Architectural Design of a TypeScript Physics Engine for Speculative Cosmology and Advanced Propulsion Dynamics

The development of a custom simulation environment and computational physics engine in TypeScript requires an exquisite equilibrium between rendering performance, memory management, numerical stability, and profound architectural flexibility. While contemporary, highly optimized JavaScript and TypeScript physics engines—such as Matter.js, Planck.js, and Rapier (which effectively leverages WebAssembly for performance gains)—dominate the landscape of standard 2D and 3D rigid-body simulations, their architectures are fundamentally bound to classical Newtonian mechanics and absolute Galilean relativity.[1] These standard engines operate under rigid foundational axioms: they presume that mass is an invariant scalar under all forms of acceleration, that space acts as an absolute and immovable background grid, and that the speed of light is either completely ignored or treated as a universal, unchanging constant. However, demonstrating speculative cosmology and advanced theoretical propulsion systems—specifically encompassing Tired Light (TL) mechanics, Variable Speed of Light (VSL) frameworks, Covarying Coupling Constants (CCC+TL), and the highly theoretical Dark Matter Drive—demands a radical, foundational departure from classical mechanics.[5]

To accurately model and simulate a digital universe where light autonomously exhausts its kinetic energy over immense cosmological epochs, where mechanical inertia is generated dynamically through relational Machian interactions, and where faster-than-light (FTL) transit is engineered by harvesting macroscopic Bose-Einstein condensates composed of massive photons, the entire computational physics engine must be engineered from scratch.[5] Standard third-party libraries lack the modularity to dynamically alter the fundamental constants of nature during runtime.[7] This exhaustive report delineates the complete architectural design of a specialized TypeScript physics engine tailored to demonstrate these exact speculative phenomena, detailing the core software engineering principles, the integration of non-standard kinematic algorithms, and the quantum optical subsystems required for advanced propulsion modeling.

Core Software Architecture and Memory Optimization in TypeScript

Before addressing the complex mathematics of variable speed of light or relational inertia, the foundational software architecture of the TypeScript engine must be established to handle

macroscopic computational loads. A standard object-oriented approach, where every particle and photon is a distinct class instance containing its own update methods, inevitably leads to severe performance degradation in V8 JavaScript engines due to cache misses and aggressive garbage collection (GC) pauses.[9]

To circumvent these computational bottlenecks, the engine is structured using stringent Data-Oriented Design (DOD) principles. Rather than an Array of Structures (AoS), the architecture employs a Structure of Arrays (SoA) methodology. The kinematic properties of theoretical entities—such as PhotonEntity and ParticleEntity—are decoupled from their conceptual classes and stored in contiguous, pre-allocated Float64Array buffers. By utilizing typed arrays, the engine ensures that spatial coordinates, velocities, variable masses, and localized physical constants remain sequentially packed in system memory. When the core simulation loop executes, it iterates over these flat memory arrays, maximizing CPU cache coherency and entirely eliminating GC pressure during the millions of calculations required per frame.[5]

Furthermore, the physical simulation loop is strictly segregated from the visual representation layer. The computational engine operates within a dedicated Web Worker, utilizing SharedArrayBuffer mechanisms to pass updated tensor states and positional data to the main thread. A distinct VisualizationGrapher hooks into these shared arrays in a read-only capacity, translating the physical data into graphical plots or rendering them via WebGL and Three.js.[1] This decoupling guarantees that the numerical integrity of the highly sensitive cosmological integrations is never compromised by front-end rendering lag or frame-rate fluctuations.

Multi-Scale Time-Stepping Mechanisms

Simulating physics that ranges from localized quantum optical interactions to the 15-billion-year evolution of the cosmos necessitates a highly specialized time-stepping architecture.[5] The engine cannot rely on the standard requestAnimationFrame delta-time for physical updates, as browser inconsistencies would rapidly introduce floating-point drift. Instead, the SimulationOrchestrator module utilizes a fixed time-step accumulator for micro-scale interactions alongside a multi-scale temporal mapping system.

An absolute macroscopic variable, t_{cosmic}, tracks the evolutionary timeline of the simulated universe, governing the slow decay of fundamental constants. Conversely, a high-frequency variable, t_{local}, processes immediate interactions such as photon-electron scattering and the rapid localized dynamics of the Ramscoop vortex.[5] By maintaining distinct temporal scales, the engine seamlessly scales its integration precision based on the proximity and velocity of the entities being observed by the visualization camera.

Advanced Numerical Integration: Adaptive Runge-Kutta Formulations

In standard commercial game engines, the semi-implicit Euler integration method is the industry standard due to its computational cheapness, ease of implementation, and relative

stability in resolving resting contacts and stacking objects.[4] However, semi-implicit Euler introduces significant energy drift when dealing with complex, non-linear orbital mechanics or rapidly decaying physical constants over extended periods.

