Qwen

Implementing Neurorights-Compliant Harm Reduction in Unreal Engine: A Blueprint for the Evil-Decay Framework

Architectural Foundations: Neurorights-Informed System Design

The development of the Evil_Decay framework is predicated on a foundational architecture that prioritizes user safety, privacy, and cognitive liberty above all else. This design philosophy is not an optional feature but a core constraint embedded within the system's logic, nomenclature, and operational protocols. The primary objective is to create a deterministic, neurorights-compliant mechanism for governing experiences in Extended Reality (XR) environments, particularly during sleep-gated states. The entire structure is built upon a set of non-negotiable invariants designed to prevent misuse and protect the user's mental integrity. These principles are derived from the nascent field of neurotechnology ethics, which seeks to establish guardrails for technologies that interface directly with the human brain [46] [53]. The most critical invariant is `nopersonscoring`, which mandates that the concept of "evil-rate" must never be tied to the identity, beliefs, or moral standing of an individual user . Instead, it is explicitly defined as a zone-level or scene-level property, analogous to concepts like a `nightmareindex` or `psychriskscore` [35]. This distinction is paramount; it ensures the metric functions purely as a measure of structural environmental risk, not as a tool for personal judgment, commercial profiling, or punitive action [47] [52]. By anchoring the metric to the environment, the framework aligns with global ethical guidelines, such as UNESCO's Recommendation on the Ethics of Neurotechnology, which calls for robust legal frameworks to govern neural data and prohibits its use for purposes beyond direct health applications [42] [43] [62].

To translate these high-level principles into enforceable technical rules, the framework proposes treating key constraints as formal invariants . These include `unitintervalscores` (ensuring all metrics are clamped between 0 and 1), `nopersonscoring`, and `nonwaivableflags` . An integration harness, operating as a server-side service, would be responsible for validating all incoming data streams against these invariants before any data is permitted to influence the Unreal Engine's state . If a

violation is detected—for instance, an attempt to inject a person-scoring variable or a metric outside the valid range—the system would block the transmission and log the event, creating a verifiable audit trail . This transforms abstract ethical ideals into concrete, automated checks, providing a layer of procedural security that is far more reliable than voluntary adherence. This approach is crucial for building trust and ensuring long-term compliance, especially as the technology matures and regulatory scrutiny increases. The requirement for `nonwaivableflags` further solidifies this protection, ensuring that users cannot opt out of these fundamental safeguards, thereby upholding the principle that certain rights, such as mental privacy, are inherent and non-transferable [46]. This design anticipates future legal requirements and establishes a precedent for ethical neurotechnology deployment [40] [41]. The entire system is structured as a multi-layered safety net, where each component serves to reinforce the others, culminating in a governance model that is both technologically sophisticated and ethically grounded.

The system's architecture is also designed for modularity and portability, with a clear separation of concerns between the external data acquisition and processing pipeline and the internal Unreal Engine implementation. The external microservice, exemplified by the `unreal_evil_decay_bridge.js` module, is responsible for decoding raw biosignals and computing the necessary scalar values . This service acts as a secure gateway, ensuring that only anonymized, normalized, and validated data ever reaches the game engine. Unreal itself is treated as a passive recipient of these scalar inputs, using them to drive its own internal logic for rendering, routing, and governance without ever having access to the underlying physiological data or attempting to perform complex computations itself . This client-server model enhances security by minimizing the attack surface within the engine and simplifies the integration process. The communication between the microservice and Unreal can be handled through standard network protocols like HTTP or Open Sound Control (OSC), leveraging existing Unreal plugins for UDP sockets [1] [2] [31]. This decoupled architecture allows the BCI processing algorithms to be updated or replaced independently of the XR environment, fostering innovation while maintaining a stable and safe interface to the user. The ultimate goal is to create a system that is not only effective at reducing harm but is also transparent, auditable, and respectful of the user's autonomy, setting a new standard for the ethical application of immersive technologies.

Principle	Description	Technical Implementation
No Person-Level Scoring	The evil-rate metric must be environment-centric and never reference or score the user's identity, beliefs, or history .	Computation based solely on zone metadata (`nightmareIndex`) and scalar feeds (`arousal`, `psychRisk`). No personal identifiers are used in calculations [52] .
Non-Waivable Cognitive Liberty	Core safety protections, including mental privacy and the absence of punitive scoring, are fundamental rights and cannot be opted out of by the user .	`nonwaivableflags` are enforced by a server-side integration harness that blocks any attempts to disable these safeguards [46] .
Unit Interval Scores	All computed metrics, including evil-rate and eligibility, must be normalized to a value between 0 and 1, inclusive .	Inputs are clamped using `Math.min(1, Math.max(0, value))` before being used in any calculations [15] .
Invariance Enforcement	Key rules like `nopersonscoring` and `unitintervalscores` are treated as formal invariants that must be checked before data is processed .	A dedicated integration harness runs these checks. Violations result in blocked data transmission and logging for audit purposes [50] .
Cryptographic Grounding	Each epoch's calculation must be provably linked to its input features without exposing raw data, ensuring data integrity and verifiability .	A `hexCommit` is generated using a hash function (e.g., XXH-like) over a vector of integer-scaled features, providing a unique proof-of-commitment [10] .

