Relational Cosmology and the ArcSecs Physics Engine: A Deterministic Enterprise Architecture for Multi-Messenger Astrophysics

The foundational architecture of modern astrophysics is overwhelmingly dictated by the geometric tenets of General Relativity and the ΛCDM standard cosmological model. In this traditional framework, gravity is not interpreted as a conventional kinetic force but rather as the physical manifestation of curvature within a four-dimensional spacetime manifold.[1]

Consequently, this paradigm enforces the speed of light (c) as an invariant, universal speed limit, which mathematically necessitates the invocation of time dilation and length contraction to preserve causality across varied frames of reference.[1] While the equations formulated by Albert Einstein in the early 20th century provided extraordinary mathematical shortcuts for modeling macroscopic astrophysical geometry in an era prior to high-performance computing, the imposition of a physical spacetime fabric introduces profound epistemological and computational crises.[1] These crises include the formulation of infinite singularities at black hole event horizons, an inherent incompatibility with the probabilistic, discrete nature of quantum mechanics, and a reliance on undetected dark matter and dark energy constructs to account for the vast majority of the universe's mass-energy density.[1]

The ArcSecs Cosmological Physics Engine, and its integrated Dark Matter Drive Simulator, are constructed to transcend these limitations by establishing a framework of the universe that explicitly functions without the constructs of physical spacetime and relativistic paradoxes such as time dilation.[1] The core philosophy of this simulation architecture operates on the principle that the immense computational power available today allows for exact, relational kinetic calculations of billions of discrete particles, rendering macroscopic geometric shortcuts obsolete.[1] Rather than modeling a universe where a geometric fabric dictates the behavior of matter, the engine maps physical reality strictly through the scalar distances between discrete masses and the temporal derivatives of those distances.[1]

The visual and mathematical goal of the ArcSecs architecture is twofold: to deliver a level of exactness, deterministic mathematical stability, and data-rich visualization rigorous enough for a professional physicist, while simultaneously maintaining an interactive, accessible, and visually engaging simulation environment that is fun and comprehensible for high school students.[1] Achieving this requires an enterprise-level software architecture built entirely in native TypeScript, leveraging Data-Oriented Design (DOD), Entity-Component-System (ECS) pipelines, automated unit testing, and highly optimized WebGL2 rendering to simulate light traveling at all velocities from all directions.[1] This report provides an exhaustive analysis of the theoretical mechanics, cosmological resolutions, and software engineering protocols required to

construct this relational cosmology.

The Metaphysical Departure: Eliminating Spacetime and Time Dilation

To engineer a simulation environment devoid of spacetime, the substantivalist perspective—which views space as a physical container capable of stretching, rippling, and expanding—must be systematically deconstructed.[1] The ArcSecs framework replaces the pseudo-Riemannian continuous manifold with a perfectly static, continuous Euclidean void.[1] This void possesses no physical substance, topological geometry, or material properties, meaning the space itself cannot metric-expand.[1]

Because the void is not a physical substance, absolute coordinate tracking within it is computationally meaningless. Instead, the universe is represented mathematically as a massive relational graph where entities function as material nodes and physical distances serve as the relational edges connecting them.[1] Within the engine's internal solver, Cartesian spatial coordinates are treated merely as cosmetic rendering labels projected onto the screen to aid the user's visual comprehension; they possess no mathematical weight in calculating interactions.[1] This aligns with the principles of diffeomorphism invariance and the Hole Argument, which mathematically dictate that coordinate systems lack intrinsic physical reality, demonstrating that gravitational phenomena can be accurately predicted without relying on a physical spacetime.[1]

Within an empty Euclidean void where absolute motion is a mathematical non-entity, only relative distance, velocity, and acceleration exist.[1] Without an absolute background geometric grid establishing a universal speed limit, the separation velocities between distant galaxies can easily exceed the speed of light without violating any physical laws.[1] The ArcSecs engine visually demonstrates this non-paradoxical superluminal expansion through its Big Bang Velocity Paradox Scenario. In this simulation mode, a central origin point spawns two entities that travel in strictly opposite directions, each at a relative speed of $0.9c$. Utilizing purely Galilean relative velocity calculations ($v_{rel} = v_a - v_b$), the user interface displays a true relative separation velocity of $1.8c$, successfully proving the kinetic viability of a universe unrestricted by invariant light-speed bounds.[1]

Reassessing Absolute Time and Quantum Gravitational Damping

If space and time are not merged into a unified four-dimensional block, time must be reasserted as an absolute, universal constant.[1] Relativistic models assert that time itself physically slows down relative to an observer's velocity or gravitational environment, pointing to the desynchronization of highly precise cesium atomic clocks (such as the 38 microseconds per day gained by GPS satellites in orbit) as empirical proof.[1]