For the speculative cosmology engine, simulating interconnected differential equations—such as the continuous exponential decay of the speed of light or the covariant fluctuation of the universal gravitational constant—demands vastly superior numerical precision to prevent catastrophic calculation drift.[5] Therefore, the CorePhysicsEngine eschews Euler methods entirely in favor of a dynamically adaptive 4th-Order Runge-Kutta (RK4) integration algorithm.[5]

The state vector $\mathbf{y}(t)$ of any given entity—encompassing its Cartesian position, velocity, dynamically varying mass, and its localized experienced speed of light—is updated by evaluating four distinct derivatives at various points across the time-step h:

$$\mathbf{k}_1 = \mathbf{f}(t, \mathbf{y})$$
$$\mathbf{k}_2 = \mathbf{f}(t + \frac{h}{2}, \mathbf{y} + h\frac{\mathbf{k}_1}{2})$$
$$\mathbf{k}_3 = \mathbf{f}(t + \frac{h}{2}, \mathbf{y} + h\frac{\mathbf{k}_2}{2})$$
$$\mathbf{k}_4 = \mathbf{f}(t + h, \mathbf{y} + h\mathbf{k}_3)$$
$$\mathbf{y}(t + h) = \mathbf{y}(t) + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$$

To implement this efficiently in TypeScript without allocating new array objects per derivative, the engine pre-allocates auxiliary Float64Array spaces specifically for the intermediate $\mathbf{k}_1$ through $\mathbf{k}_4$ vectors. This adaptive RK4 integrator acts as the computational heartbeat of the simulation, ensuring that non-linear variables follow their mathematically predicted trajectories flawlessly, even when simulated over billions of virtual years.

Collision Detection and High-Resolution Kinematics

Despite the macroscopic scale of the simulation, robust collision detection remains a critical requirement, particularly for modeling the spacecraft harvesting dark matter particles via the macroscopic scoop field or evaluating the physical trajectories of celestial bodies.[5] The collision pipeline is rigorously divided into distinct broad-phase and narrow-phase algorithms to maintain computational viability.[10]

Broad-Phase Spatial Partitioning

Given the vast scale of the simulation grid, executing $O(N^2)$ collision checks per frame would instantly halt execution. The engine utilizes a dynamic Bounding Volume Hierarchy (BVH) structured as a highly optimized spatial tree.[7] As entities navigate the static Euclidean void, the BVH rapidly culls non-colliding pairs, reducing the algorithmic complexity to $O(N \log N)$.

For regions of extreme density—such as the interior of the Ramscoop vortex where dark matter quanta are highly compressed—the engine dynamically switches to a spatial hashing

grid, which provides faster insert and lookup times for uniform-sized entities tightly clustered in space.[5]

Narrow-Phase Algorithms: SAT, GJK, and EPA

Once the broad-phase identifies potential intersections, the narrow-phase extracts precise contact manifolds. For strictly 2D projections or simplified cross-sectional analyses, the engine deploys the Separating Axis Theorem (SAT), which perfectly handles arbitrary convex polygons.[10] However, to simulate complex 3D topological intersections—such as a spacecraft geometry navigating through a dense swarm of Proca-mass photons—the engine implements the Gilbert-Johnson-Keerthi (GJK) algorithm paired with the Expanding Polytope Algorithm (EPA).[8]

The GJK algorithm relies on calculating the Minkowski Difference of two convex shapes; if the spatial origin is contained within the resulting simplex, a collision is mathematically guaranteed.[10] The algorithm relies on a highly efficient support() function, which simply returns the point on the surface of the shape furthest away from the center in a given direction, eliminating the need to explicitly check every vertex.[10]

While GJK efficiently provides a binary boolean indicating an intersection, it does not provide the penetration depth or contact normal necessary for physical resolution. To extract this, the engine passes the leftover simplex data to the EPA algorithm.[10] EPA iteratively expands the simplex outward until it finds the face closest to the origin, mathematically extracting the collision normal and penetration depth.[10] Once these contact manifolds are securely generated, the CollisionResolver module applies impulse-based sequential impulses, calculating restitution and friction to resolve the physical penetration without introducing artificial kinetic energy into the system.[4]

Cosmological Metrics: VSL and Tired Light Implementations

The standard cosmological model (ΛCDM) inherently relies on the concept of metric expansion—where space itself stretches over time—to explain the observed cosmological redshift. The custom TypeScript engine discards metric expansion entirely. Instead, it models the universe as a strictly static, continuous Euclidean void governed by a centralized ConstantsManager class that dynamically decays fundamental parameters over macroscopic time scales.[5]

