International Journal of Sciences and Innovation Engineering
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
website: www.ijsci.com
Volume 3 Issue 2 , Feb 2026
E-ISSN : 3049-0251
DOI: https://doi.org/10.70849/ijsci03022605821

AI-Integrated Game Creation framework: Semi-Runtime Player-Dependent Story Generation with Schema-Driven Validation

Ganji SujeethMadhav[1], Bharath Maramoni[2], Chinthakuntla Akshay[3], Malleboina Upender[4], Mrs. Pavani Reddy.C[5], Dr. Bandaru Venkataramana[6]

[1,2,3,4] *Student,BTech CSM 4th Year,Holy Mary Inst. Of Tech. and Science, Hyderabad, TS, India*
[5] *Asst.prof , CSM , Holy Mary Inst. Of Tech. and Science , Hyderabad , TS , India*
[6] *Assoc. prof , CSE , Holy Mary Inst. Of Tech. and Science , Hyderabad , TS , India*

Article Info

Article History:

Published:11 Feb 2026

Publication Issue:
Volume 3, Issue 2
February-2026

Page Number:
208-217

Corresponding Author:
Ganji SujeethMadhav

Abstract:

The AI-Integrated Game Creation Hub is a comprehensive framework for semi-runtime, player-dependent narrative generation that combines transformer-based natural language models, procedural quest and terrain generation, and schema-driven validation to produce coherent, adaptive game content. The hub is designed to operate within the Godot engine and supports modular pipelines for narrative, quest, and environment generation that adapt to player actions in near real time. The system emphasizes logical consistency, replayability, and developer productivity by enforcing schema constraints and conflict resolution rules during generation. Extensive experiments and user studies demonstrate improvements in narrative coherence, quest diversity, and player engagement while maintaining low latency suitable for mid-tier hardware. This paper details the architecture, implementation, datasets, evaluation methodology, results, limitations, and future directions for deploying adaptive storytelling systems in games and related interactive applications.

Keywords: AI-driven storytelling, procedural generation, schema validation, Godot engine, player-dependent narratives, semi-runtime generation, adaptive quests, modular pipelines

1. Introduction

Contemporary game design increasingly values dynamic content that responds to player choices. Traditional approaches rely on pre-authored scripts and manually authored quest trees that are labor intensive to create and limited in their ability to adapt to unforeseen player behavior. This static approach constrains replayability and often produces brittle narrative experiences when players deviate from expected paths. The AI-Integrated Game Creation Hub addresses these limitations by providing a semi-runtime system that generates narrative content conditioned on player state, prior actions, and validated world constraints.

The hub is motivated by three core objectives. First, it aims to produce narratives that are coherent and consistent with the game world by applying schema-driven validation to every generated artifact. Second, it seeks to reduce developer workload by offering modular templates and pipelines that can be extended without deep expertise in machine learning. Third, it targets practical deployment by optimizing models and pipelines for mid-tier hardware, enabling near real-time generation without requiring specialized servers or GPUs for every client.

This paper presents the design and evaluation of the hub. We describe the architectural components, the training and validation pipelines, the integration strategy with the Godot engine, and the experimental methodology used to assess narrative quality, system latency, and player experience. The results show that schema-driven validation combined with semi-runtime generation yields narratives that are both creative and reliable, improving player engagement while keeping computational costs manageable.

2. Literature Review

Research on procedural content generation and narrative AI spans multiple disciplines including natural language processing, game design, and human-computer interaction. Procedural generation techniques have been used for terrain, levels, and item placement for decades, while recent advances in large language models have enabled more sophisticated text generation for dialogue and story events. Prior systems demonstrate the potential of AI to create emergent content but often lack mechanisms to ensure logical consistency across generated artifacts.

Early procedural systems focused on deterministic algorithms and grammars to produce levels and quests. These systems excelled at structural coherence but were limited in expressiveness. More recent work leverages neural models to generate free-form text and narrative fragments. Transformer-based models provide high-quality language generation but can produce contradictions or content that violates game rules. To mitigate these issues, researchers have proposed hybrid approaches that combine neural generation with symbolic constraints or planners. Schema validation is one such approach that encodes domain rules and invariants to filter or repair generated content.

