A Literature Review of Special Relativity in Video Games

Gabriel Faes

Abstract

Synthesised from the literature, we present the fundamentals of special relativity and various specific and practical implementations to introduce special relativity in real-time computer applications, with a focus on video games. Games can implement Lorentz length and time dilation by taking a global inertial frame of reference and applying Lorentz boost on points and checking the intersection of those points with the player's past light cone; this requires storing at least the start and end of life spacetime vectors of inertial objects, and the worldline (or an approximation thereof) of any non-inertial object. We also briefly discuss some relativistic visual effects in the available literature. These can enhance the accuracy of the simulation with effects such as the Doppler shift and the searchlight effect, which change the visible colour of pixels; these effects can, however, be computationally expensive to calculate or detract from the effects of length and time dilation, which are arguably more important for the player.

I. Introduction

Most video games use classical mechanics to simulate physics, render graphics, and determine how the player interacts with the world around them. This is sufficient when all objects are macroscopic and move at non-relativistic speeds. However, to accurately simulate the world, the introduction of quantum mechanics or special relativity is necessary when dealing with microscopic or fast-moving objects (respectively). We deal with the latter case.

Limited work has been done in this specific area. While there are plenty of visual simulations of special relativity, they are reliant on the viewer's frame of reference being inertial, which would severely limit interactions for a video game. However, some simulations lift that restriction and allow the player to move freely (MIT Game Lab, 2012) (TestTubeGames, 2012) (Nakayama & Oda, 2017). These simulations still usually require all other objects to be inertial; this makes the implementation simpler and still allows for plenty of creativity in making a video game but prohibits other objects to move based on player action. One exception to this is with Nakayama & Oda's simulation which allows Non-Playable Characters (NPCs) to move non-inertially. Both implementations (limiting or allowing non-inertial NPCs) will be discussed.

Special relativity, while not exceedingly complicated from a mathematical point of view, can be incredibly unintuitive, especially given the speeds and scales required for its effects to be non-negligible. Games generally avoid implementing special relativity, even when the game takes place in space and could approach those speeds, opting either to ignore it or go for a science-fiction approach rather than a physics-based one.

Games taking place in space, especially over long distances, could benefit from implementing special (and general) relativity, as this can immerse the player further if explained correctly to them. Rather than resorting to fictitious explanations for travel between galaxies taking less time than otherwise necessary, special relativity allows an object to travel long distances in a (perceived) short amount of time.

For representation on more human scales rather than in space, visual simulations almost always lower the speed of light, often as low as a few meters per second (such is the case with "A Slower Speed of Light" by the MIT Game Lab).

Whether the simulation has a lower speed of light or not, the equations and implementation details remain the same.

II. Special Relativity

We introduce the notation and define the terms used throughout as well as the first steps for implementation.

1. Spacetime Vectors

We write the 3-dimensional (3D) space vector of a point (or particle/object) as

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{1}$$

where the components of $\boldsymbol{x}$ are real numbers. This 3D vector can also suitably be used for 2D positions by setting x_3 to zero. Generalising to higher dimensions is possible but, given most video games operate in a 2D or 3D space, we apply the theory of special relativity to such uses directly.

We extend a space vector to a spacetime vector

$$\vec{x} = \begin{bmatrix} x_0 \\ \boldsymbol{x} \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{2}$$

by adding a real component (x_0) for time. In classical mechanics games, one could implicitly view this component as being the same value, at any given point in time, in all vectors representing positions (and can, therefore, simply be omitted in classical implementations).

Wel also let the speed of light $c = 1$ and use the natural units instead of meters and seconds; this means a value of one simultaneously represents both one second and the distance light travels in one second. This becomes useful when calculating a vector's Lorentzian norm.

2. Lorentz Transformation

We can define the Lorentzian inner product of two spacetime vectors as

$$\begin{aligned} \vec{x} \cdot \vec{y} &\coloneqq -x_0 y_0 + \sum_i x_i y_i \\ &= -x_0 y_0 + x_1 y_1 + x_2 y_2 + x_3 y_3. \end{aligned} \tag{3}$$

From this inner product, we may define the Lorentzian norm of a spacetime vector as

$$\|\vec{x}\| = \vec{x} \cdot \vec{x}. \tag{4}$$

Notice that the Lorentzian inner product is not a proper Euclidian inner product as it violates positive definiteness. This means the Lorentzian norm is not a proper Euclidian norm. But this is useful as it allows the classification of spacetime vectors according to their norm: if $\|\vec{x}\| > 0$ then $\vec{x}$ is spatial, if $\|\vec{x}\| = 0$ then $\vec{x}$ is null (light-like), and if $\|\vec{x}\| < 0$ then $\vec{x}$ is temporal (Nakayama & Oda, 2017). We mostly concern ourselves with results where the vector's Lorentzian norm is non-positive.