Variable Speed of Light (VSL) Frameworks

The simulation architecture acts as a virtual laboratory, engineered to support interchangeable algorithmic modules for the kinematic decay of the speed of light (c).[5]

The primary implementation is the Exponential Decay Model, rooted deeply in Giuseppe

Pipino's VSLT (Variable Speed of Light with Time) framework.[5] Under this model, the ConstantsManager dictates that the speed of light decays continuously according to a differential equation where the rate of change is proportional to the universal Hubble constant (H_0):

$$\frac{dc}{dt} = -H_0 c(t)$$

Through integration, the engine evaluates this as an exponential decay function: $c(t) = c_0 e^{-H_0 t}$.[5] Consequently, the cosmological redshift (z) generated by the simulation is explicitly not modeled as a Doppler shift or a consequence of space expanding. Instead, redshift is an emergent kinematic property derived directly from the temporal variation of light's velocity. The formula hardcoded into the VisualizationGrapher calculates redshift directly from the speed delta: $z = \frac{c_0 - c(t)}{c(t)}$.[5] This mechanism effectively synthesizes standard redshift observations without requiring the universe to physically expand.[11]

To ensure theoretical flexibility, the engine also supports an alternate framework: the Inverse Proportionality Model proposed by Alfonso-Faus.[5] In this module, the speed of light decreases inversely proportional to cosmological time: $c(t) = c_0 \left(\frac{t_0}{t}\right)$.[5] Visually, this manifests on the graphical output as a sharp hyperbolic curve that descends asymptotically toward zero over the 15-billion-year timeline.[5]

Energetics: Tired Light and Intergalactic Scattering

Simultaneous to the kinematic decay of velocity, the engine must simulate the continuous drain of photon energy, fulfilling the requirements of Tired Light (TL) cosmology.[5] This energy attenuation is simulated through two concurrent routines to balance computational efficiency with microscopic realism.

First, a continuous exponential energy decay is applied to all active light. Photon energy attenuation is governed by an energy attenuation coefficient (μ) acting over the total distance (r) traversed. However, because c is dynamically changing in this engine, distance cannot be a simple scalar multiplication; it must be continually calculated via the integral of the variable speed of light: $r = \int c(t)dt$.[5] The decay formula is thus expressed as $E(r) = E_0 e^{-\mu r}$.[5] By linking the traversed distance to the decaying variable speed, the attenuation time constant within the code dynamically shifts to $\tau_0 = \frac{1}{\mu c(t)}$, culminating in the final energy equation: $E(t) = E_0 e^{-t/\tau_0}$.[5]

To achieve a higher degree of microscopic realism, the engine runs a secondary Free Electron Compton (FEC) scattering subroutine.[5] Rather than treating energy loss purely as a smooth, continuous mathematical curve, this subroutine tracks highly discrete, physical collision events

between propagating photons and free electrons suspended in a simulated intergalactic plasma. The volume is seeded with an average electron number density of $n_e = 0.01\ \mathrm{m}^{-3}$, aligning with New Tired Light predictive models.[5] Each time an intersection is detected by the narrow-phase collision pipeline, the engine calculates the energy transfer during absorption and re-emission. This discrete interaction results in a localized frequency reduction, generating a Dispersion Measure (DM) that directly scales with the synthetic redshift: $DM \approx \int n_e dr \propto z$.[5]

The Covarying Unified Model: CCC+TL and Thermodynamic Conservation

To successfully emulate the visual illusion of an expanding universe—and to naturally replicate the effects typically attributed to dark matter and dark energy without introducing a cosmological constant (Λ)—the engine implements the CCC+TL (Covarying Coupling Constants plus Tired Light) framework popularized by Rajendra Gupta.[5]

Correlating Universal Constants

The CCC+TL framework posits that constants do not decay in isolation. In the simulation, the ConstantsManager is programmed to dynamically alter the universal gravitational constant (G) in tandem with the speed of light (c).[5] This interrelated variation alters the Friedmann equations driving the simulation, generating terms that mathematically mimic the repulsive effects of dark energy and the attractive anomalies of dark matter.[6] The model has been statistically tested against Pantheon+ supernovae data and Baryon Acoustic Oscillation (BAO) features, significantly reducing cosmological tension without requiring ΛCDM expansions.[6]

The Variable Mass Hypothesis and Energy Conservation

A severe architectural and physical challenge in designing a VSL physics engine is maintaining the absolute integrity of the First Law of Thermodynamics.[5] In classical physics, the equivalence principle $E = mc^2$ is paramount. If the engine algorithmically decays the speed of light c to a fraction of its original value over billions of years, evaluating $E = mc^2$ directly would result in the simulated universe catastrophically losing massive quantities of rest energy, violating thermodynamic conservation.