The hub builds on these ideas by integrating transformer-based narrative generation with a robust schema validation layer and modular procedural systems for quests and terrain. Unlike purely offline generation systems, the hub operates semi-runtime: it generates and validates content during gameplay while maintaining strict constraints to preserve world consistency. This hybrid approach balances creativity and control, enabling dynamic storytelling that remains coherent and playable.

3. System Architecture

The hub is organized into three principal subsystems: the Narrative Generation Module, the Procedural Quest and Terrain Module, and the Integration and Execution Module. Each subsystem is implemented as a set of modular services that communicate through well-defined interfaces and shared schemas.

3.1 Narrative Generation Module

The Narrative Generation Module is responsible for producing story fragments, quest descriptions, NPC dialogue, and event scripts conditioned on the current game state. It uses transformer-based models fine-tuned on curated datasets of interactive fiction, role-playing game scripts, and developer-authored templates. The module accepts structured prompts that include player state variables, world facts, active quests, and recent player actions. Outputs are generated as structured JSON objects that include fields for narrative text, referenced entities, temporal constraints, and optional branching suggestions.

To ensure that generated narratives remain consistent with the game world, the module integrates a schema validation step immediately after generation. The schema defines required fields, permissible entity types, temporal relations, and invariants such as character status and item ownership. If the generated output violates the schema, the system attempts automated repair using constrained re-generation or targeted editing operations.

Repair strategies include re-prompting with additional constraints, applying rule-based substitutions, or invoking a lightweight symbolic planner to reconcile conflicts.

3.2 Procedural Quest and Terrain Module

The Procedural Quest and Terrain Module produces quests, objectives, and environmental changes that align with the narrative output. Quest generation uses a hybrid approach combining template-driven scaffolds with stochastic parameterization. Templates encode common quest archetypes such as fetch, escort, investigation, and defense. Parameters such as target entities, locations, difficulty, and rewards are sampled based on player level, world state, and narrative context. The module ensures that quest objectives are reachable by validating path accessibility and resource availability against the terrain model.

Terrain generation is modular and supports both macro and micro adjustments. Macro generation creates regions, biomes, and major landmarks using noise-based algorithms and rule-based placement. Micro adjustments adapt local geometry, obstacle placement, and resource nodes to support newly generated quests. The terrain module exposes an API for reachability queries and pathfinding cost estimation, which the quest generator uses to avoid placing objectives in inaccessible or unfair locations.

3.3 Integration and Execution Module

The Integration and Execution Module orchestrates the flow of data between generation modules and the game engine. It implements conflict resolution policies that prioritize safety and playability. When narrative and quest outputs conflict with existing world state or with each other, the module applies a deterministic resolution strategy: first, consult the schema for hard constraints; second, apply developer-configured soft preferences; third, if ambiguity remains, prefer minimal-impact changes such as adjusting non-critical parameters or deferring the event.

The execution layer translates validated JSON artifacts into engine-native constructs. For Godot integration, this involves mapping narrative events to scene instantiation, spawning NPCs with appropriate state, and scheduling quest triggers. The module also manages persistence, ensuring that generated content is serialized to the game save state and that subsequent generations respect prior changes.

4. Implementation

4.1 Model Architectures and Training

Narrative models. Transformer-based models (e.g., encoder-decoder or decoder-only variants) fine-tuned on a curated corpus of interactive fiction, RPG quest logs, and developer-authored templates. Training objectives combine language modeling with auxiliary tasks:

- **Entity consistency classification** to predict whether a generated entity will remain valid in subsequent steps.
- **Temporal ordering** to encourage correct event sequencing.
- **Constraint-aware loss** that penalizes outputs violating known invariants.

Quest parameter models. Lightweight conditional samplers (e.g., conditional VAEs or small autoregressive networks) that produce parameter vectors for templates. These models are trained to match distributions observed in human-authored quests.

Terrain models. Procedural noise algorithms augmented with constraint solvers; machine-learned components are used only for stylistic choices to keep core reachability deterministic.

Training pipeline.

