this needs to happen deep within the engine at the *shader* level.

The surface of objects in game engines is generally represented as a polygon mesh, and in the vast majority of cases, those polygons are triangles. Figure 1 shows an example of the internal representation of game objects using a triangular mesh. The shader works on the graphics processing unit (GPU) and renders the mesh structure of the game objects to the screen, based on the set of *vertices*. Each vertex contains the position, the normal vector, and the color of a point on the *surface* of the mesh. Each triangle of vertices can also have an associated two-dimensional image mapped to a plane, known as a *texture*. The more vertices used to represent a surface (the finer the mesh), the higher the quality of the rendering. OpenRelativity involves the real-time computation of each vertex and fragment geometry to simulate the 3D Lorentz transformation of stationary or constant-velocity geometry, time dilation of moving objects, the relativistic Doppler shift of objects moving relative to the camera, searchlight and headlight effects as perceived by a moving camera, and runtime of light effects when events are perceived by the camera. As each vertex is calculated separately, and as there can be millions of vertices in a game scene, computational performance is an issue if frame rates are supposed to be acceptable for a real-time, interactive simulation. The situation is aggravated by the fact that the meshes need to be finer than in non-relativistic games. Since each surface can become highly distorted, it is important that the mesh is fine enough to represent smoothness even if distances between vertices are enlarged.

Figure 2 shows a rendering of a geometric scene for an observer moving to the left. Scenes like these can be used for testing the different effects.

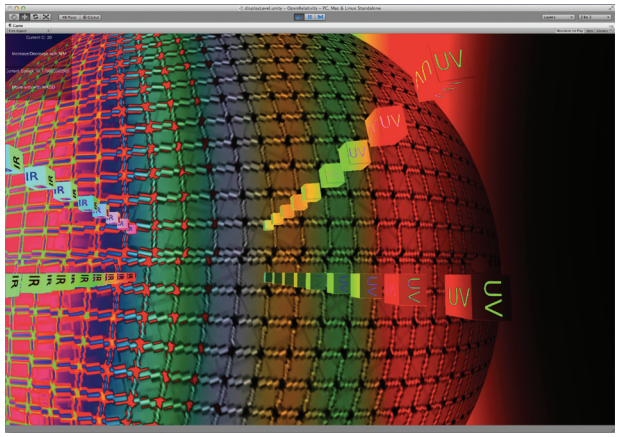


Fig. 2. A screen shot from OpenRelativity. The geometric scene contains a sphere with a gridded texture, as well as a number of cubes lined up. The observer is moving toward the left side. In this setup, the sphere needs to remain spherical, the infrared texture of the cubes on the left should become visible, and so should the ultraviolet texture of the cubes on the right. The searchlight effect makes the right side of the image appear darker than the left side.

OpenRelativity has the following limitations, due to physics or performance reasons:

- Naturally, the player's speed must never reach or exceed the (slower) speed of light.
- Only the player object may move freely. All other objects must either be stationary or have a constant velocity. Due to the lack of a "history buffer"[14] to determine the location of objects at which light signals that reach the player were emitted, our engine is limited to calculating rather than "remembering" the location of objects in the past. For performance reasons, this is only implemented along straight trajectories.
- As a corollary to the above limitation, the player only functions in first-person view; the scene cannot be viewed from any other reference frame.
- Collision detection is limited to collision boxes; meshes are not used in resolving collision between objects. The mesh collision system within the *Unity* game engine is currently not optimized to compute vertex transformations in real-time.
- There is no collision detection for moving scene objects except with the player and specially defined stationary "receiver" objects; calculating the Lorentz-transformed collision boxes for a large number of moving objects can be computationally expensive.