To explicitly prevent this fatal flaw, the TypeScript architecture enforces the Variable Mass Hypothesis within the ParticleEntity class structure.[5] The rest mass (m) of every simulated particle is dynamically calculated inversely proportional to the square of the decaying speed of

light:

$$m(t) = m_0 \left(\frac{c_0}{c(t)} \right)^2$$ [5]

By substituting this dynamically scaling mass into the foundational energy equivalence formula, the engine ensures stability:

$$E = m(t)c(t)^2 = \left[m_0 \left(\frac{c_0}{c(t)} \right)^2 \right] c(t)^2 = m_0 c_0^2$$.[5]

The c^2 terms perfectly cancel out in the mathematics. Consequently, the rest mass of all baryonic particles in the simulation proportionally increases as the speed of light slows down. This ingenious software architecture guarantees that the total energy of the simulated universe remains a perfectly invariant scalar across the entire 15-billion-year epoch, verifying thermodynamic conservation through rigorous computational symmetry.

Massive Electrodynamics: Integrating the Proca Formalism

Classical game physics engines invariably operate under the assumption that photons are inherently massless corpuscles, conforming strictly to standard Maxwellian electrodynamics. The custom TypeScript engine categorically discards this assumption to accurately model the dark matter substrate required for the propulsion mechanics.[5] The physics engine transitions from standard electrodynamics into the domain of massive electromagnetism, utilizing the de Broglie-Proca equations and Stueckelberg formalisms.[5]

Instantiating the Massive Vector Field

Within the codebase, the PhotonEntity constructor explicitly assigns a definitive, non-zero invariant rest mass (m_γ) to every light particle. This mass is established near the current experimental upper bound of roughly 10^{-51} grams.[5] While this parameter is incredibly minute, its inclusion fundamentally alters the engine's wave mechanics.

By introducing a mass term into the electromagnetic field tensor—represented by the Lagrangian $\mathcal{L}_{\text{Proca}} = -\frac{1}{4}F_{\mu\nu}F^{\mu\nu} + \frac{1}{2}m_\gamma^2 A_\mu A^\mu$ —the engine mathematically breaks standard $U(1)$ gauge invariance.[5] However, the code rigorously maintains Lorentz covariance, treating the photon as a massive vector field A_μ.[5] To align with alternative symmetries, the engine can be toggle-configured to apply Chern-Simons theory in odd dimensions, allowing the photon to acquire mass without permanently breaking gauge symmetry computationally.[5]

Computational Vacuum Dispersion and Phase Velocities

The most profound phenomenological consequence rendered by the MetricTensorSolver under Proca electrodynamics is strict vacuum dispersion.[5] In this simulated universe, the universal constant c ceases to function as an impenetrable geometric barrier or an absolute propagation speed limit; instead, it is reclassified merely as the localized phase velocity limit of the underlying electromagnetic substrate.[5]

The MetricTensorSolver forces propagation speeds to become strictly wavelength-dependent.[5] The engine dynamically calculates both the group velocity (v_g) and the phase velocity (v_p) of propagating electromagnetic waves based on their localized frequency (ω):

$$v_g = c\sqrt{1 - \frac{m_\gamma^2 c^4}{\hbar^2 \omega^2}}$$

$$v_p = \frac{c}{\sqrt{1 - \frac{m_\gamma^2 c^4}{\hbar^2 \omega^2}}}$$

Proca Massive Electrodynamics Variables	Physical Implementation in Simulation Code
Photon Mass Energy (E_{ph})	Calculated natively as $E_{ph} =$ [21]
Relativistic Photon Velocity (v_{ph})	Calculated dynamically as $v_{ph} =$ [21]
Group Velocity (v_g)	Wave packet propagation speed ($v_g =$) [21]
Photon Mass Wavelength (λ_{ph})	Derived inversely from mass: $\lambda_{ph} =$ [21]

This frequency-dependent drag calculation is computationally heavy but physically vital. It ensures that high-energy, high-frequency active photons experience negligible mass-induced drag, traveling at velocities approaching c. Conversely, lower-frequency photons—specifically those that have been structurally exhausted by the Tired Light energy attenuation routines—experience profound intrinsic structural resistance, forcing their group velocity to drop significantly below the luminal limit.[5]

Relational Mechanics and Machian Inertia Formulation

Standard propulsion modeling inside a physics engine dictates that force is applied against an

object's fixed mass to generate acceleration, derived directly from Newton's $\mathbf{F} = m\mathbf{a}$. Furthermore, classic relativistic frameworks impose an absolute limit on velocity by artificially inflating the object's "relativistic mass" as it approaches c, asymptotically demanding infinite energy for further acceleration.[5] To simulate the Dark Matter Drive, the engine must systematically dismantle both of these paradigms.[5]