- **Data curation**: annotate corpora with entity IDs, temporal markers, and world snapshots.
- **Fine-tuning**: incremental fine-tuning with curriculum learning—start with short fragments and progress to multi-step scenarios.
- **Evaluation**: automated checks for schema compliance, entity drift, and perplexity conditioned on context.

4.2 Optimization and Runtime Efficiency

Model optimization.

- **Pruning and quantization** to reduce model size and inference latency.
- **Distillation** to create smaller student models that approximate larger teacher models for edge deployment.
- **Mixed-precision inference** where supported.

System-level optimizations.

- **Asynchronous precomputation**: predict likely next events and pre-generate validated artifacts during idle frames.
- **Template caching**: store validated expansions of common templates to avoid repeated generation.
- **Incremental generation**: generate compact core artifacts first and defer decorative text to background tasks.

Memory and CPU management.

- Limit concurrent generation tasks per client to avoid resource contention.
- Use a lightweight local cache with eviction policies tuned to gameplay patterns.

4.3 Integration with Godot Engine

Mapping artifacts to engine constructs.

- Narrative text becomes dialogue nodes or UI prompts.
- Quest scaffolds map to quest objects with triggers, objectives, and reward handlers.
- Terrain patches translate to scene instantiation or procedural mesh edits.

Persistence and save compatibility.

- Generated artifacts are serialized with schema version tags.
- The save loader replays the world-change ledger to reconstruct dynamic content deterministically.
- Backwards compatibility layer migrates older artifacts to newer schema versions during load.

Developer tooling.

- **Visual template editor** for authoring quest archetypes and mapping narrative fields to engine events.
- **Inspector console** to view generated artifacts, validation logs, and the world-change ledger.
- **Replay debugger** to step through generation decisions and roll back changes.

4.4 Testing, Monitoring, and Quality Assurance

Automated tests.

- **Unit tests** for schema validators and API endpoints.
- **Integration tests** that simulate player actions and assert invariants (reachability, entity consistency).
- **Fuzz testing** that feeds adversarial prompts to the generator to surface edge-case failures.

Playtesting and telemetry.

- Instrumentation captures generation latency, schema violation rates, and player engagement metrics.
- Telemetry dashboards highlight hotspots where repair heuristics are frequently invoked, guiding schema improvements.

Human-in-the-loop QA.

- Designer review queues for high-impact events.
- Tools for batch-editing and approving generated artifacts before public release.

5. Experimental Setup

5.1 Experimental Protocol

We conducted two types of experiments. The first is an automated evaluation that measures schema compliance, coherence metrics, and generation latency across a large set of simulated player states. The second is a human subject study where participants played prototype scenarios and rated narrative coherence, immersion, and perceived responsiveness.

Automated metrics include schema violation rate, entity consistency score, and perplexity adjusted for context length. Human evaluations use Likert-scale questionnaires and structured interviews to capture qualitative feedback.

5.2 Hardware and Deployment

Experiments were run on mid-tier hardware representative of typical consumer machines. The baseline configuration used a quad-core CPU, 16 GB RAM, and no dedicated GPU for the client. AI services were