This architectural blueprint provides a comprehensive foundation for developing a trustworthy and effective harm reduction framework. It moves beyond theoretical discussions of ethics by embedding them directly into the system's code and operational procedures. The emphasis on procedural enforcement, modular design, and strict data handling protocols ensures that the framework is not just well-intentioned but technically robust. As the research progresses, this architecture will serve as the guiding structure for empirical testing, implementation, and eventual deployment, ensuring that the pursuit of safer XR experiences never comes at the cost of user rights.

Data Ingestion and Scalarization Pipeline

The successful implementation of the Evil_Decay framework hinges on a clean, secure, and efficient data ingestion pipeline that translates raw physiological signals from a user's BCI into a stream of normalized scalar indices consumable by Unreal Engine. This process is the critical bridge between the user's biological state and the virtual environment's governance system. The design of this pipeline is meticulously crafted to adhere to neurorights principles, ensuring that no personally identifiable information or raw neural data ever leaves the secure processing environment. The core of this pipeline is a JavaScript-based bridge module, `unreal_evil_decay_bridge.js`, which encapsulates the entire scalarization logic . This module acts as a deterministic function, taking a structured input object containing various biosignal posteriors and other metrics,

and returning a JSON object with the requisite output scalars and cryptographic proofs . This functional approach is ideal for integration, as it can be easily exposed to Unreal Engine via an HTTP call from a custom C++ plugin or a Blueprint-native web request node.

The input to the `evaluateEpoch()` function represents a single 30-second epoch of data, mirroring the granularity of sleep staging analysis [6] [7] . The primary input is the `posteriors` object, which contains the classifier's confidence scores for each sleep stage: `pWake`, `pN1`, `pN2`, `pN3`, and `pREM` . These floating-point values, typically in the range [0,1], represent the probability of the user being in each respective state. Alongside these, the function receives several other critical scalar inputs: `sleeptoken` (S), `psychRisk` (R), `enstasis` (Es), `nightmareIndex`, `arousal`, and a boolean flag `invariantsOk` . These inputs collectively form a holistic snapshot of the user's state, blending their current sleep stage with environmental context and higher-level psychological indicators. The `posteriors` are immediately passed to the `computeSleepIndices` function, which normalizes them into the five sleep-stage indices: `IN1`, `IN2`, `IN3`, `IN2.N3`, and `I?` . This normalization step is crucial; it creates a standardized representation of the user's consciousness level that is independent of the absolute power of the underlying EEG signals. The denominator for this calculation is the sum of all posterior probabilities, which prevents artifacts if one class has a very low probability . The resulting indices are clamped to the [0,1] interval to ensure they remain valid inputs for subsequent calculations .

Once the sleep indices are computed, the next step is to calculate the eligibility gate, `E`. This is done by calling the `computeEligibility` function with the `sleeptoken`, `psychRisk`, and `enstasis` inputs . Each of these values is also clamped to [0,1], ensuring the final product, `E = S(1-R)Es`, remains a valid eligibility scalar . A high `E` value signifies that the user is in a suitable sleep state (high `S`), is psychologically stable (low `R`), and is in a coherent state (high `Es`), thus making them eligible for passage into potentially hazardous dreamscapes. The final evil-rate is then computed by the `computeEvilMetric` function, which takes the `nightmareIndex` from the Unreal zone and the previously calculated `arousal`, `psychRisk`, `eligibilityE`, and `invariantsOk` flags . This function combines the `structuralRisk` (the danger posed by the environment under current arousal/stress) and `gateTension` (the danger posed by attempting to enter a risky area when gates are weak) into the final `evilrate` score . Throughout this entire process, the bridge module maintains a strict separation of data types and formats. The only data passed to Unreal is a simple JSON object containing numbers and a string (the `hexCommit`) . This design completely isolates the Unreal Engine from any complex signal processing or decision-making logic, allowing it to focus solely on rendering and responding to the pre-computed safety metrics.