The TypeScript architecture adopts a stringent axiomatic foundation built exclusively on Relational Mechanics and the rigorous implementation of Mach's Principle.[5] Heavily influenced by the mathematical models of André Koch Torres Assis, the engine completely rejects the concept of a background absolute space.[5]

Dynamic Emergence of Inertia

In this relational framework, the structural mass of the spacecraft entity (m_0) remains an invariant Lorentz scalar.[5] It does not mathematically inflate as velocity increases, entirely eliminating the "relativistic mass" barrier that classically prevents FTL transit.[5]

Instead, inertia is engineered not as an intrinsic property of the vessel, but as a dynamic, emergent scalar field. The RelationalMechanicsSolver calculates inertial resistance based exclusively on the gravitational and electromagnetic interactions between the local vessel and all other massive bodies distributed across the observable cosmic grid.[5] The engine applies a force law based directly on Weber gravitation, which determines force as a function of relative distance, relative velocity, and relative acceleration between bodies.[5]

Because evaluating Weber force interactions between the spacecraft and every single particle in the simulated universe would trigger an $O(N^2)$ computational collapse, the engine implements a "cosmic background shell" approximation. The inertia of the spacecraft is dynamically determined by integrating the Weber forces against a procedurally generated, isotropic far-field mass distribution representing distant galaxies.[5]

Frictionless Transit in Deep Cosmic Voids

This relational formulation yields a spectacular physical phenomena directly leveraged by the propulsion systems: frictionless transit.[5] Because inertial resistance is dynamically generated by ambient mass concentrations, when a spacecraft entity navigates away from dense galactic clusters and deep into intergalactic voids, the gravitational pull from the cosmic background becomes entirely uniform and highly attenuated.[5]

The physics engine calculates a near-total collapse of inertial resistance in these deep voids.[5] The structural resistance to acceleration drops precipitously toward zero. This mathematical reality transforms the vast emptiness of the digital cosmos from an insurmountable navigation challenge into a functional prerequisite for achieving frictionless, superluminal acceleration.[5]

Simulating the Dark Matter Drive: Substrate Freeze-out and Quantum Harvesting

The synthesis of Tired Light energetics, Proca electrodynamics, and relational mechanics provides the physical framework for the Dark Matter Drive, specifically utilizing the ArcSecs framework parameters.[5] The physics engine does not model dark matter as theoretical Weakly Interacting Massive Particles (WIMPs) or axions. Instead, the engine physically simulates dark matter as a vast, highly attenuated macroscopic Bose-Einstein Condensate (BEC) composed entirely of ancient, massive "tired photons".[5]

Thermodynamic Freeze-Out State Machine

As the SimulationOrchestrator runs its chronological loops, active high-frequency light gradually loses energy via the FEC scattering and exponential decay algorithms.[5] As the frequency plummets, the Proca vacuum dispersion mechanism rapidly degrades the photon's group velocity.[5]

The engine monitors the kinetic energy of every PhotonEntity. When this value drops below a hardcoded thermodynamic threshold, the engine triggers a "cosmic freeze-out" event.[5] The entity's state machine transitions from active radiation into a cold, non-relativistic, sub-luminal bound state referred to as "graviballs" or "slow quanta".[5]

Thermodynamic State Transition	Active Electromagnetic Light	Sub-luminal "Tired Light" (Dark Matter)
Energy Localization	Dominated by relativistic kinetic energy ($\hbar\omega$) [5]	Almost entirely localized in invariant rest mass (m_γ) [5]
Propagation Velocity	Localized phase velocity limit near c [5]	Sub-luminal, continually decaying to near-zero [5]
Physical Interaction Mode	Standard EM irradiation, momentum transfer [5]	Strictly gravitational, optically invisible [5]
Cosmological Function	Energy transfer, cosmic illumination [5]	Forms massive, dense galactic halos (Fuel Substrate) [5]

These dark matter quanta retain their invariant rest mass but lack the kinetic energy to trigger standard atomic electron transitions, making them strictly optically invisible to standard

detection arrays within the simulation.[5] However, because they possess rest mass, the RelationalMechanicsSolver ensures they continuously pool into galactic gravitational wells over millions of iterations, forming the dense halos that the spacecraft requires for fuel.[5]

Macroscopic EIT and the Ramscoop Vortex

To achieve propulsion in deep voids where structural inertia is effectively zero, the spacecraft must ingest and expel this dark matter substrate. However, the substrate is highly diffuse in these voids, presenting a severe volumetric scarcity challenge.[5]

The TypeScript engine implements a complex quantum optics subsystem to solve this. The spacecraft entity mathematically projects an Electromagnetically Induced Transparency (EIT) scoop field up to 4,000 kilometers ahead of its bow.[5] Within the simulation grid, this is modeled as a massive, localized alteration of the spatial refractive index arrays.