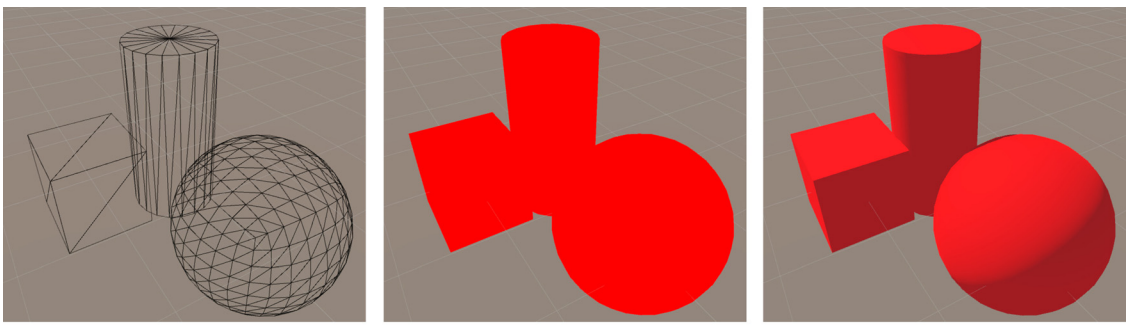


Fig. 1. Example of a polygon wire mesh (left), the same mesh with a solid color (middle), and with burned-in shadows (right).

- Gravity is not modeled in the game. Both this and the earlier restriction of third-party objects moving with constant speed along straight lines means that objects do not follow gravitational trajectories.
- The library does not implement a shadow-casting lighting system, thus, all shadows depicted must be permanent. Any shadow that is present on an object must be part of its texture and assumed to never change. A fully relativistic, dynamic treatment of lighting and shadows is possible but would greatly add to the complexity of the system,[9] and is thus not supported at this time.
- The "infinitely" far away background of the game scenes (the so-called "skybox") needs to be a solid color, e.g., the engine does not currently handle clouds or stars. This limitation is due to the fact the underlying engine does not transform the skybox (however, the Doppler shift and the searchlight effect are treated correctly).

The library is available in source code on GitHub,[16] under the open-source MIT License.[17] The README file at the top-level of the code repository gives a technical overview of the library, and the code has embedded comments with further explanations. We have also implemented one game using this toolset, *A Slower Speed of Light*.[18] Figure 3 shows a screenshot of this game, which is based on a simple object collection paradigm—a task that becomes increasingly more difficult as light slows down.

III. IMPLEMENTATION

Rather than focusing on technical implementation details (see code documentation), Secs. III A–III C give an overview of the employed physics.

A. The player and camera

The "camera" in a video game is a virtual device that "films" what the user sees on the screen. The player object is designed to provide a first-person view of the scene, and thus the camera is permanently attached to the player.

Time passes linearly in this camera-frame C, i.e., the game proceeds along realtime time steps Δt_C, which correspond to the passage of time for the player in front of the screen. The player's velocity can be modified within the game's environment. However, within each time step, the player's movement is assumed to be constant, and the C-frame is assumed to be a momentarily co-moving inertial reference frame.

The implementation of acceleration requires some judgment calls involving how the game should react when the player changes direction and speed. As opposed to non-relativistic games, constant acceleration cannot be assumed. Assuming constant power, i.e., constant increase of kinetic energy per time, would lead to uncontrollable game play at low speeds versus the virtual impossibility of reaching speeds close to the speed of light. As it turns out, it is playable to have fixed player-initiated velocity changes $\Delta\vec{v}_C$ within each time step Δt_C, calculating the new velocity using relativistic velocity addition; this also automatically guarantees that the player will not move faster than the speed of light. Also, because there is no gravity the player needs to be kept on the ground by restricting the player's velocity to be perpendicular to the surface vector.

Internally, within the C-frame, we keep the player at rest at the origin. Instead of moving the player, we are moving the world-frame W underneath the player—the world literally revolves around the player.

B. The scene

The scene is built within the world-frame W. We support the built-in scene editor of *Unity* with the exception of the above mentioned restriction of a solid-color skybox. Collision detection is active for the player with any other object in the scene. Objects in the scene only detect collisions with special "receiver" objects, which destroy the object. A moving object can thus be spawned anywhere in the scene, and moves with a constant velocity until it collides with such a receiver object.