The communication protocol between this bridge microservice and the Unreal Engine is another critical aspect of the pipeline. While the provided context suggests using HTTP calls, Unreal's networking capabilities offer several viable options. The Open Sound Control (OSC) plugin provides a native and type-safe library for sending and receiving OSC messages, which are often transported over UDP [31] [55]. This protocol is lightweight and well-suited for real-time streaming of data like the JSON payload from the bridge [1]. A Blueprint could be configured to act as an OSC client, listening for messages from the microservice's IP address and port. Upon receiving a message, it could parse the JSON string and update corresponding variables in the game world. Alternatively, a simpler UDP socket implementation could be used, leveraging Blueprint nodes for sending and receiving datagrams [2]. For developers with C++ experience, a custom plugin could be developed to create a more tightly integrated connection, perhaps using TCP for guaranteed delivery or a more optimized binary format instead of JSON. Regardless of the specific protocol chosen, the underlying principle remains the same: a separate, dedicated process handles the sensitive BCI data and computation, and communicates only the necessary, sanitized results to the game engine. This architecture not only enhances security but also improves scalability and maintainability, as changes to the BCI processing algorithm or the communication protocol do not require recompiling or modifying the Unreal project itself.

Zone-Level Harm Quantification: The Evil-Rate Metric

At the heart of the Evil_Decay framework lies the "evil-rate" metric, a quantitative measure of harm propensity designed to operate exclusively at the zone level within the Unreal Engine environment. Its definition is deliberately engineered to be free of any person-level attributes, adhering strictly to the `nopersonscoring` invariant . The metric is not a judgment of the user but a dynamic assessment of the interaction between the user's current physiological state and the inherent properties of the virtual space they occupy. The formula for evil-rate is a weighted combination of two distinct components: `structuralRisk` and `gateTension` . This dual-component structure allows for nuanced control over different facets of safety. `structuralRisk` quantifies the intrinsic danger of the environment itself, acting as a baseline hazard level. It is calculated as the product of the zone's `nightmareindex` and the average of the user's physiological arousal and psychological risk scores: $\text{structuralRisk} = \text{nightmareindex} \cdot 0.5(\text{arousal} + \text{psychrisk})$. The `nightmareindex` is a piece of metadata assigned to each Unreal level or volume, representing its predisposition towards generating disturbing or threatening content . The `arousalmeter` and

psychriskscore are scalar inputs derived from the user's biosignals, reflecting their current level of stress or distress . By multiplying these factors, the formula correctly identifies that a high-risk environment is only dangerous when the user is physiologically primed to react to it. A user in a low-arousal, low-stress state (arousal=0, psychrisk=0) within a highly intense zone (nightmareindex=1) would still have a structuralRisk of zero.

The second component, gateTension, models a different kind of risk: the danger associated with attempting to access a hazardous area when the protective gating mechanisms are compromised. It is defined as $\text{gateTension} = \text{nightmareindex} \cdot (1 - E)$. Here, the nightmareindex again represents the zone's inherent danger, but the risk is modulated by (1 - E), where E is the eligibility scalar computed from the sleep token, psych risk, and enstasis scores . When the user is deemed fully eligible (E approaches 1), the term (1 - E) approaches zero, effectively neutralizing the gateTension and indicating a safe passage. Conversely, if the user's state deteriorates (e.g., psychrisk increases, lowering R and thus E), the gateTension rises, signaling increased risk even within the same physical location in the dreamspace. The final evil-rate is a linear blend of these two components: $\text{evilrate} = 0.6 \cdot \text{structuralRisk} + 0.4 \cdot \text{gateTension}$. The coefficients (0.6 and 0.4) are tunable parameters that allow developers to prioritize either environmental safety or gating integrity based on the specific therapeutic goals of the XR session. The entire result is clamped to the unit interval, [0,1], satisfying the unitintervalscores invariant . This entire computation is performed by the external bridge module before the value is sent to Unreal, ensuring the engine only receives the final, ready-to-use metric .

This zone-centric approach offers significant advantages for both safety and development. From a safety perspective, it prevents the creation of persistent, stigmatizing user profiles based on negative experiences. A user who has a bad dream in a particular zone does not receive a penalty score; rather, the system learns that the combination of that zone's nightmareindex and the user's physiological response at that moment resulted in a high evil-rate, triggering a corrective action like a forced teleport to a safe room. This focuses the intervention on the immediate environmental and physiological context, not on the user's character. For developers, it provides a powerful tool for procedurally designing and balancing dreamscapes. They can assign nightmareindex values to different areas, creating zones of varying difficulty or therapeutic challenge. The evil-rate metric then becomes the arbiter of whether a user is prepared to enter those zones, enabling adaptive and personalized journeys through the XR environment. The system's governance logic in Unreal can be built around simple thresholds on this metric. For example, if evilrate exceeds a moderate threshold, the system might apply intensity caps to visual and auditory stimuli. If it crosses a higher threshold, it could trigger a

forced teleport to a predefined `safe_room_anchor` location within the level . If the metric indicates a rapidly escalating threat, the governance state could be escalated to `Rogue`, leading to a hard stop in data streaming and a full emergency return to a safe shell environment . This tiered response system, driven by a neurorights-compliant metric, forms the basis of a truly adaptive and responsive XR safety framework.