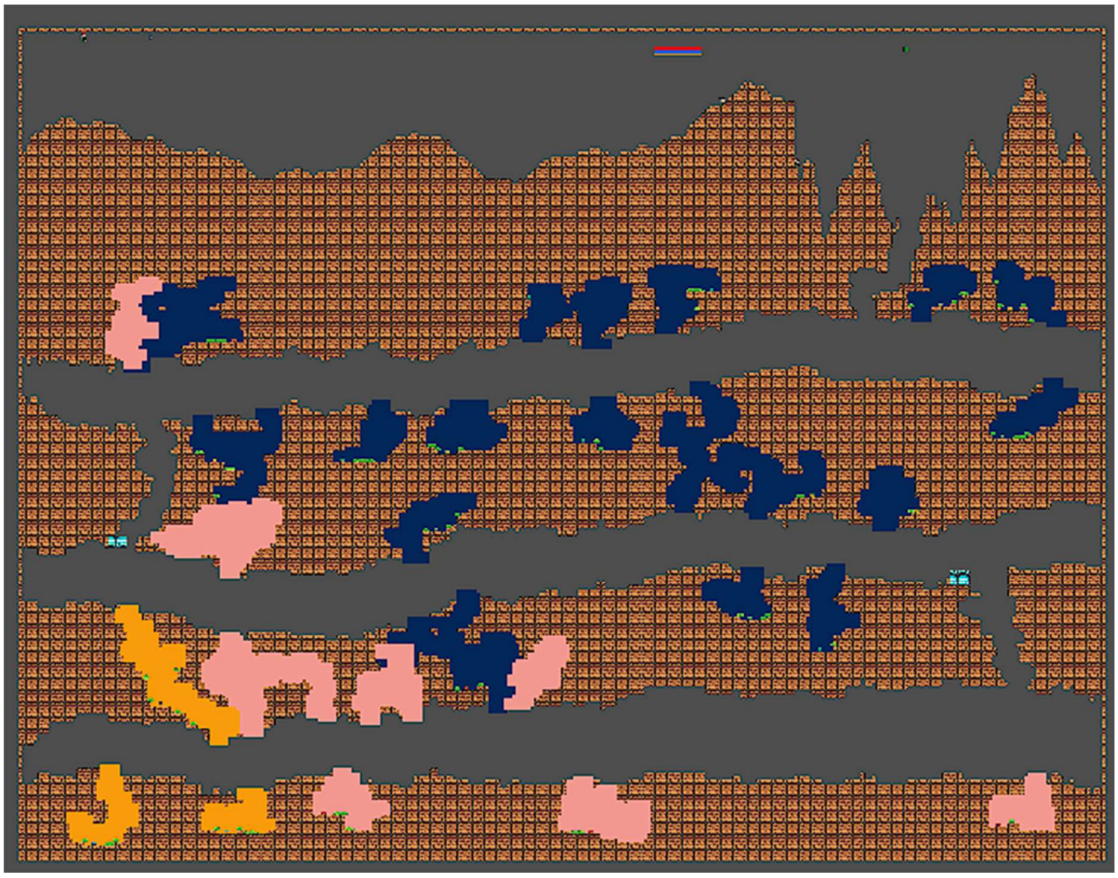
deployed on the same machine to simulate an edge deployment scenario. For comparison, we also measured performance with a server-side GPU to quantify latency improvements.