However, the ArcSecs framework rejects time dilation as a geometric illusion, asserting instead that particles and physical mechanisms are tangibly affected by gravity.[1] The simulation models the Euclidean void as being permeated by an ambient electromagnetic substrate—a massive

vector field governed by modified Proca equations.[1] Deep within a planetary gravity well, this ambient substrate is highly concentrated. When a cesium atomic clock operates on the surface of the Earth, this dense substrate exerts microscopic kinetic drag, termed "quantum gravitational damping," on the internal electron transitions of the cesium atoms.[1] This physical friction mechanically retards the ticking rate of the clock. In higher orbits, the substrate diffuses, causing less physical drag and allowing the atomic transitions to occur faster.[1] Therefore, the clocks do not measure a warping of time itself; they merely reflect mechanically degraded operations caused by gravitational influence on subatomic particles.[1]

This mechanism is meticulously modeled within the TypeScript architecture via the AtomicOscillatorSystem.[1] Rather than applying Lorentz transformations to an entity's local temporal state, the system queries the local gravitational force and substrate density from a 3D Barnes-Hut Octree and applies a direct mechanical damping scalar to the oscillation frequency of atomic components.[1] Thus, the absolute flow of time within the main TypeScript loop remains invariant, tracked by a globally consistent deltaTime parameter that is entirely decoupled from any entity's spatial coordinates or velocity.[1]

Kinematic Stabilization and the Rejection of Relativistic Mass

The survival of high-velocity atmospheric muons (which travel at approximately $0.999c$ and reach the Earth's surface before decaying) is historically cited as definitive proof of time dilation stretching their $2.2\mu s$ mean lifespan. The relational physics engine explains this phenomenon mechanically rather than temporally.[1] As a muon traverses the dense ambient massive substrate at extreme velocities, it generates a highly localized electromagnetic "bow shock".[1] This extreme physical interaction physically binds and stabilizes the internal structural coherence of the particle, mechanically delaying its decay into electrons and neutrinos.[1] The principle is analogous to classical mechanics, wherein a spinning top gains gyroscopic stability and resists falling when operating at extreme rotational velocities.

Furthermore, the concept of relativistic mass—wherein an object's mass theoretically inflates toward infinity as it approaches the speed of light—is explicitly abolished in the ArcSecs engine.[1] Mass is treated strictly as a Lorentz scalar (m_0) that does not mathematically inflate.[1] Consequently, the kinetic energy required to accelerate objects to extreme velocities remains finite, establishing a mathematically stable environment for simulating the Dark Matter Drive's speculative inertial decoupling behavior.[1]

Measuring the Universe: Rejecting Lightyears and Cesium Clocks

Because standard cosmological models rely entirely on the invariant speed of light to measure vast distances (lightyears) and utilize clocks affected by gravitational drag to measure time, their foundational metrics are inherently flawed within a relational framework.[1] If the speed of light is not a consistent universal constant, then the lightyear ceases to be a valid or reliable unit of

spatial measurement. Furthermore, if atomic clocks are physically degraded by ambient gravitational fields, they cannot be trusted to establish a consistent timeline across disparate cosmological environments.[1]

Therefore, the ArcSecs framework must establish entirely new methodologies for calculating distance and time without relying on light or standard clocks. The engine adopts the parsec (and its subdivisions into arcseconds) as the absolute geometric standard for measuring spatial separation between nodes in the relational graph.[1] The mathematical law governing the simulation's distance matrix requires seamless scaling from local arcseconds to cosmological parsecs using the rigid baseline where distance (d) equals the inverse of the parallax angle ($d = 1/p$).[1]

To calculate a consistent flow of time throughout the universe, the engine looks to interactions that are strictly indifferent to the Proca substrate drag that slows electromagnetic waves and atomic particles. Gravitational waves, representing the direct relational propagation of kinetic shifts between massive bodies, are modeled as traveling at an invariant, absolute velocity ($v_{GW} = c_0 = 299,792,458 \text{ m/s}$), entirely unaffected by the electromagnetic friction that slows both physical clocks and optical light.[1] Thus, gravitational wave propagation serves as the ultimate, uncorrupted metric for universal time interval calibration.[1]

The Unified Theory of VSL and Energy Degeneration

To map the universe using parsecs and gravitational wave timing, the engine models electromagnetic radiation under a framework combining Variable Speed of Light (VSL) dynamics with a modernized Tired Light mechanism.[1]

While gravitational waves travel at a constant velocity, electromagnetic waves (light) experience a continuous, secular deceleration due to fluid-dynamic drag against the finely distributed intergalactic medium or quantum vacuum.[1] The instantaneous velocity of electromagnetic radiation, $v_{EM}(d)$, decays exponentially as a function of the absolute physical distance traversed, d, governed by a universal deceleration constant, α[1]:

$$v_{EM}(d) = c_0 e^{-\alpha d}$$

Simultaneously, the engine utilizes energy degeneration to account for cosmological redshift (z) without invoking the metric expansion of space. As photons traverse the vacuum, they continuously lose intrinsic energy, $E(d)$, over distance governed by a spatial attenuation coefficient, μ[1]:

$$E(d) = E_0 e^{-\mu d}$$

This unified framework forms the core mathematical logic allowing the ArcSecs engine to computationally dissect multi-messenger astronomical events, separating the invariant timing of gravitational waves from the degraded, decelerated timing of their optical counterparts.[1]

Multi-Messenger Astrophysics as Calibration Anchors

The ArcSecs engine utilizes highly detailed data from multi-messenger astronomical events to calibrate its internal variables, mapping realistic cosmological scenarios that are visually appealing and mathematically robust enough for a professional physicist.[1] The mathematical pipeline disentangles the kinematic delays caused by the vacuum from the intrinsic delays generated at the astrophysical source.