Component	Formula	Description
Sleep Stage Indices	$IN1=\frac{pN1}{\sum p}, IN2=\frac{pN2}{\sum p}, IN3=\frac{pN3}{\sum p}$ $IN2.N3=0.5\cdot IN2+1.0\cdot IN3$ $I?=1-\max(p\text{Wake}, pN1, pN2, pN3, p\text{REM})$	Derived from sleep stage classifier posteriors. Represents the probability distribution across wakefulness and sleep stages N1, N2, N3, and REM .
Eligibility Gate (E)	$E=S\cdot(1-R)\cdot Es$	A composite score determining eligibility to access certain dreamscapes. High `E` requires a high sleep token (`S`), low psych risk (`R`), and high enstasis (`Es`) .
Structural Risk	structuralRisk=nightmareindex·0.5(1−arousal_psychRisk)	(Measures the intrinsic danger of the zone given the user's current arousal and psychological state. Depends on zone metadata and physiological inputs .
Gate Tension	gateTension=nightmareindex·(1−E)	Measures the risk associated with attempting to access a hazardous zone when the safety gates are weak or compromised .
Final Evil-Rate	evilrate=0.6·structuralRisk+0.4·gateTension	A weighted combination of structural risk and gate tension, clamped to the interval [0, 1]. Used to drive governance actions in Unreal .

Temporal Dynamics and Deterministic Governance

The "Decay" in the Evil_Decay framework refers to its sophisticated modeling of how harm propensity evolves over time following an intervention. This temporal dimension is crucial for moving from a reactive safety system to a predictive and preventative one. The baseline model for this evolution is exponential decay with a delay, formally defined as $e(t)=e_0$ for $t \leq \tau$, and $e(t)=e_0 e^{-\lambda(t-\tau)}$ for t > \tau . In this equation, e_0 is the initial evil-rate at the moment an intervention is triggered, t is the time elapsed since the intervention, λ (lambda) is the decay rate constant that determines the speed of the reduction, and τ (tau) is the delay, which accounts for the biological and computational lag before the effect of the intervention becomes apparent . This delayed start acknowledges that simply applying a clamp or changing a zone's state does not instantly erase the preceding buildup of risk. The secondary analysis phase of the research aims to empirically fit this model—and potentially simpler or more complex alternatives like hysteresis or threshold models—to logged time-series data to determine the most accurate representation of the system's behavior . Choosing the simplest model that adequately fits the data is a key part of this process, ensuring the framework remains computationally efficient while being predictive .

A particularly advanced feature of the framework is its implementation of a hysteresis model to handle situations where the system's safety invariants fail, such as entering a "rogue" state . Instead of resetting the evil-rate to a baseline when conditions improve, the system allows for a partial rebound. The post-failure rate is modeled as $e'(t) = e(t) + \rho(e_0 - e(t))$, where $e(t)$ is the decayed rate before the failure, e_0 is the original peak rate, and ρ (rho) is the rebound factor in the range [0,1] . This parameter controls the fraction of the original gap between the current rate and the peak that is recovered. A ρ of 0 means no rebound (the rate stays at the decayed level), while a ρ of 1 means a full recovery towards the peak (though it is capped at e_0) . This hysteresis effect introduces resilience into the system, preventing it from becoming complacent after a period of low risk. It reflects the real-world understanding that recovering from a state of high distress or instability is not instantaneous and may leave lingering effects. This model ensures that even after a temporary crisis is resolved, the system remains in a heightened state of vigilance until sufficient time has passed and the risk has genuinely dissipated, as dictated by the exponential decay component of the model.

Once computed, the temporal dynamics of the evil-rate feed directly into Unreal's deterministic governance system. This system uses the `evilrate` value, along with other state variables, to make real-time decisions about the user's experience. The governance logic is driven by a series of thresholds that map the continuous evil-rate metric to discrete actions. These actions are designed to mitigate harm progressively, escalating only when necessary. The primary governance states are `Safe`, `Suspicious`, and `Rogue` . In the `Safe` state, the user can explore normally. If the `evilrate` crosses a first, moderate threshold, the system transitions to `Suspicious`. In this state, actions might include clamping the intensity of sensory stimuli (e.g., dimming lights, lowering sound volume) or introducing calming environmental elements. If the `evilrate` continues to rise and crosses a higher threshold, the system enters the `Rogue` state. This triggers more forceful interventions, such as a forced teleport to a predefined `safe_room_anchor` location or a complete throttling of the XR stream . The hex commitment (`hexCommit`) associated with each epoch's evil-rate calculation provides an immutable record of the decision-making process, allowing for detailed post-session audits to understand why a particular action was taken . This entire governance loop operates entirely within Unreal, consuming the scalar outputs from the external bridge and using Unreal's native event-driven architecture to execute the appropriate responses . This ensures that the XR environment itself is the ultimate agent of safety, dynamically adapting its behavior to the user's physiological state in a predictable and neurorights-compliant manner.