A transformation $\Delta : \mathbb{R}^4 \to \mathbb{R}^4$ of a spacetime vector is a Lorentz transformation if and only if

$$\forall \vec{x} \in \mathbb{R}^4, \|\vec{x}\| = \|\Delta(\vec{x})\| \tag{5}$$

that is, a Lorentz transformation does not modify a vector's Lorentzian norm (Nakayama & Oda).

This transformation allows for changes in the frame of reference in accordance with the theory of special relativity; two spacetime vectors in the same frame of reference transformed by the same Lorentz transformation now represent the same two spacetime vectors (or events) in a new frame of reference.

3. Frames of Reference

Because of the nature of games, we are mostly concerned with two frames of reference: the player's (referred to as $\boldsymbol{P}_\tau$) which can be different at any given point in time τ experienced by the player, and a stable inertial frame of reference (**W**) (Boffi, Puppin, & Contran, 2024). Any spacetime vector in a different frame of reference will always be converted to the "world's" frame of reference **W**. Only if it is less computationally

expensive to keep the original frame of reference would we not convert to **W**; this is almost never the case as it would require an inertial object or NPC that makes constant decisions based on its position in its own frame of reference; most such objects would be non-inertial.

The frame of reference $\boldsymbol{P}_\tau$ is necessary to correctly display objects to the player. While keeping all spacetime vector in $\boldsymbol{P}$ might seem like the most practical option, $\boldsymbol{P}_\tau$ can move erratically and is therefore non-inertial (the player can accelerate in any direction at any time); it would be prohibitively expensive to recalculate all stored spacetime vectors every frame. This means we will convert all visible objects by the player to $\boldsymbol{P}_\tau$ every frame. Because this can be a computationally expensive operation, we prefer only converting from **W** to be more efficient.

The frame of reference **W** remains the same throughout the simulation. All worldlines will always be converted to **W** and stored as such, including the player's worldline, and reconverted every frame to the necessary frames of reference (notably, $\boldsymbol{P}_\tau$).

4. Worldlines and Proper Time

For our purposes, the proper time τ of a particle is the amount of time which passes according to an object at rest in that frame of reference.

A worldline is a continuous function $w : I \to \mathbb{R}^4$ which parametrizes the motion of a particle (where $I \subset \mathbb{R}$ is the interval representing the lifespan of the particle). A worldline must be constructed from an inertial frame of reference. Worldlines will be stored in **W** so they can later be converted to any frame of reference needed.

To construct an object's worldline in **W**, we need the spacetime vector of the object (in that object's frame of reference) at a given proper time τ_o (this is the object's proper time) and the velocity $\boldsymbol{v}(\tau_o)$ of the object at τ_o seen by **W**. From there, applying Lorentz Boost to the vector gives a vector in **W**. Applying this for all τ_o gives a continuous worldline in **W**.

5. Lorentz Boost

For the sake of simplicity, we limit our case to movement in the x_1 axis. Changing the coordinate system temporarily or extrapolating the equations to all three position axes will allow for free movement in any direction.

Given a spacetime vector $\vec{x}'$ in a frame of reference travelling at speed v along x_1 according to **W**, then

$$\vec{x} = \begin{bmatrix} \gamma\left(x_0' + \frac{vx_1'}{c^2}\right) \\ \gamma(x_1' + vx_0') \\ x_2' \\ x_3' \end{bmatrix} \tag{6}$$

where the vector $\vec{x}$ corresponds to $\vec{x}'$ but in **W**, c is the speed of light and

$$\gamma = \frac{1}{\sqrt{1 - \frac{v^2}{c^2}}}. \tag{7}$$

This transformation is called the Lorentz boost and is a Lorentz Transformation. It is, therefore, physically correct according to the theory of special relativity (OpenStax, n.d.).