The MetricTensorSolver utilizes Kramers-Kronig relations, meaning a sharp change in optical absorption within the tailored EIT window induces a steep, positive gradient in the real part of the localized refractive index.[5] As the drifting dark matter quanta intersect this manipulated gradient, their group velocity is violently decelerated to near true zero.[5] This extreme deceleration triggers a localized spatial compression cascade. The sparse wave packets, originally stretching for kilometers, are forcefully compressed down to mere micrometers.[5]

The physics engine models this dynamically compressed area as a highly coherent "Ramscoop vortex form".[5] Acting as a massive, frictionless fluid-dynamic funnel, the 4,000-kilometer cross-section gathers the sparse substrate, multiplies its volumetric density exponentially, and guides the dense fuel stream directly into the vessel's compact 1.2-kilometer physical intake throat.[5]

Overcoming the Stationary Light Paradox with DSPs

A critical error occurs in classical simulation models when bringing light to a halt: if a massless photon is brought to absolute rest, its momentum drops to zero, and according to $E = mc^2$, its energy ceases to exist.[5] This violates absolute conservation of energy. Navigating a spacecraft through stationary light would typically imply colliding with the dense atomic cloud required to stop the light, resulting in immediate structural destruction due to massive baryonic drag.[5]

Because the simulation uses massive Proca photons and advanced quantum optics, it bypasses this paradox entirely through the mathematical application of Dark-State Polaritons (DSPs) and Stationary Light Pulses (SLPs).[5] The macroscopic EIT field is modeled as a three-level Λ-type quantum system, featuring two stable ground states and one excited state.[5]

The engine's quantum subsystem applies forward and backward counter-propagating control beams—simulated as highly coherent two-color stationary light pulses.[5] This establishes a dynamic photonic Bragg grating directly within the medium.[5] The targeted dark matter

massive photons reflect back and forth over microscopic distances via Bragg scattering, effectively freezing the light in localized space.[5]

Crucially, the electromagnetic energy is not destroyed. The SimulationOrchestrator dictates that the coupled light-matter Hamiltonian transitions the state vector into a linear superposition of the electromagnetic field and the atomic spin coherence operator.[5] The energy is safely mapped onto the medium by forming Dark-State Polaritons.[5] The macroscopic reduction in kinetic energy is perfectly compensated by a conservative increase in the internal potential energy of the DSP quasiparticles.[5] This ultra-dense, energized matter-light hybrid is then violently expelled by the engine's propulsion routines to generate infinite relational thrust without encountering the catastrophic baryonic drag of a traditional medium.[5]

System Integration: The Test-Driven Development (TDD) Matrix

Given the highly speculative, interconnected mathematics governing this architecture, relying on visual debugging or standard physics engine unit tests is wholly insufficient. A specialized, rigorous Test-Driven Development (TDD) matrix is hardcoded directly into the TypeScript repository to assert that these complex modifications to general relativity, electrodynamics, and quantum optics do not yield internal logical contradictions or violate the First Law of Thermodynamics.[5]

The SimulationOrchestrator executes comprehensive unit and integration tests every time the core engine initializes, halting execution with a fatal error stack if any assertion is breached.

TDD Integration Suite	Software Module	Speculative Hypothesis Tested	Algorithmic Assertion Condition
VSL_Kinematic_Decay	ConstantsManager	Exponential decay of c over absolute time	$c(t) <$ AND $c(t) >$ [5]
Energy_Attenuation	PhotonEntity	Continuous energy loss via Tired Light	$E(t) <$ [5]
Covariant_Mass	ParticleEntity	Thermodynamic energy conservation via	$m(t) \cdot c(t)^2 =$ [5]

		VSL	
Dimensionless_Stability	ConstantsManager	Invariance of the Fine Structure Constant	$\alpha(t) =$ [5]
Kinematic_TimeDilation	MetricTensorSolver	Pulse stretching strictly via c variation	$\Delta t_o = \Delta t_e(1+z)$ [5]
Proca_Frequency_Drag	MetricTensorSolver	Vacuum dispersion via massive photon	$v(f_{\text{high}}) >$ [5]

Crucial Assertion Validations

1. The Invariance of the Fine Structure Constant: The fine structure constant, represented as $\alpha \approx 1/137$, is a dimensionless scalar that inherently dictates the strength of all electromagnetic interactions. The Test_FineStructureConstant_Invariance suite guarantees that as the speed of light decays continuously over 15 billion simulated years, the ConstantsManager simultaneously covaries the elementary charge (e) or the reduced Planck constant ($\hbar$).[5] The assertion requires that the ratio $\alpha = \frac{e^2}{\hbar c}$ remains perfectly balanced to infinity.[5] If α drifts even by a fractional floating-point margin, fundamental atomic bonds in the simulated universe would mathematically dissolve, rendering the universe uninhabitable.