Empirical Validation and Procedural Enforcement

The credibility and efficacy of the Evil_Decay framework depend on a rigorous, two-layered validation process that combines empirical measurement of real-world outcomes with procedural enforcement of technical invariants. The primary emphasis of this validation is to empirically verify that the implemented neurorights-compliant interventions—namely the E-gate, mid-N2/N3 timing, psychrisk/arousal caps, and AuraBounds—actually lead to a measurable reduction in distress, awakenings, and nightmare frequency, all while preserving the integrity of deep sleep . To achieve this, the research plan includes conducting controlled pilot studies with a pre-post intervention comparison design. Baseline data on user distress levels, self-reported nightmare frequency, and objective sleep metrics (such as the duration of N2 and N3 stages) will be collected before the framework is enabled. After a period of adaptation, the Evil_Decay framework will be activated, and the same metrics will be collected for a comparable duration. A statistical comparison between the pre- and post-intervention periods will provide the primary evidence for the framework's effectiveness. This approach directly addresses the central research question by linking the technical implementation to tangible human outcomes.

To ensure the evil-rate metric itself is a reliable predictor of harm, a secondary validation task involves creating calibration curves . This process entails binning the logged evil-rate values (e.g., 0.0-0.1, 0.1-0.2, etc.) and then calculating the empirical frequency of adverse events (e.g., user-initiated interruptions, reports of distress, or automatic safety interventions) within each bin. A well-calibrated metric will show a monotonic increase in incident frequency as the evil-rate bins increase; for instance, events in the 0.8-1.0 bin should occur far more frequently than those in the 0.0-0.1 bin. Deviations from this pattern would indicate that the metric is miscalibrated and needs refinement [50] . This method, borrowed from fields like machine learning and weather forecasting, provides a quantitative way to assess the quality of the evil-rate prediction [49] . The goal is to demonstrate that the evil-rate is not just a theoretical construct but a practical tool whose predictions align with observed reality. This calibration loop is essential for building trust in the system's ability to anticipate and prevent harm before it occurs.

Beyond empirical validation, a critical layer of assurance comes from procedural enforcement through invariants. As previously discussed, the system treats rules like `evilrate ∈ [0,1]` and `nopersonscoring` as non-negotiable invariants . This is not merely a coding convention but a system-enforced policy. An integration harness, running as a separate service, is responsible for intercepting all data flows and checking them against these rules. This harness would extend the existing invariant checks to

specifically include `evilrate_in_unit_interval`. Any data batch that fails this check—a scenario where evil-rate is, for example, 1.5 or -0.1—would be rejected, quarantined, and logged. This creates an automated audit trail of any violations, whether accidental or intentional, providing a clear record for debugging and accountability [59]. The Unreal Engine itself would act as a compliant client, simply receiving and honoring the "allowed/blocked" decisions and governance flags from the harness without ever having the authority to override them . This strict separation of concerns ensures that the safety logic remains centralized, verifiable, and impervious to manipulation from within the game engine. This procedural rigor is what distinguishes the Evil_Decay framework from a simple collection of scripts; it is a formal system of governance built on enforceable rules, combining the best of empirical science with the discipline of computer science.

Practical Implementation and Cross-Engine Conformance

The path to achieving the research goal is centered on Unreal Engine as the primary integration target due to its mature toolset and suitability for creating immersive XR environments [3] [4]. The implementation strategy is pragmatic, focusing on delivering a set of concrete, actionable artifacts that will facilitate adoption and ensure consistency. The first and most important deliverable is an ALN (Agreement Language Notation) spec shard, such as `xr-dream.engine-evil-decay.v1.aln` . This file will serve as the canonical specification, formally defining every scalar field used in the pipeline—including `IN1`, `IN2`, `IN3`, `E`, `evilrate`, and the associated invariants . By providing a machine-readable contract for the data exchange, the ALN shard eliminates ambiguity and ensures that all components, regardless of the engine they run in, interpret the data in the exact same way. This is the foundational artifact for achieving the second major deliverable: a cross-engine conformance suite . This suite will consist of a series of test cases, likely in CSV or ALN format, containing a diverse set of input traces (combinations of sleep indices, risk scores, and zone metadata). Godot, Unreal, and Omniverse engines will all be required to process these identical traces and produce bitwise-identical outputs for the `E` gate decisions and the final `evilrate` values, and consequently, the same roaming or blocking actions . Passing this conformance suite will be a prerequisite for any engine to be considered compatible with the Evil_Decay framework, guaranteeing deterministic and reproducible behavior across platforms.