We need to keep track of the position and movement of objects in W, as well as the time passing in this frame. According to our imposed limitations, all vertices i move with constant velocities $\vec{v}_{i,W}$ in the world frame, and are zero for stationary objects. The time passage in W is governed by

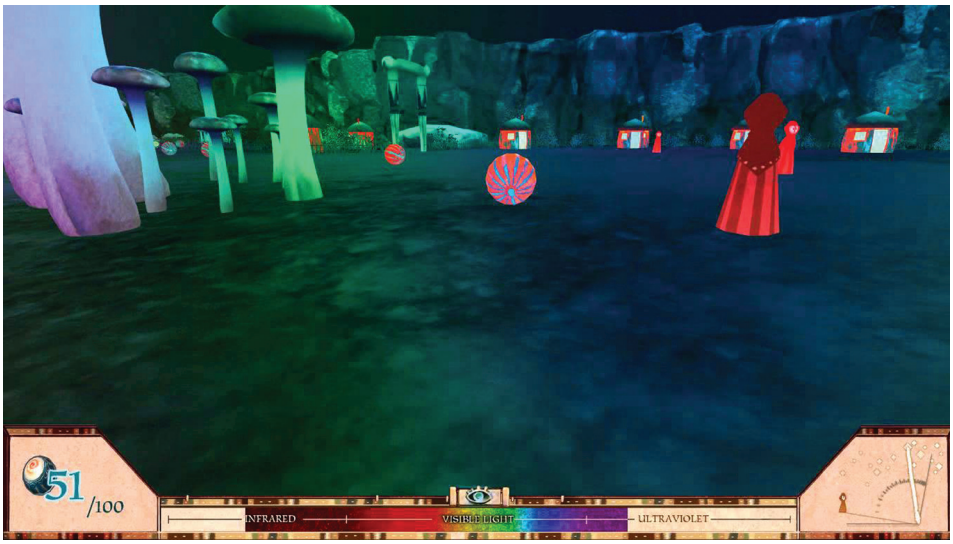


Fig. 3. A screen shot from our game *A Slower Speed of Light*.

$$\Delta t_W = \frac{\Delta t_C}{\sqrt{1 - v_{C,W}^2/c^2}}, \tag{1}$$

where $v_{C,W}$ is the speed of the camera in the W-frame. Thus, the time steps Δt_W are longer than the time steps Δt_C, which underlines the necessity of high frame rates to guarantee smooth gameplay.

The material of objects has the standard Red-Green-Blue (RGB) color component encoding. Two colors were added to represent the infrared and ultraviolet components of the object's emission spectrum, as those could become visible in the C-frame due to the Doppler shift. In the interest of computational speed, we refrained from modeling these spectral components using, for example, blackbody radiation. In effect, we thus have an "IRGBU" color encoding for the object material colors (see Fig. 2 as an example, where all cubes have the letters "IR" and "UV" painted on them in those additional colors).

C. Codes

In order for the scene and its objects to behave relativistically, several codes must be attached to all objects, the skybox, and the player. In particular, we needed to implement the following physics.

1. Lorentz transformation

The Lorentz transformation needs to be applied to all vertices of all third-party (non-player) objects in the scene, or world frame W. In addition, we have the camera frame C, in which the player exists at the origin. In each time step Δt_C, we need to calculate how the scene appears in the C-frame.

The player is at rest at the origin of the C-frame and the scene's W-frame is moved under the player such that within the W-frame the player momentarily moves in the positive z-direction, while within the C-frame the scene momentarily moves in the negative z-direction. Rotating the coordinate axes in this fashion allows us to use the Lorentz transformation formulas in the standard configuration; however, this rotation needs to be carried out in every time step to re-align the coordinate systems C and W along their axes. With $\tilde{R}$ being the rotation matrix, and "$\rightarrow$" denoting the updating between time steps, we use

$$\vec{v}_{C,W} \rightarrow \tilde{R} \cdot \vec{v}_{C,W}, \tag{2}$$

which is now in the z-direction

$$\vec{v}_{i,W} \rightarrow \tilde{R} \cdot \vec{v}_{i,W}, \tag{3}$$

and

$$\vec{r}_{i,W} \rightarrow \tilde{R} \cdot \vec{r}_{i,W}. \tag{4}$$

The positions thus need to be updated in the W-frame using

$$\vec{r}_{i,W} \rightarrow \vec{r}_{i,W} + (\vec{v}_{i,W} - \vec{v}_{C,W})\Delta t_W, \tag{5}$$

which leaves the camera at the origin.