6. Past Light Cone

We define the Past Light Cone of a spacetime vector $\vec{y}$ as

$$PLC(\vec{y}) = \{\vec{x} \mid \|\vec{x} - \vec{y}\| = 0, x_0 < y_0\} \tag{7}$$

where x_0 and y_0 are the 0th coordinates of the corresponding spacetime vectors. The spacetime vectors must be in the same frame of reference (we can always convert between frames of reference with equation (6)).

To know when and where an object is visible to the player, we calculate the point of that object's worldline which intersects with the player's past light cone (if any). There can only ever be one such point between the PLC of a temporal vector and a massive object's worldline (Nakayama & Oda, 2017).

III. Implementation Details & Discussion

1. Restrictions on Non-Inertial Objects

The simplest implementation foregoes worldlines entirely by making the speed of light instantaneous (like classical mechanics games) but still applying the effects of special relativity based on the distance light had to travel; this results in inaccurate relativistic effects, as it does not correctly represent the change in causality, but does correctly showcase length contraction and is easy to compute (TestTubeGames, 2012).

Since the worldline of inertial objects is linear in any frame of reference, games can store the starting and ending spacetime positions of any inertial objects instead of the whole worldline. This eases the implementation and computational cost when making a game where all objects except for the player are inertial. All simulations except for the one by Nakayama & Oda in "Relativity for Games" have this restriction; this includes "OpenRelativity", a special relativity tool for the "Unity" game engine which does, however, manage to implement some relativistic visual effects which are briefly discussed in section 3. (MIT Game Lab, 2016).

Work on the implementation of worldlines is limited and more research is required; as previously mentioned, there exist practically no simulations which have no restrictions on object movement. However, keeping the worldlines continuous is likely to be prohibitively expensive as they would need to be reconstructed from the stored acceleration of an object (which are non-constant, complicating reconstruction further) every frame. Instead of keeping continuous worldlines, one may discretize them. This requires both efficient methods to store discrete approximations of the worldline and a dynamic garbage collector.

2. Discrete Worldlines in Unrestricted Simulations

The simplest approach is storing a spacetime vector every fixed proper timestep and linearly interpolating between missing points. This is the method used by "Relativity for Games" where every object except for the player has their worldline updated up until the player's current proper time (Nakayama & Oda, 2017).

This approach may store vastly more vectors than necessary and require more computation to calculate a worldline's intersection with the player's Past Light Cone. While more research for the discretization of worldlines is necessary, we propose here a method to limit the number of vectors in the worldline. When adding a vector $\overrightarrow{x_n}$ to the worldline (the nth vector for that worldline), given some small ε, one may remove $\overrightarrow{x_{n-1}}$ from the worldline if

$$\exists t \in [0,1] \subset \mathbb{R}, \text{such that} \\ \|\overrightarrow{x_{n-1}} - (t\overrightarrow{x_n} + (1-t)\overrightarrow{x_{n-2}})\| < \varepsilon \qquad (8)$$

that is, the linear interpolation between $\overrightarrow{x_n}$ and $\overrightarrow{x_{n-2}}$ can already result in approximately $\overrightarrow{x_{n-1}}$.

Additionally, vectors which are no longer required need to be garbage collected, else the worldlines will grow indefinitely for as long as the program runs and will quickly cause the program to run out of memory since all objects have a worldline.

3. Relativistic Visual Effects

"OpenRelativity" implements visual effects aimed at correctly portraying what one would view when travelling near the speed of light, namely the Doppler shift and the searchlight effect (MIT Game Lab, 2016).

The Doppler effect causes the wavelength of light travelling towards the player to change based on the relativistic difference in speed between the source and the observer, the factor of which can be calculated using equation (7) (OpenStax, n.d.). This also causes light which would have been imperceptible to humans (such as UV or Gamma rays) to become visible due to their wavelength shifting to the visible part of the

spectrum. The searchlight effect is similar and causes more (or less) photons to be perceived by an observer travelling towards (or away) from a source, and therefore making an object appear brighter (or dimmer). These effects can be implemented efficiently on the Graphical Processing Unit (GPU) via a pixel shader.

However, relativistic visual effects can be distracting to the player as they can make it much harder to visualize regular objects (for example, a player moving away from an object near the speed of light will be unable to see that object as they are receiving very few photons from that object). Similarly, the Doppler shift can cause objects to exhibit colours the player is not used to, making special relativity even more unintuitive (MIT Game Lab, 2012).