To lower the barrier to entry for researchers and operators who may not have extensive C++ programming experience, the third key deliverable is a Blueprint-compatible plugin or library . This plugin would package the functionality of the `unreal_evil_decay_bridge.js` into a set of easy-to-use Blueprint nodes. For example, it could provide a `DEPRECATED_EvaluateEpoch` node that accepts the various scalar inputs and outputs the calculated `E` gate, `evilrate`, and `hexCommit` string. This would allow users to wire up the entire Evil_Decay logic directly within Unreal's visual scripting environment, without needing to write any code [19] . The plugin would also expose the governance logic, providing nodes for checking the evil-rate against thresholds and triggering actions like teleporting to a safe zone or adjusting material parameters [68] [69] . This Blueprint-centric approach democratizes access to the technology, enabling a wider community of creators and scientists to build upon the framework. The vision is to create a "hands-free" operation mode where an operator can configure experiments through a UI, and the underlying system autonomously collects data once the BCI-to-Unreal pipeline is established, relying on the robustness of the bridge module and the Unreal-side plugin .

The table below outlines the planned deliverables and their specific contributions to the project's success.

Deliverable	Format/ Type	Purpose	Key Components
ALN Spec Shard	`.aln` file	Provides a formal, machine-readable contract for all data fields and invariants. Ensures semantic consistency across all implementations.	Definitions for `E`, `evilrate`, `IN1`, `IN2`, `IN3`, `IN2.N3`, `Iq`, `hexCommit`; declarations of `evilrate ∈ [0,1]`, `nopersonscoring`, `nonwaivableflags`.
Cross-Engine Conformance Suite	Test cases (CSV/ALN files)	Verifies that different game engines produce identical, deterministic outputs from the same input traces. Guarantees portability and reliability.	A diverse set of input vectors covering edge cases for sleep indices, risk scores, and zone metadata. Expected `E` and `evilrate` outputs for each.
Blueprint-Compatible Plugin	Unreal Engine Plugin	Enables rapid prototyping and use by non-programmers. Integrates the BCI scalar pipeline and governance logic directly into Unreal's visual scripting system.	Blueprint nodes for `evaluateEpoch`, evil-rate threshold checks, and triggering governance actions (teleport, intensity cap).
Hex Commit Proof-of-Commitment	64-bit hash (e.g., `0x...`)	Provides cryptographic grounding, proving that a calculation was based on a specific set of features without exposing raw data. Enhances auditability and security.	Hash of a vector of scaled integer features derived from sleep indices [10] .

In summary, the practical implementation plan is comprehensive and forward-looking. It balances the need for deep technical specifications like the ALN shard with the goal of broad accessibility through a Blueprint plugin. The emphasis on a cross-engine conformance suite is a particularly strong point, as it builds a portable and interoperable

system from the outset, avoiding vendor lock-in and promoting healthy competition and collaboration among different engine developers. By focusing on these tangible deliverables, the research will not only validate the core Evil_Decay concept but also produce a toolkit that can be used by the wider XR research and development community to build safer, more ethical immersive experiences.

Reference

1. OSC Plugin Overview for Unreal Engine https://dev.epicgames.com/documentation/en-us/unreal-engine/osc-plugin-overview-for-unreal-engine
2. How To use Blueprint to send/receive UDP messages? https://forums.unrealengine.com/t/how-to-use-blueprint-to-send-receive-udp-messages/66207
3. Unreal Engine 5.4 Release Notes https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5.4-release-notes?application_version=5.4
4. Virtual, Augmented and Mixed Reality - Springer Link https://link.springer.com/content/pdf/10.1007/978-3-031-61044-8.pdf
5. hw3_stats_google_1gram.txt https://www.cs.cmu.edu/~roni/11761/2017_fall_assignments/hw3_stats_google_1gram.txt
6. Validation of a sleep staging classification model for ... https://pmc.ncbi.nlm.nih.gov/articles/PMC11145037/
7. Validation Framework for Sleep Stage Scoring in Wearable ... https://pmc.ncbi.nlm.nih.gov/articles/PMC8161815/
8. Lenka Lhotska · Lucie Sukupova Igor Lacković https://link.springer.com/content/pdf/10.1007/978-981-10-9023-3.pdf
9. Civ Bci R19 | PDF | Internet Of Things | Smart Grid https://www.scribd.com/document/906025539/CIV-BCI-R19
10. On the mathematical validity of the Higuchi method https://www.sciencedirect.com/science/article/abs/pii/S0167278919303859
11. Nonlinear EEG analysis based on a neural mass model https://www.researchgate.net/publication/12709366_Nonlinear_EEG_analysis_based_on_a_neural_mass_model
12. A Study of Cardiac Hysteresis During Physical Stress Test https://arxiv.org/pdf/2410.19667