The next challenge is to figure out not only where the objects are located, but also *when* we see them. This calculation would be very hard in the C-frame (inside which we actually "see" them), but fortunately it can be carried out in the W-frame. For the camera to "see" an object, the invariant four-distance needs to be light-like, and due to the invariance we can calculate it in the W-frame. For each vertex, we need to figure out the time $t_{i,S,W}$ and location $\vec{r}_{i,S,W}$ *when* we see it, or the place where it intercepts the surface of the light cone. With the camera constrained to the origin, the time at which the vertex is *seen* is determined by

$$c^2 t_{i,S,W}^2 = \vec{r}_{i,S,W}^2 = (\vec{r}_{i,W} + \vec{v}_{i,W} t_{i,S,W})^2 . \tag{6}$$

From this, the time $t_{i,S,W}$ can be calculated analytically using the quadratic equation. We will get two solutions, and we need to choose the *past* one, which allows us to calculate the location from

$$\vec{r}_{i,S,W} = \vec{r}_{i,W} + \vec{v}_{i,W} t_{i,S,W}, \tag{7}$$

telling us where the object is spotted in the W-frame.

The final step is the application of the Lorentz transformation Λ. For all vertices, we need to calculate

$$\vec{r}_{i,S,C} = \Lambda(\vec{v}_{C,W})\vec{r}_{i,S,W}. \tag{8}$$

As discussed, the distance remains light-like, since it is invariant, and because of the setup we can use the standard configuration for Λ with the boost in the z-direction, having the C and W origins at same position at $t = 0$.

2. Doppler shift and searchlight effect

As already mentioned in Sec. III B, colors in game engines (and elsewhere) are encoded in Red-Green-Blue (RGB) numbers. Unfortunately, those numbers correspond to the three different color receptors in human eyes and their spectral sensitivity. As such, they are a very incomplete representation of the actual color of an object, they simply make it appear correctly to the human eye.[19] RGB values need to be translated into so-called xyz-values; Fig. 4 shows the normed correspondence between these values and wavelengths, including our new UV and IR values. We can use this to construct an approximate full emission spectrum of the object, then shift this spectrum by the appropriate Doppler shift factor

$$D = \frac{1 - (v/c)\cos\theta}{\sqrt{1 - (v/c)^2}}, \tag{9}$$

and

$$\lambda_C = D\lambda_W. \tag{10}$$

In these equations, θ is the angle between the velocities of the source and the observer, and λ_W represents all wavelengths emitted. We then translate λ_C back into new RGB colors that are displayed on the screen.[20] The resulting RGB-values can, unfortunately, be outside the allowed range so we had to cap them at their maximum values.

The only alternative for this approximation would have been to store the full emission spectrum for every vertex in the scene, requiring the storage of millions of fine-grained spectral distributions. Such a procedure would have meant relinquishing the ability to use the built-in scene editor of the

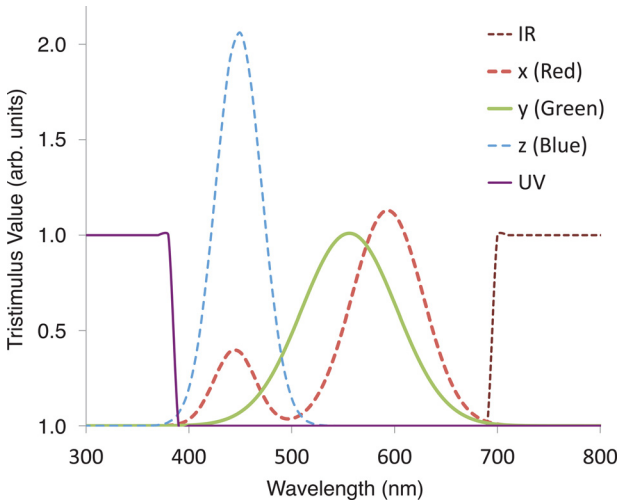


Fig. 4. Stimulus values of the RGB colors.

game engine, as this editor can only handle RGB and our additional IR and UV "paint" (actually internally implemented as grayscale textures).