The total observed arrival time delay between a gravitational wave and its electromagnetic counterpart (Δt_{obs}) is defined as:

$$\Delta t_{obs} - \Delta t_{src} + \Delta t_{vac}$$

Where Δt_{src} is the physical delay at the source (such as the time required for a relativistic jet to break out of stellar ejecta) and Δt_{vac} is the propagation delay caused by the slowing of light over a physical distance d .[1]

GW170817: The Primary Calibration Anchor

On August 17, 2017, the Advanced LIGO and Virgo gravitational-wave detectors observed GW170817, the inspiral and coalescence of a binary neutron star system located in the host galaxy NGC 4993.[6] The event was detected with a combined signal-to-noise ratio of 32.4 and generated a gravitational wave signal that extended down to 23 Hz, lasting until the merger.[7] Exactly 1.7 seconds after the coalescence was recorded by LIGO, the Fermi Gamma-ray Burst Monitor detected the electromagnetic counterpart, GRB 170817A.[8]

Parameter	Legacy Metric	ArcSecs Relational Metric
Component Masses	1.6 and 1.1 $M_{\odot}$ [9]	1.6 and 1.1 $M_{\odot}$ (Scalar Rest Mass)
Total Mass	$\approx$ [7]	$\approx$
Host Galaxy	NGC 4993 [6]	NGC 4993
Localization	16 to 28 square degrees [7]	Exact Graph Node

		Coordinate
Observed Redshift (z)	0.0099 [6]	0.0099 (Energy Degeneration)
Calculated Distance	144 million light-years / 40 Mpc [6]	40 Megaparsecs ($d = \frac{\ln(1+z)}{\mu}$)

Because GW170817 is a close, well-constrained event with a highly accurate distance measurement of 40 Mpc, the engine uses it as the foundational anchor to calculate the absolute attenuation coefficient (μ) for the simulated universe.[1] By applying the redshift ($z = 0.0099$) to the equation $\mu = \frac{\ln(1+z)}{d}$, the engine calibrates $\mu \approx 8.0 \times 10^{-27}\ \mathrm{m}^{-1}$.[1]

By equating the velocity deceleration constant with the energy decay constant ($\alpha = \mu$), the engine computes the expected vacuum propagation delay (Δt_{vac}). Over a span of 40 Mpc, the deceleration of the electromagnetic wave yields a vacuum latency of roughly 0.020 seconds relative to the invariant gravitational wave.[1] Subtracting this from the total observed delay of 1.7 seconds isolates a true source delay (Δt_{src}) of approximately 1.68 seconds, accurately representing the precise physical duration required for the gamma-ray jet to puncture the neutron star merger's dense ejecta shell before entering the vacuum.[1]

GW150914: Strain Frequency and Inertial Dynamics

The first direct observation of gravitational waves, GW150914, involved the cataclysmic merger of two massive black holes.[10] The physics engine utilizes the highly precise strain frequency footprint of this event to simulate relational inertial dynamics and Teleparallel gravitational attraction accurately.[1]

Parameter	Legacy Metric	ArcSecs Relational Metric
Primary Mass	36 $M_\odot$ [11]	36 $M_\odot$
Secondary Mass	29 $M_\odot$ [11]	29 $M_\odot$
Final Remnant Mass	62 $M_\odot$ [11]	62 $M_\odot$

Radiated Energy	3.0 $M_\odot$ [11]	3.0 $M_\odot$ (Transferred to Graph Edges)
Peak Power Output	$3.6 \times$ watts [12]	$3.6 \times$ watts
Strain Frequency Sweep	35 Hz to 250 Hz over 0.2s [10]	35 Hz to 250 Hz (Relational Velocity)
Orbital Separation	350 km at merger [12]	350 km (Graph Node Distance)
Redshift (z)	0.093 [12]	0.093
Calculated Distance	1.3 - 1.4 billion light-years / 410-440 Mpc [11]	410-440 Megaparsecs

During the final 20 milliseconds of the merger, the relative tangential velocity of the black holes increased from 30% to 60% of the speed of light, producing a strain amplitude that peaked at 10^{-21}.[11] In the ArcSecs environment, these extreme velocities are calculated without introducing geometric spacetime drag. Instead, the WeberTether components connecting the binary nodes evaluate the relative radial velocity and acceleration, applying purely relational forces to dictate the inspiral.[1] The 3.0 solar masses radiated away as energy are tracked meticulously by the internal ledger and dispersed as kinetic shifts across the relational edges of the entire simulated universe.[1]