2. Kinematic Supernova Time Dilation: A major historical criticism of classical Tired Light theories is their apparent inability to explain the temporal stretching of Type Ia supernova light curves, which ΛCDM attributes to metric expansion.[16] The Test_SNIa_LightCurve_TimeDilation suite asserts that a simulated light pulse emitted with a specific duration (Δt_e) from a high redshift distance (z) is received by the simulation observer geometrically stretched to a new duration: $\Delta t_o = \Delta t_e(1+z)$.[5] The TypeScript engine naturally passes this test via Pipino's VSL kinematic logic.[5] Because the speed of light continuously decreases during transit, the leading edge of a photon packet travels marginally faster than its trailing edge, which is emitted fractions of a second later when c is infinitesimally slower. This minute velocity

differential geometrically stretches the wave packet over cosmic distances, confirming the observation without metric expansion.[5]

3. Proca Mass Vacuum Dispersion: The Test_PhotonMass_ProcaDispersion suite formally validates the massive vector field updates in the MetricTensorSolver. The test fires two PhotonEntity instances—one with an exceptionally high frequency, and one with a significantly lower frequency—simultaneously across the static spatial grid. The absolute assertion requires that the high-frequency photon arrives at the destination array index definitively before the low-frequency photon. This confirms the engine is successfully applying the Proca mass-induced drag equations, validating the core mechanic required for the dark matter substrate freeze-out.[5]

Architectural Synthesis

The comprehensive design and implementation of a custom TypeScript physics engine dedicated to speculative cosmology and advanced theoretical propulsion dynamics requires a methodical deconstruction of nearly all assumptions held by classical software physics. By replacing semi-implicit Euler integration with highly adaptive RK4 solvers, swapping absolute spatial arrays for the relational mechanics of Assis and Weber gravitation, and abandoning rigid gauge invariance for the frequency-dependent drag of Proca massive electrodynamics, the resulting software architecture serves as a perfectly calibrated virtual laboratory.
It proves capable of not only visually simulating the unifying CCC+TL cosmological model but accurately tracking the complex thermodynamic freeze-out of exhausted photons into sub-luminal macroscopic BECs. It seamlessly renders the highly localized quantum optics required to scoop, compress, and freeze this dark matter substrate using two-color stationary light pulses and macroscopic Electromagnetically Induced Transparency fields. Governed continuously by a rigorous, structurally embedded TDD matrix ensuring absolute thermodynamic and dimensionless stability across billion-year epochs, this architectural blueprint brilliantly bridges the gap between frontier theoretical physics and high-performance software engineering, demonstrating conclusively that complex scalar fields, varying natural constants, and frictionless non-Newtonian propulsion dynamics can be seamlessly resolved within a deterministic digital environment.

Works cited

1. physics-engine · GitHub Topics, accessed May 28, 2026, https://github.com/topics/physics-engine?l=typescript
2. Server-Side Matter.js with Socket.io, P5, and Typescript | by Dilum Bandara | Medium, accessed May 28, 2026, https://dilumbandara.medium.com/server-side-matter-js-with-socket-io-p5-and-typescript-bb55219ad754
3. Physics Engine 13x Faster than Matter.js : r/typescript - Reddit, accessed May 28, 2026, https://www.reddit.com/r/typescript/comments/14ogysu/physics_engine_13x_fas