The searchlight effect (or relativistic aberration) modifies the overall luminosity of the object. With the Doppler shift factor already calculated, this effect decreases the frequency-dependent luminosity by a factor[9] of $1/D^3$. Due to the third power, the onset of the searchlight effect is rather more rapid than that of the Doppler shift, and it can lead to blocking large sections on the rim of the visual field, as well as overexposing the center of vision.

3. Senders and receivers

Third-party moving objects can run with constant velocity (speed and direction) between defined sender and receiver objects. Sender objects and their associated code create moving objects, receiver objects destroy them. In addition, they keep track of the time when these events happened. If the light cone condition determines a time outside the interval in which the object existed, the object is not shown.

There is no animation for creating ("spawning") and destroying these objects; thus, ideally this should happen in a hidden place. For example, in our game the moving villagers are created and destroyed inside of strategically placed huts.

IV. EXPERIENCES AND OUTLOOK

Even with all of the limitations, we were able to create immersive and engaging relativistic scenarios. Unfortunately, it turns out that moving in this environment can quickly induce severe motion sickness. The game has been downloaded over 117,000 times to mostly positive reviews. More than 130 commented "gameplay" videos can be found on YouTube alone,[21] and it would be a worthwhile project that is immediately open to any interested physics education researcher to analyze the narratives of these documented, authentic interactions with the game.

The OpenRelativity library has been forked 42 times in the popular open-source repository GitHub, meaning 42 developers have copied the codebase in order to independently experiment with it. We regularly receive inquiries regarding computational and physics details of the engine, however, to our knowledge no other games or educational software have been developed using our library. We would have hoped that the current game was only the first of a number of games based on this engine. Of particular interest would be "lab experiments" in university physics courses, such as, for example, the pole-barn scenario. We believe that the main hurdle toward development of additional software is the complexity of the *Unity* 3D development platform. While physicists tend to be proficient in computer programming, they usually do so in very different environments. We thus propose a graphical "level editor," in which pre-defined relativistic objects (moving objects in different shapes and colors, senders/receivers, switches, wires, clocks, light emitters and detectors, etc.) can be assembled to generate new educational challenges and puzzles. Each object would already have the correct properties and methods, and relativistic scenarios could be assembled without knowledge of any of the inner workings of the game platform, perhaps even without any knowledge of programming. Popular puzzle games such as Portal have similar level editors as add-ons,[22] and using such a tool both instructors and students could readily construct and modify experimental setups themselves.

We have experimented with OpenRelativity using immersive technology such as *Oculus Rift*,[23] and we are exploring using this library in connection with a *Digistar* projection system[24] in planetarium domes; this would allow the audience to see both in front of and behind a traveling observer and thus illustrate red- and blue-shift within one image. The planetarium project would be of particular interest, as it would allow one to explain the effects of the light runtime on looking into the past of the universe.

V. CONCLUSION

We have implemented a library to modify the shader algorithms of a popular game engine to provide a first-person view of a relativistic world with an arbitrary speed of light. The technology can be used to provide experiences with the otherwise abstract concepts and topics of special relativity. It is our hope that within these human-scale environments, intuition about special relativity can be built. However, thus far no formal studies have been conducted regarding educational effectiveness. Analyzing existing self-narrated gameplay videos might be a first step toward such studies.

We also found that while OpenRelativity provides all necessary functionality to build games and educational software, it may not be sufficiently accessible to developers who usually work within physics computing environments. To remedy this hurdle, a proposed additional editor software may not only sufficiently abstract away the complexity of the underlying game engine, but also make building relativistic environments accessible to both instructors and students.