GW190521: Deep-Space Optical Diffusion

GW190521 represents the most extreme multi-messenger scenario managed by the engine. Detected by LIGO and Virgo with a three-detector network signal-to-noise ratio of 14.7, this event was generated by the inspiral of two black holes (85 and 66 $M_\odot$), yielding a 142 $M_\odot$ intermediate-mass black hole (IMBH).[13]

Parameter	Legacy Metric	ArcSecs Relational Metric
Component Masses	85 and 66 $M_\odot$ [13]	85 and 66 $M_\odot$

Remnant Mass	142 $M_\odot$ (IMBH) [13]	142 $M_\odot$
Radiated Energy	8.0 $M_\odot$ [13]	8.0 $M_\odot$
Duration	< 0.1 seconds [15]	< 0.1 seconds
Observed Redshift (z)	0.82 [14]	0.82
Distance Measurement	17,000 million light-years / 5.3 Gpc [13]	5.3 Gigaparsecs (Static Mapping)

Because this event occurred at a high cosmological redshift ($z = 0.82$), legacy models rely on the metric expansion of space to place it at a luminosity distance of 5.3 Gigaparsecs.[13] The ArcSecs multi-messenger algorithm outright rejects this expansion-derived mapping. Instead, it calculates the true, static physical distance strictly from the redshift using the μ attenuation coefficient derived from GW170817: $d = \frac{\ln(1+z)}{\mu}$.[1]

GW190521 is highly notable for a debated optical flare that occurred in the Active Galactic Nucleus (AGN) environment surrounding the merger, observed approximately 34 days post-coalescence.[1] Using the VSL pipeline, the engine calculates the vacuum propagation delay (Δt_{vac}) for light traveling 5.3 Gpc to be roughly 27.6 seconds.[1] Because the expected vacuum delay is astronomically smaller than the observed 34-day delay, the engine correctly categorizes the massive discrepancy as localized environmental diffusion lag (Δt_{src}).[1] This successfully quantifies the physical time required for the electromagnetic flare to diffuse and escape the dense, optically thick accretion disk of the AGN before entering the vacuum.[1]

Resolving Cosmological Anomalies Through Relational Physics

For a relational framework to supplant standard cosmology, the physics engine must computationally resolve observational anomalies that typically require the assumption of metric expansion or geometric curvature.[1]

Supernova Time Dilation: Rectifying the Weakness of Tired Light

The original Tired Light model proposed by Fritz Zwicky in 1929 elegantly accounted for the redshift-distance relationship without requiring an expanding universe.[5] However, the model was overwhelmingly dismissed by mainstream astrophysics because it theoretically failed to account

for cosmological time dilation.[5] In standard expanding universe models, the light curves of high-redshift Type Ia supernovae are temporally stretched; a supernova that requires 20 days to decay locally will appear to take 40 days to decay when observed at a redshift of $z = 1$.[17] Extensive studies, such as Goldhaber's 2001 analysis and Blondin's 2008 spectral aging analysis, confirmed that the width factor (w) of supernova light curves scales exactly as $w = (1 + z)$.[17] A pure energy-loss Tired Light model predicts that the duration of a supernova would remain constant regardless of distance, contradicting empirical data at a 10-sigma rejection level.[17]

The ArcSecs unified VSL pipeline completely resolves this historical weakness.[1] By coupling energy degeneration with electromagnetic velocity attenuation (light slowing), the time dilation is reinterpreted kinetically. If two sequential photons are emitted from a supernova with a brief interval between them (Δt_{emit}), the arrival interval at the observer (Δt_{obs}) is physically stretched because the velocity of the arriving light has decayed drastically over the vast parsec distance.[1] Because the engine unifies the deceleration constant and the energy decay coefficient ($\alpha = \mu$), the arrival interval yields exactly $\Delta t_{obs} = \Delta t_{emit}(1 + z)$.[1] This mathematically guarantees that the observed time dilation of high-redshift supernovae is perfectly replicated in the simulator's static universe, eliminating the necessity for space to expand.[1]

The Hubble Tension and Opaque Regions

The "Hubble Tension" represents a critical failure in the ΛCDM model, driven by severe discrepancies in measuring the Hubble constant (H_0). Measurements derived from local "standard candles" (like Type Ia supernovae via the SH0ES project) conflict sharply with measurements inferred from the sound horizon in the Cosmic Microwave Background (CMB).[20] The ArcSecs engine reframes the Hubble tension not as a paradox of metric expansion, but purely as a question of optical propagation history.[1] Recent high-redshift galaxy distributions observed by the JWST stand in direct tension with standard cosmological timelines, breathing new mathematical life into the Tired Light framework.[23] The engine processes the CMB not as the relic radiation of a primordial Big Bang plasma, but as the thermalized superposition of microwave radiation from countless galaxies subjected to extreme energy degeneration.[23] The energy degeneration model predicts a finite optical path per Hubble radius, establishing a volumetric "opaque region" mathematically defined as a sphere with a radius of $R_{max} = 240\ \mathrm{Mpc}$ ($z \approx 0.051$) centered on the observer.[23] Within this simulated region, the superposition of severely scattered, degenerate light mimics a perfect black body, accurately generating the CMB power spectrum and resolving the tension without invoking an expanding universe.[23]