13. Finite-time complete periodic synchronization of memristive ... https://www.nature.com/articles/s41598-023-37737-2.pdf
14. Simulating Debris Flow and Levee Formation in the 2D ... https://agupubs.onlinelibrary.wiley.com/doi/full/10.1029/2022EA002590
15. Synchronization in STDP-driven memristive neural networks ... https://link.springer.com/article/10.1007/s10867-023-09642-2
16. 虚幻引擎5.3版本说明 https://dev.epicgames.com/documentation/zh-cn/unreal-engine/unreal-engine-5.3-release-notes?application_version=5.3
17. How to get streaming level blueprint's tags? https://forums.unrealengine.com/t/how-to-get-streaming-level-blueprints-tags/2523157
18. Unreal Engine 5 Console Variables and Commands https://www.cnblogs.com/kekec/p/17841097.html
19. Unreal Engine Blueprint API Reference https://dev.epicgames.com/documentation/en-us/unreal-engine/BlueprintAPI
20. UNREAL ENGINE 4.12 RELEASED! 转载 https://blog.csdn.net/pizi0475/article/details/51567315
21. Digital Transformation https://digitalreality.ieee.org/wp-content/uploads/2025/09/DRI_White_Paper_-_Digital_Transformation_-_Final_25March21.pdf
22. A Data-Driven Systematic Review of the Metaverse in ... https://www.sciencedirect.com/org/science/article/pii/S152614922500270X
23. Next-generation probes, particles, and proteins for neural ... https://pmc.ncbi.nlm.nih.gov/articles/PMC5466371/
24. Augmented Reality as a Countermeasure for Sleep ... https://www.researchgate.net/publication/290472759_Augmented_Reality_as_a_Countermeasure_for_Sleep_Deprivation
25. Skin-Integrated Soft Wearable XR Interfaces for Seamless and ... https://pubs.acs.org/doi/10.1021/acs.chemrev.4c00966
26. The use of extended reality (XR) for people with moderate ... https://journals.sagepub.com/doi/10.3233/TAD-210363
27. Prolonged sleep deprivation induces a cytokine-storm-like ... https://www.sciencedirect.com/science/article/pii/S0092867423011765
28. Neural–Computer Interfaces: Theory, Practice, Perspectives https://www.mdpi.com/2076-3417/15/16/8900
29. How Build Brain | PDF https://www.scribd.com/doc/232674373/How-Build-Brain
30. Clinic: Better Sleep | PDF https://www.scribd.com/document/820627872/Mayo-Clinic-Guide-to-Better-Sleep-Find-relief-from-insomnia-sleep-apnea-and-other-sleep-disorders-Jan-7-2025-9798887700496-Mayo-Clinic-Pres

31. Unreal Engine 4.23 released! https://www2.unrealengine.com/blog/unreal-engine-4-23-released
32. 5 https://worksheets.codalab.org/rest/bundles/0xd74f36104e7244e8ad99022123e78884/contents/blob/frequent-classes
33. Dicionario portugues https://www.academia.edu/32592435/Dicionario_portugues
34. Benchmarking Automated Sleep Staging Algorithms for ... https://pmc.ncbi.nlm.nih.gov/articles/PMC12592834/
35. The Nightmare Disorder Index: Development and Initial ... https://www.semanticscholar.org/paper/The-Nightmare-Disorder-Index%3A-Development-and-in-a-Dietch-Taylor/02002b2d598969e53d087dd80823f8ab582b132c
36. Overview of a Sleep Monitoring Protocol for a Large Natural ... https://pmc.ncbi.nlm.nih.gov/articles/PMC10163293/
37. Applications of Long Short-Term Memory (LSTM) Networks in ... https://pmc.ncbi.nlm.nih.gov/articles/PMC11435440/
38. Perspectives on adaptive dynamical systems https://arxiv.org/pdf/2303.01459
39. Edge AI for Smart Cities: Foundations, Challenges, and ... https://www.mdpi.com/2624-6511/8/6/211
40. Ethics of neurotechnology: UNESCO adopts the first global ... https://www.unesco.org/en/articles/ethics-neurotechnology-unesco-adopts-first-global-standard-cutting-edge-technology
41. Ethics of neurotechnology https://www.unesco.org/en/ethics-neurotech
42. Draft text of the Recommendation on the Ethics ... https://unesdoc.unesco.org/ark:/48223/pf0000393395
43. First draft of the Recommendation on the Ethics ... https://unesdoc.unesco.org/ark:/48223/pf0000391444
44. Towards an International Instrument https://www.unesco.org/en/ethics-neurotech/recommendation
45. The Ethics of Neurotechnology: UNESCO appoints ... https://www.unesco.org/en/articles/ethics-neurotechnology-unesco-appoints-international-expert-group-prepare-new-global-standard
46. Advancing neurotechnology while protecting the human ... https://www.unesco.org/en/articles/advancing-neurotechnology-while-protecting-human-brain-unesco-global-ethical-framework
47. Metaverse - European Parliament https://www.europarl.europa.eu/RegData/etudes/STUD/2023/751222/IPOL_STU(2023)751222_EN.pdf
48. 31/12/2024 Annual Report 2024 (PDF) https://www.reply.com/contents/REP25_Bilancio_ENG_2024.pdf