ter_than_matterjs/
4. Sopiro/Physics: 2D Physics engine written in Typescript - GitHub, accessed May 28, 2026, https://github.com/Sopiro/Physics
5. Designing Tired Light Simulation Software.md
6. The evolution of various energy densities in the CCC+TL model plotted... - ResearchGate, accessed May 28, 2026, https://www.researchgate.net/figure/The-evolution-of-various-energy-densities-in-the-CCC-TL-model-plotted-against-the_fig1_400549105
7. Build a simple 2D physics engine for JavaScript games - IBM Developer, accessed May 28, 2026, https://developer.ibm.com/tutorials/wa-build2dphysicsengine/
8. Collision Detection: Custom or Physics Library? : r/gameenginedevs - Reddit, accessed May 28, 2026, https://www.reddit.com/r/gameenginedevs/comments/1lw6vvg/collision_detection_custom_or_physics_library/
9. Dev Log #1 — Custom Engine : Writing my collision system | by Erikkubiak | Medium, accessed May 28, 2026, https://medium.com/@erikkubiak/dev-log-1-custom-engine-writing-my-collision-system-2a97856f9a93
10. I am making my own 2D game engine in TypeScript but I always get stuck with collisions, accessed May 28, 2026, https://www.reddit.com/r/gameenginedevs/comments/12jti4t/i_am_making_my_own_2d_game_engine_in_typescript/
11. Variable Speed of Light with Time and General Relativity - SCIRP, accessed May 28, 2026, https://www.scirp.org/journal/paperinformation?paperid=108887
12. Standard Cosmological Model vs. Time Variable Light Speed Model - ResearchGate, accessed May 28, 2026, https://www.researchgate.net/publication/400023091_Standard_Cosmological_Model_vs_Time_Variable_Light_Speed_Model
13. (PDF) Variable Speed of Light with Time and General Relativity - ResearchGate, accessed May 28, 2026, https://www.researchgate.net/publication/351243234_Variable_Speed_of_Light_with_Time_and_General_Relativity
14. International Journal of Theoretical & Computational Physics - Tired Light : How it is The Missing Stone in the Infinite Universe with no Beginning - Uniscience Publishers, accessed May 28, 2026, https://unisciencepub.com/wp-content/uploads/2025/01/Tired-Light-How-it-is-The-Missing-Stone-in-the-Infinite-Universe-with-no-Beginning.pdf
15. Curvature pressure in a cosmology with a tired-light redshift - arXiv, accessed May 28, 2026, https://arxiv.org/pdf/astro-ph/9904131
16. Dark Energy Survey Supernova Program: slow supernovae show cosmological time dilation out to z ~ 1. | Monthly Notices of the Royal Astronomical Society | Oxford Academic, accessed May 28, 2026, https://academic.oup.com/mnras/article/533/3/3365/7738388
17. (PDF) Testing CCC+TL Cosmology with Observed Baryon Acoustic Oscillation

Features, accessed May 28, 2026, https://www.researchgate.net/publication/379007561_Testing_CCCTL_Cosmology_with_Observed_Baryon_Acoustic_Oscillation_Features
18. Erratum: “Testing CCC+TL Cosmology with Observed Baryon Acoustic Oscillation Features” (2024, ApJ, 964 , 55) - ResearchGate, accessed May 28, 2026, https://www.researchgate.net/publication/391624654_Erratum_Testing_CCCTL_Cosmology_with_Observed_Baryon_Acoustic_Oscillation_Features_2024_ApJ_964_55
19. Laboratory Medicine, accessed May 28, 2026, https://perpus-utama.poltekkes-malang.ac.id/assets/file/jurnal/03_Vol_55,_Issue_3,_May_2024.pdf
20. A new equation may explain the Universe without dark matter | ScienceDaily, accessed May 28, 2026, https://www.sciencedaily.com/releases/2025/11/251106003906.htm
21. Proca Metamaterials, Massive Electromagnetism, and Nonlocality - TechRxiv, accessed May 28, 2026, https://www.techrxiv.org/doi/pdf/10.36227/techrxiv.13567529
22. The mass of the photon, accessed May 28, 2026, http://home.ustc.edu.cn/~gengb/190927/10.1088.0034-4885.68.1.R02.pdf
23. Proca Electrodynamics Approach to The Massive Photon - ResearchGate, accessed May 28, 2026, https://www.researchgate.net/publication/381028782_Proca_Electrodynamics_Approach_to_The_Massive_Photon
24. Gravitational induction with Weber's force - Canadian Science Publishing, accessed May 28, 2026, https://cdnsciencepub.com/doi/10.1139/cjp-2015-0285
25. Relational Mechanics - Unicamp, accessed May 28, 2026, https://www.ifi.unicamp.br/~assis/Relational-Mechanics-Mach-Weber.pdf
26. Relational Mechanics - isidore.co, accessed May 28, 2026, https://isidore.co/misc/Physics%20papers%20and%20books/Zotero/storage/8B6LWQHZ/Assis%20-%202014%20-%20Relational%20Mechanics%20and%20Implementation%20of%20Mach's%20.pdf
27. (PDF) Relational Mechanics - ResearchGate, accessed May 28, 2026, https://www.researchgate.net/publication/314510532_Relational_Mechanics
28. 1 Quantum manipulation of two-color stationary light: Quantum wavelength conversion - arXiv, accessed May 28, 2026, https://arxiv.org/pdf/quant-ph/0512165
29. 1 Manipulation of Two-Color Stationary Light Using Coherence Moving Gratings - arXiv, accessed May 28, 2026, https://arxiv.org/pdf/quant-ph/0512042
30. Enhancement of optical nonlinearities with stationary light - Niels Bohr Institutet, accessed May 28, 2026, https://nbi.ku.dk/english/theses/phd-theses/ivan-iakoupov/PhD_Thesis_Ivan_Iakopov.pdf
31. Vibration induced transparency: Simulating an optomechanical system via the

cavity QED setup with a movable atom - PMC, accessed May 28, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC11197654/
32. Slow supernovae show cosmological time dilation out to z∼1. - arXiv, accessed May 28, 2026, https://arxiv.org/html/2406.05050v2