Flat-Space Gravitational Lensing and the Soldner Discrepancy

To visually and mathematically render the deflection of light around massive bodies without resorting to curved spacetime, the engine utilizes Machian and Weber proxy mechanics within a Teleparallel Gravity framework. This architecture computes gravity strictly as flat-coordinate force and torsion-style vectors.[1]

However, simulating gravitational lensing in a strictly flat-space environment exposes a severe mathematical vulnerability known as the Soldner corpuscular discrepancy.[1] Treating photons purely as classical massive particles, the Newtonian deflection angle calculated for light grazing the Sun is approximately 0.83 arcseconds.[1] This drastically underpredicts observational reality, where the relativistic baseline stands at 1.75 arcseconds (General Relativity dictates that spatial curvature effectively doubles the Newtonian deflection).[1] To ensure the visuals meet the exactness required by professional physicists, the engine's internal Relational Architecture Console introduces theoretical, synthetic "refractive corrections".[1] These algorithms execute concurrently with the corpuscular deflection proxy, allowing the massive photons to bend correctly around massive nodes without invoking a metric tensor.[1]

The Forward-Scattering Problem and Vacuum Latency

Another classical objection to Tired Light hypotheses is the forward-scattering problem.[5] If photons lose energy via physical collisions with the intergalactic medium or the Proca substrate, the conservation of momentum requires that the photon's trajectory must be altered. Consequently, a massive number of collisions over parsec distances would hopelessly blur and smear distant galaxies across the sky, destroying crisp optical resolution.[5]

The ArcSecs engine directly tackles this limitation. When users manipulate the tired-light decay parameter in the UI, the simulation computes the trajectory randomization in real-time, displaying "forward-scattering cautions" to demonstrate the exact optical degradation.[1] The engine mitigates severe theoretical blurring by applying the principles of massive photon dispersion governed by the Proca equations.[1] Replacing standard Maxwellian vacuum electrodynamics with a massive spin-1 particle model irrevocably alters classical symmetry.[1] In this speculative vacuum, light experiences extreme frequency dispersion; lower-frequency photon packets lag behind high-frequency gamma rays.[1] This vacuum latency creates a unique temporal smear rather than a purely spatial blur, allowing the engine to mathematically justify relatively coherent optical imaging over vast distances while still satisfying the energy degeneration required for redshift.[1]

Enterprise Architecture: TypeScript and ECS Implementation

Translating these complex relational mathematics into an interactive demo capable of running natively in a web browser requires an uncompromising approach to software architecture. The ArcSecs engine is engineered entirely in vanilla TypeScript, deliberately eschewing heavy,

high-level wrapper libraries (such as Babylon.js or Three.js) to maintain absolute control over the memory heap and execution pipeline.[1]

Data-Oriented Design and Contiguous Memory

Traditional Object-Oriented Programming (OOP) physics models instantiate entities as discrete objects containing localized methods and properties (e.g., class Photon { position: Vector3, update() }). In dense N-body simulations managing tens of thousands of particles, this OOP approach scatters data across the memory heap.[1] Iterating over these fragmented objects results in massive CPU cache misses and triggers devastating garbage collection pauses within the V8 JavaScript engine, destroying the steady 60 FPS necessary for accurate physics rendering.[1]

The ArcSecs architecture strictly adheres to Data-Oriented Design (DOD) via an Entity-Component-System (ECS) framework:

- **Abstract Entities:** Entities possess no inherent object data; they are strictly tracked as integer indices ranging from 0 to 5,000,000.[1]
- **Contiguous Components:** Data is completely decoupled from logic and stored in pre-allocated, flat typed-arrays using a Structure of Arrays (SoA) layout.[1]
- **Stateless Systems:** The logic operates as pure, side-effect-free functions that iterate sequentially across the memory buffers, maximizing L1 and L2 cache utilization.[1]

The physical coordinates are housed in the KinematicMemoryBuffer, utilizing 64-bit precision Float64Array structures.[1] This double-precision formatting is absolutely critical to prevent spatial jitter, as the engine must simultaneously resolve astronomical distances measured in Gigaparsecs and quantum-scale interactions within the atomic oscillator subsystems.[1] The PhysicalPropertyBuffer maintains discrete, contiguous arrays for mass, radius, and the critical internalEnergy parameter, which tracks the exact remaining thermodynamic energy in tired light photons.[1] A Uint8Array bitmask dictates the entity type, enabling the systems to rapidly filter Massive Bodies, Photons, Spacecraft, and Atomic Oscillators using bitwise operations.[1]

To entirely eradicate garbage collection overhead during the dense rendering of photon emissions and energy harvesting, the engine implements a Ring Buffer Object Pool. When a photon entity is deactivated or harvested by a spacecraft's warp bubble, its integer ID is instantly pushed to the ring buffer for immediate recycling.[1]