49. words_SG_upto2020.txt https://zenodo.org/record/5516252/files/words_SG_upto2020.txt
50. Mathematical Foundations of Deep Learning https://hal.science/hal-04928560v2/file/Sourangshu_Ghosh_IISc_Bangalore_Mathematical_Foundations_of_Deep_Learning_Version_2.pdf
51. 首届中英三校博士生联合论坛 暨2025年 ... https://www.xjtlu.edu.cn/wp-content/uploads/2025/04/2025-Symposium-Abstract-Brochure.pdf
52. Recommendation on the Ethics of Artificial Intelligence https://unesdoc.unesco.org/ark:/48223/pf0000380455
53. Draft Recommendation on the Ethics of Neurotechnology https://unesdoc.unesco.org/ark:/48223/pf0000394861
54. Recommendation on the Ethics of Artificial Intelligence https://www.unesco.org/en/articles/recommendation-ethics-artificial-intelligence
55. Simple OSC Output/Input Examples (4.23.1) https://forums.unrealengine.com/t/simple-osc-output-input-examples-4-23-1/133224
56. glove.6B.100d.txt-vocab.txt https://worksheets.codalab.org/rest/bundles/0xadf98bb30a99476ab56ebff3e462d4fa/contents/blob/glove.6B.100d.txt-vocab.txt
57. 333333 23135851162 the 13151942776 of 12997637966 ftp://ftp.cs.princeton.edu/pub/cs226/autocomplete/words-333333.txt
58. bing.txt ftp://ftp.cs.princeton.edu/pub/cs226/autocomplete/bing.txt
59. Final report on the draft Recommendation on the Ethics of ... https://unesdoc.unesco.org/ark:/48223/pf0000393266
60. First draft of a Recommendation on the Ethics ... https://unesdoc.unesco.org/ark:/48223/pf0000391074
61. Ethics of Artificial Intelligence - AI https://www.unesco.org/en/artificial-intelligence/recommendation-ethics
62. UNESCO Adopts First Global Framework on ... https://www.globalpolicywatch.com/2026/01/unesco-adopts-first-global-framework-on-neurotechnology-ethics/
63. 3617623 12|3594859 time|3570693 year|3516017 publisher http://svn.apache.org/repos/asf/lucene/dev/tags/realtime_DWPT_final_2011-05-02/solr/src/test-files/Top50KWiki.utf8
64. Final report on the draft text of the Recommendation ... https://unesdoc.unesco.org/ark:/48223/pf0000393400
65. 5.5 OSC Plugin not working when using localhost / 127.0.0.1 https://forums.unrealengine.com/t/5-5-osc-plugin-not-working-when-using-localhost-127-0-0-1/2124509

66. How can I receive OSC data in real time without need ... https://forums.unrealengine.com/t/how-can-i-receive-osc-data-in-real-time-without-need-to-press-play-into-game-mode/499507
67. Current Affairs Apr-Jun | PDF | Separation Of Powers https://www.scribd.com/document/897504994/Current-Affairs-Apr-Jun
68. How to get the current value of a scalar parameter ... https://forums.unrealengine.com/t/how-to-get-the-current-value-of-a-scalar-parameter-on-a-material-via-blueprint/457812
69. Get Value of a Material Input in Blueprint? - #3 by RyanB https://forums.unrealengine.com/t/get-value-of-a-material-input-in-blueprint/317448/3
70. Scalar Subsurface input handling inconsistent in 4.5 https://forums.unrealengine.com/t/scalar-subsurface-input-handling-inconsistent-in-4-5/296340
71. (PDF) Science Research Papers https://www.academia.edu/43731129/Science_Research_Papers
72. (PDF) May 30th IISTE peer-review journal publication https://www.academia.edu/3692155/May_30th_IISTE_peer_review_journal_publication
73. Proceedings of the Third National Conference on RECENT ... https://www.academia.edu/7358719/Proceedings_of_the_Third_National_Conference_on_RECENT_TRENDS_IN_INFORMATION_AND_COMMUNICATION_TECHNOLOGY