Video games which require the player to be aware of their surroundings, such as enemies or walls, need the player to be able to distinguish those objects regardless of their speed (unless it is a deliberate design decision for those objects to be difficult to see). As such, limiting the effects of the Doppler shift and searchlight effect can be a viable alternative instead of not implementing them at all. This can be achieved by transforming the multiplication factor with a fixed function (for example, $\max(\log(\gamma), a)$ instead of γ, which would significantly reduce and cap the effects to a factor of a). This allows the video game to still exhibit physics-realistic phenomena, without them overwhelming the player.

IV. Conclusion

Lorentz transformations are the basis of special relativity. Video games may implement Lorentz transformations in their games in multiple ways, each with different drawbacks and advantages. The most important choice when making such a simulation is deciding which objects may accelerate.

Video games which restrict the acceleration of objects can keep the implementation of worldlines simple by linearly interpolating between the start and end spacetime vectors of an object. Lifting the movement restriction requires implementing discrete worldlines or an equivalent solution; such visual simulations are limited, and most games or tools implementing special relativity have restrictions for all frames of reference to be inertial except that of the player. If discrete worldlines are implemented, special care needs to be taken to efficiently store and then delete vectors, such as by removing vectors which can already be reconstructed from the linear interpolation of its two neighbours on the worldline.

Video games may also have relativistic visual effects, but these can easily distract the player or make it confusing for them to evolve in the environment, even further than Lorentz transformations already do. Developers should consider whether their implementation would add realism, and if so, whether dimming their effects would be suitable to avoid the player's disorientation being magnified.

More research is needed to efficiently implement special relativity in games, more specifically worldlines and their intersection with the player's Past Light Cone. Additional simulations which allow for other actors other than the player to accelerate to near the speed of light should be developed and studied. Additionally, other rendering methods, such as relativistic raytracing, could be further studied to ascertain whether they are feasible.

References

Boffi, A., Puppin, E., & Contran, M. (2024, July 20). *VirtualRelativity: An Interactive Simulation of the Special Theory of Relativity in Virtual Reality*. Retrieved April 2026, from arXiv: https://arxiv.org/abs/2408.01442

Carr, D. (2010). Visual Computer Game Features for Teaching Relativity. *2010 Seventh International Conference on Computer Graphics, Imaging and Visualization* (pp. 35-40). Sydney, Australia: IEEE. doi:10.1109/CGIV.2010.13

Carr, D. N., Bossomaier, T., & Lodge, K. (2007). Designing a Computer Game to Teach Einstein's Theory of Relativity. *Computer Graphics, Imaging and Visualisation (CGIV 2007)* (pp. 109-114). Bangkok, Thailand: IEEE. doi:10.1109/CGIV.2007.35

MIT Game Lab. (2012). *A Slower Speed of Light*. Retrieved April 2026, from MIT Game Lab: https://gamelab.mit.edu/games/a-slower-speed-of-light/

MIT Game Lab. (2016). *OpenRelativity*. Retrieved April 2026, from MIT Game Lab: https://gamelab.mit.edu/research/openrelativity/

Nakayama, D., & Oda, K.-y. (2017, November). Relativity for games. *Progress of Theoretical and Experimental Physics, 2017*(11). Retrieved April 2026, from https://doi.org/10.1093/ptep/ptx127

OpenStax. (n.d.). *Doppler Effect for Light*. Retrieved April 2026, from LibreTexts Physics: https://phys.libretexts.org/Bookshelves/University_Physics/University_Physics_(OpenStax)/University_Physics_III_-_Optics_and_Modern_Physics_(OpenStax)/05%3A__Relativity/5.08%3A_Doppler_Effect_for_Light

OpenStax. (n.d.). *The Lorentz Transformation*. Retrieved April 2026, from LibreTexts Physics: https://phys.libretexts.org/Bookshelves/University_Physics/University_Physics_(OpenStax)/University_Physics_III_-_Optics_and_Modern_Physics_(OpenStax)/05%3A__Relativity/5.06%3A_The_Lorentz_Transformation

TestTubeGames. (2012). *Velocity Raptor: An adventure in 2+1 dimensions*. Retrieved April 2026, from TestTubeGames: https://www.testtubegames.com/srel101.html