Relational Execution Pipeline and Barnes-Hut Octrees

The core execution loop of the ECS processes the relational graph in a strict sequential order:

1. **Substrate Field System:** Evaluates the density and flux of the Proca vector potential field across the simulation space.[1]
2. **Weber Force System:** Computes the pairwise interaction forces across all WeberTether components.[1] If a spacecraft engages its Dark Matter Drive, this system applies a dynamic suppression multiplier, effectively decoupling the vessel from the gravitational drag of the distant cosmic shell and lowering its effective inertia.[1]
3. **Relational Inertia System:** Dynamically updates the MachianMass of all entities based

on the summation of active tether forces.[1]

4. **Vector Potential Thrust System:** Reads the ProcaCoupler data to generate electromotive physical thrust by pushing the vessel against the ambient Proca substrate.[1]
5. **Relational Integration System:** Calculates movement utilizing a symplectic Velocity Verlet integrator. Because the Velocity Verlet method requires only a single force evaluation per time step, it rigorously conserves total kinetic energy over long simulation horizons, preventing the artificial orbital expansion that plagues standard explicit Euler integrators.[1]

Computing gravitational vectors and relational tether forces for N bodies typically results in an $O(N^2)$ computational explosion.[1] To bypass this limitation, the engine integrates a 3D Barnes-Hut Octree algorithm. The simulation volume is recursively subdivided into eight cubic regions. For entities separated by massive distances, the engine aggregates the properties of entire nodes and treats them as singular centers of mass, dramatically reducing the complexity to $O(N \log N)$.[1] Furthermore, to prevent the traditional inverse-square law ($F \propto \frac{1}{r^2}$) from returning infinite forces or division-by-zero errors when particles collide, a configurable "Softening Radius" (ϵ) is applied to the denominator ($F \propto \frac{1}{r^2+\epsilon^2}$), guaranteeing numerical stability at subatomic proximities.[1]

Deterministic Mathematical Stability and Unit Testing

To serve as an enterprise-level testbed for alternative physics theories, the engine cannot rely on external astronomical validation to prove its logic; its highest mandate is absolute internal mathematical consistency.[1] To achieve this, the TypeScript/WASM architecture must be entirely cross-platform deterministic.[24] Running the same simulation with identical initial conditions on different operating systems or browser engines must yield the exact same physical state and byte hash.[2]

Mitigating IEEE-754 Floating-Point Imprecision

A significant obstacle in JavaScript-based physics engines is the inherent unreliability of IEEE-754 double-precision floating-point numbers.[26] Because numbers are stored with a finite bit allocation (a 52-bit mantissa), irrational fractions and extensive decimals cannot be perfectly represented; computationally, $0.1 + 0.2$ equates to 0.30000000000000004 rather than exactly 0.3.[26]

When developing automated unit tests using the Jest testing framework, attempting to evaluate physical coordinates using strict equality (===) routinely results in test failure.[27] A common, yet flawed, mitigation strategy is to use the native Number.EPSILON constant (the difference between 1 and the next representable value) as a tolerance threshold, formatted as Math.abs(f1 - f2) < Number.EPSILON.[26] However, the ArcSecs engine calculates distances

spanning vast Megaparsecs; as the exponent of a floating-point number grows, its absolute accuracy diminishes, causing Number.EPSILON evaluations to fail catastrophically for massive inputs.[26]

To ensure stability across the unit tests, the engine's Jest suites enforce dynamic, scaled error tolerances. The testing architecture utilizes custom comparative algorithms, such as epsilon = Math.max(Math.abs(f1), Math.abs(f2)) * 1e-4, which scales the acceptable accuracy margin appropriately based on the magnitude of the cosmological bodies being tested.[28] Alternatively, specific mathematical subsystems leverage Jest's native expect.toBeCloseTo() parameter with explicitly defined sub-property unpackers to guarantee precise evaluation limits.[28]

Fixed-Point Trigonometry and Immutable Ticks

Native JavaScript transcendental functions, such as Math.sin and Math.cos, are fundamentally non-deterministic across platforms.[24] The engine sidesteps this fatal flaw by completely replacing native functions with custom fixed-point trigonometric Look-Up Tables (LUTs). While fixed-point addition operations carry a minor performance cost compared to native numbers, LUT-based trigonometric calculations significantly outperform native calls (e.g., executing in 62.75 ms versus 190.57 ms over 5 million iterations) while guaranteeing absolute cross-platform determinism.[31] Statistical mechanisms within the engine, such as the Monte Carlo distributions used to model random photon absorption within the CMB opaque region, are strictly driven by deterministic seeded pseudorandom number generators (PRNGs), ensuring that scattering trajectories are perfectly repeatable across all validation benchmarks.[2]