[a]Present address: Quora, Inc., Mountain View, California 94041.
[b]Electronic mail: kortemey@msu.edu

[1]W. Isaacson, *Einstein— His Life and Universe* (Simon and Schuster, New York, 2007).
[2]A. Einstein, Zur Elektrodynamik Bewegter Körper, Ann. Phys. **322**, 891–921 (1905).
[3]A. Einstein, Über Einen die Erzeugung und Verwandlung des Lichtes Betreffenden Heuristischen Gesichtspunkt, Ann. Phys. **322**, 132–148 (1905).
[4]G. Gamow, *Mr. Tompkins in Paperback* (Cambridge U.P., Cambridge, UK, 2012).
[5]A. Lampa, "Wie erscheint nach der Relativitätstheorie ein bewegter Stab einem ruhenden Beobachter?," Z. Phys. **27**, 138–148 (1924).
[6]R. E. Scherr, "Modeling student thinking: An example from Special Relativity," Am. J. Phys. **75**, 272–280 (2007).

[7]J. Terrell, "Invisibility of the Lorentz contraction," Phys. Rev. **116**, 1041–1045 (1959).
[8]R. Penrose, "The apparent shape of a relativistically moving sphere," in *Math. Proc. Camb. Phil. Soc.* (Cambridge U.P., Cambridge, UK, 1959), pp. 137–139.
[9]D. Weiskopf, Ph.D. thesis, University of Tübingen, 2001.
[10]U. Kraus, H. Ruder, D. Weiskopf, and C. Zahn, "Was Einstein noch nicht sehen konnte," Phys. J. **1**, 77–82 (2002); available at http://www.pro-physik.de/details/articlePdf/1108491/issue.html.
[11]U. Kraus and M. P. Borchers, "Fast lichtschnell durch die Stadt: Visualisierung relativistischer effekte," Phys. unserer Zeit **36**, 64–69 (2005).
[12]C. M. Savage, A. Searle, and L. McCalman, "Real time relativity: Exploratory learning of special relativity," Am. J. Phys. **75**, 791–798 (2007).
[13]D. McGrath, M. Wegener, T. J. McIntyre, C. Savage, and M. Williamson, "Student experiences of virtual reality: A case study in learning special relativity," Am. J. Phys. **78**, 862–868 (2010).
[14]T. Doat, E. Parizot, and J. Vézien, "A carom billiard to understand special relativity," in *Joint Virtual Reality Conference of EuroVR-EGVE* (IEEE Press, 2011), pp. 201–202.
[15]Unity Corporation, *Unity* homepage, <http://unity3d.com/>.
[16]OpenRelativity GitHub repository, <https://github.com/MITGameLab/OpenRelativity/>.
[17]Massachusetts Institute of Technology, MIT License, <http://opensource.org/licenses/MIT>.
[18]G. Kortemeyer, J. Fish, J. Hacker, J. Kienle, A. Kobylarek, M. Sigler, B. Wierenga, R. Cheu, E. Kim, Z. Sherin, S. Sidhu, and P. Tan, "Seeing and experiencing relativity—A new tool for teaching?," Phys. Teach. **51**, 460–461 (2013).
[19]*Commission Internationale de l'Eclairage proceedings, 1931* (Cambridge U.P., Cambridge, UK, 1932).
[20]J. M. Kasson and W. Plouffe, "An analysis of selected computer interchange color spaces," ACM Trans. Graphics (TOG) **11**, 373–405 (1992).
[21]YouTube gameplay videos for A Slower Speed of Light, <https://www.youtube.com/results?search_query=%22slower+speed+of+light%22+gameplay>.
[22]Portal level editor, <https://developer.valvesoftware.com/wiki/Portal_2_Puzzle_Maker>.
[23]Oculus Rift-Virtual Reality Headset for 3D Gaming, <https://www.oculus.com/en-us/>.
[24]Digistar projector, <http://www.es.com/Digistar/>.

Physics Slide Collection

John W. Davis (1883-1966) was a professor of Physics at William Jewell College in Liberty, Missouri from 1913 to 1952. The day before he was due to teach his first class in August 1913, lightning struck the science building and it burned to the ground. Despite this setback, Davis started his course the next day, but without any apparatus. In the short term, he used a copying camera to make a series of 550 slides of physics phenomena and apparatus, borrowing from apparatus catalogues and textbooks. The slides, which are the standard 3.25x4 inch American size, were published by the Keystone view company, along with a set of notes. The slides, and the author's carbon copy of the notes, are now in the Greenslade Collection. (Notes and picture by Thomas B. Greenslade, Jr., Kenyon College)