Furthermore, the simulation mathematics are entirely decoupled from the visual rendering frame rate.[1] The core execution loop advances at a strict, immutable fixed step (exactly 1/60th of a second per tick). The engine bans the use of adaptive step-sizing integration methods (such as RKF45), as they introduce timing micro-variations that shatter deterministic consistency.[1] A manual "Step" command is integrated into the testing interface, allowing physicists to pause continuous runtime and advance the simulation by exactly one deterministic tick to track precise kinetic energy propagation across the relational edges.[1]

Internal Calibration Ledgers

The automated testing pipeline validates the engine's physics through two primary diagnostic ledgers:

1. **Parallax Arcsecond Calibration:** Because the spatial system relies purely on Euclidean formulations, the internal math must scale immaculately from arcseconds to parsecs using the standard baseline. If the distance matrix fails to align perfectly with this geometric conversion during multi-messenger simulations, the unit test triggers a critical baseline mismatch failure.[1]
2. **The Photon Energy Ledger:** Serving as the definitive thermodynamic accounting protocol, this ledger tracks the absolute numerical conservation of energy as photons decay via the tired-light attenuation coefficient (μ). It rigorously verifies that all kinetic

energy lost to Proca substrate friction or absorbed by spacecraft warp boundaries is mathematically accounted for, ensuring the first law of thermodynamics is perfectly preserved despite the variable speed of light.[1]

WebGL2 Visual Rendering and Educational Interface

To bridge the gap between the rigorous data requirements of a professional physicist and the engaging, interactive needs of a high school student, the ArcSecs engine couples its robust back-end ECS architecture with a highly optimized, raw WebGL2 and HTML5 Canvas frontend.[1] By avoiding high-level graphics libraries, the engine achieves maximum rendering efficiency natively in the browser.[1]

Ping-Pong Buffers and Transform Feedback

In standard web-based particle systems, the CPU computes the physical behavior of thousands of particles and must upload that data to the GPU for rendering on every single frame. This constant data transfer suffocates the system bus, frequently resulting in massive frame rate drops and stuttering (often plunging to 15 FPS when processing 10,000 particles).[3]
To bypass this bottleneck entirely, the ArcSecs visualizer utilizes WebGL2 Transform Feedback with dual "ping-pong" buffers.[3] In this architecture, all position and velocity data remains strictly GPU-resident. During frame A, buffer 1 supplies the input attributes while the GPU's vertex shader computes the updated kinetics, writing the new physical state directly into buffer 2 without ever returning the data to the CPU.[3] On frame B, the roles of the buffers swap.[3] This pre-allocated, zero-upload pipeline allows the physics engine to comfortably render 30,000 interacting particles at a flawless 120 FPS, complete with temporal decay trails, velocity-based coloring, and bloom post-processing, providing an incredibly smooth and visually striking experience for the end-user.[3]

Rendering Relational Physics Concepts

The visual interface transforms dense mathematical mechanics into intuitive, dynamic graphical representations that effectively communicate the departure from standard relativity:

- **Newtonian Dark Stars:** Rather than rendering abstract geometric spacetime funnels to represent black holes, the engine visualizes them as classical compact stars. By adjusting the photon mass scalar in the UI, users can visually observe discrete light particles decelerating under intense relational gravitational forces, reaching a maximum altitude, and falling back to the surface in parabolic arcs like physical projectiles.[1]
- **Warp Bubble Energy Harvesting:** The Dark Matter Drive Simulator visually renders the energetic boundaries of spacecraft navigating the Euclidean void. As the vessel accelerates, the engine renders incoming "tired" photon particles intersecting the bounding volume. The UI highlights these specific particles changing state, dynamically transferring their residual kinetic energy to the vessel’s propulsion reserves, and generating visible electromotive thrust against the Proca substrate.[1]
- **Macular Degeneration Analogs:** To make the concept of tired light and energy degeneration comprehensible to students, the engine draws an analog to biological

systems. Similar to retinal systems failing to regenerate cone pigments rapidly enough to process flickering light—such as seen in $Irbp^{-/-}$ mouse models undergoing photopic electroretinogram testing—the simulator visibly shifts the color mapping and diminishes the brightness of photon entities as their thermodynamic energy exhausts over parsec distances, providing a tangible analogy for cosmological energy decay.[1]

Through this exhaustive enterprise architecture—combining relational mathematical exactness, deterministic TypeScript pipelines, rigorous Jest automated testing, and zero-latency WebGL2 rendering—the ArcSecs Physics Engine successfully constructs a highly detailed, interactive framework of the universe completely free from the paradoxes of spacetime and time dilation.

Works cited

1. Rethinking Light, Time, and Spacetime.md
2. Apg-Rpr - Rapier.js deterministic physics engine tests - GitHub, accessed May 30, 2026, https://github.com/Pangeli70/apg-rpr
3. GPU-resident WebGL2 particle sim (Transform Feedback) with trails & bloom - GitHub, accessed May 30, 2026, https://github.com/senagulen/gpu-particle-system
4. Parsec confusion - astronomy - Physics Stack Exchange, accessed May 30, 2026, https://physics.stackexchange.com/questions/762253/parsec-confusion
5. Tired light - Wikipedia, accessed May 30, 2026, https://en.wikipedia.org/wiki/Tired_light
6. GW170817 - Wikipedia, accessed May 30, 2026, https://en.wikipedia.org/wiki/GW170817
7. Properties of the Binary Neutron Star Merger GW170817 - ADS, accessed May 30, 2026, https://ui.adsabs.harvard.edu/abs/2019PhRvX...9a1001A/abstract
8. GW170817: Observation of Gravitational Waves from a Binary Neutron Star Inspiral - arXiv, accessed May 30, 2026, https://arxiv.org/abs/1710.05832
9. GW170817: a binary neutron star merger, accessed May 30, 2026, https://www.aei.mpg.de/151005/gw170817-a-binary-neutron-star-merger
10. Observation of Gravitational Waves from a Binary Black Hole Merger - LIGO Caltech, accessed May 30, 2026, https://www.ligo.caltech.edu/system/media_files/binaries/301/original/detection-science-summary.pdf
11. The first binary black-hole merger observed by LIGO, accessed May 30, 2026, https://www.aei.mpg.de/168249/the-first-binary-black-hole-merger-observed-by-ligo
12. GW150914 : first observation of gravitational waves - WinStars, accessed May 30, 2026, https://winstars.net/en/2018/03/01/gw150914-first-observation-of-gravitational-waves/
13. GW190521 - Wikipedia, accessed May 30, 2026, https://en.wikipedia.org/wiki/GW190521
14. GW190521: A Binary Black Hole Merger with a Total Mass of 150 M□ - NASA

Technical Reports Server (NTRS), accessed May 30, 2026, https://ntrs.nasa.gov/citations/20210014680
15. A “bang” in LIGO and Virgo detectors signals most massive gravitational-wave source yet, accessed May 30, 2026, https://www.ligo.caltech.edu/LA/news/ligo20200902
16. GW190521: A Binary Black Hole Merger with a Total Mass of $150 ~ M_{\odot} - arXiv, accessed May 30, 2026, https://arxiv.org/abs/2009.01075
17. Errors in Tired Light Cosmology, accessed May 30, 2026, https://www.astro.ucla.edu/~wright/tiredlit.htm
18. Tired light red shift hypothesis - cosmology - Physics Stack Exchange, accessed May 30, 2026, https://physics.stackexchange.com/questions/156618/tired-light-red-shift-hypothesis
19. Type Ia supernovae provide direct evidence for an expanding universe - Berkeley News, accessed May 30, 2026, https://newsarchive.berkeley.edu/news/media/releases/99legacy/1-7-1999.html
20. Hubble Constant and Tension - NASA Science, accessed May 30, 2026, https://science.nasa.gov/mission/hubble/science/science-behind-the-discoveries/hubble-constant-and-tension/
21. The Hubble tension - CERN Courier, accessed May 30, 2026, https://cerncourier.com/a/the-hubble-tension/
22. The Unsettled Number: Hubble's Tension - MDPI, accessed May 30, 2026, https://www.mdpi.com/2218-1997/9/12/501
23. Cosmic Microwave Background Radiation within the Zwicky Tired Light Hypothesis - arXiv, accessed May 30, 2026, https://arxiv.org/html/2504.10510v1
24. Determinism - Rapier, accessed May 30, 2026, https://rapier.rs/docs/user_guides/javascript/determinism/
25. Unit testing for deterministic systems : r/Unity3D - Reddit, accessed May 30, 2026, https://www.reddit.com/r/Unity3D/comments/1qbtaiq/unit_testing_for_deterministic_systems/
26. How to compare numbers correctly in JavaScript - DEV Community, accessed May 30, 2026, https://dev.to/alldanielscott/how-to-compare-numbers-correctly-in-javascript-1l4i
27. Number.EPSILON - JavaScript - MDN Web Docs, accessed May 30, 2026, https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number/EPSILON
28. Expect - Jest, accessed May 30, 2026, https://jestjs.io/docs/next/expect
29. How to check strict equality of floating numbers in jest - Stack Overflow, accessed May 30, 2026, https://stackoverflow.com/questions/60416076/how-to-check-strict-equality-of-floating-numbers-in-jest
30. Close comparison between float numbers within objects · Issue #3654 · jestjs/jest - GitHub, accessed May 30, 2026, https://github.com/jestjs/jest/issues/3654
31. Deterministic Physics in TS: Why I Wrote a Fixed-Point Engine - DEV Community, accessed May 30, 2026,

https://dev.to/shaisrc/deterministic-physics-in-ts-why-i-wrote-a-fixed-point-engine-4b0l
32. Performance Comparison of WebGPU and WebGL for 2D Particle Systems on the Web - DiVA portal, accessed May 30, 2026, https://www.diva-portal.org/smash/get/diva2:1945245/FULLTEXT02
33. Efficient particle system in javascript? (WebGL) - WebGL2 Fundamentals, accessed May 30, 2026, https://webgl2fundamentals.org/webgl/lessons/webgl-qna-efficient-particle-system-in-javascript---webgl-.html