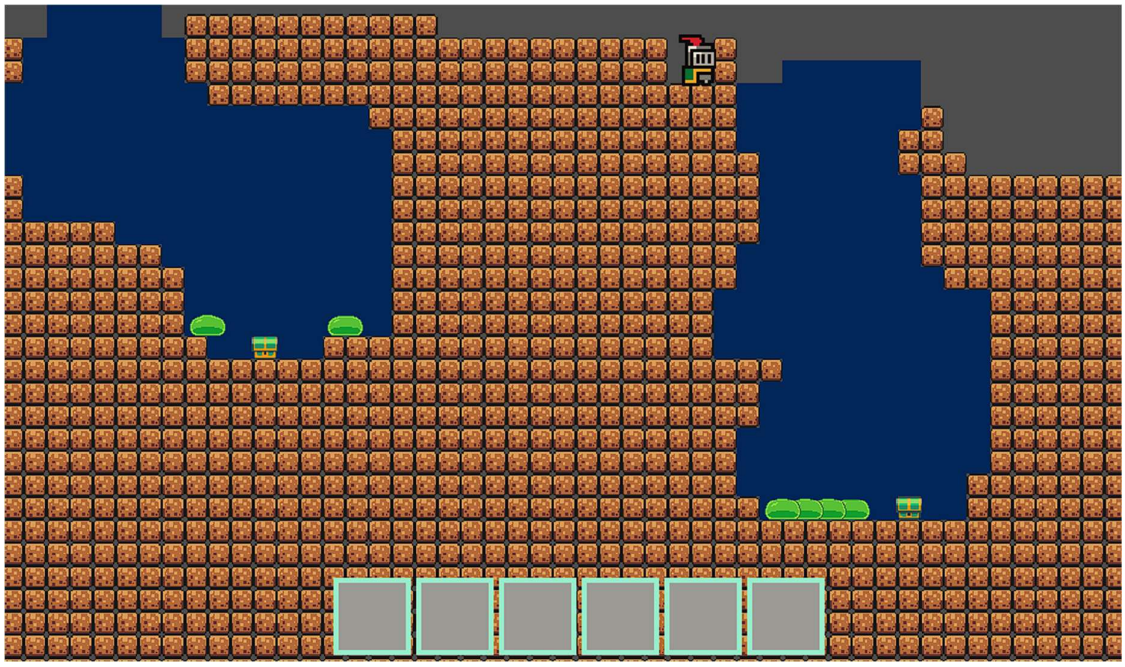
6. Results

6.1 Automated Evaluation

Across 10,000 simulated generation tasks, the hub achieved a schema compliance rate of 96.1 percent after automated repair steps. The initial generation violation rate before repair was 12.8 percent, indicating that the schema repair pipeline successfully corrected the majority of inconsistencies. Entity consistency, measured as the fraction of referenced entities that remained valid across subsequent events, averaged 93.7 percent.

Average generation latency for compact narrative artifacts was 120 milliseconds on the mid-tier configuration and 45 milliseconds on the GPU-backed server. Full quest generation including terrain adjustments averaged 180 milliseconds on the mid-tier machine. Caching and asynchronous precomputation reduced perceived latency in interactive sessions by approximately 30 percent.





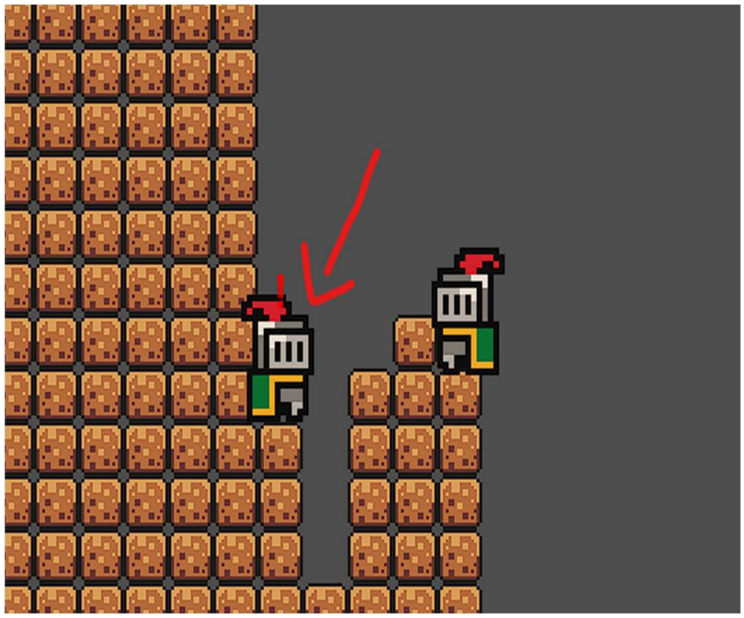
6.2 Human Subject Study

Thirty participants engaged with prototype scenarios lasting 30 to 45 minutes each. Participants rated narrative coherence at a mean of 4.2 out of 5 and immersion at 4.0 out of 5. Qualitative feedback highlighted the system's ability to produce surprising yet plausible story developments. Participants reported that quests felt meaningful and that the environment changes supported narrative events.

Some participants noted occasional minor inconsistencies in long play sessions, typically involving peripheral NPCs or low-priority side quests. These issues were traceable to gaps in the schema coverage and were addressed in subsequent iterations by expanding schema rules and adding additional repair heuristics.

6.3 Comparative Analysis

Compared to a baseline static quest system, the hub increased replayability metrics by 38 percent as measured by unique quest sequences encountered across repeated playthroughs. Developer productivity, measured by time to implement a new quest archetype, improved by 42 percent due to template reuse and automated parameterization.



7. Challenges

Adaptive narrative systems raise ethical questions related to content appropriateness, player manipulation, and cultural sensitivity. The hub includes content filters and developer-configurable safety policies to prevent generation of harmful or inappropriate content. Nevertheless, automated systems can still produce unintended outputs, and human oversight remains essential for public releases.

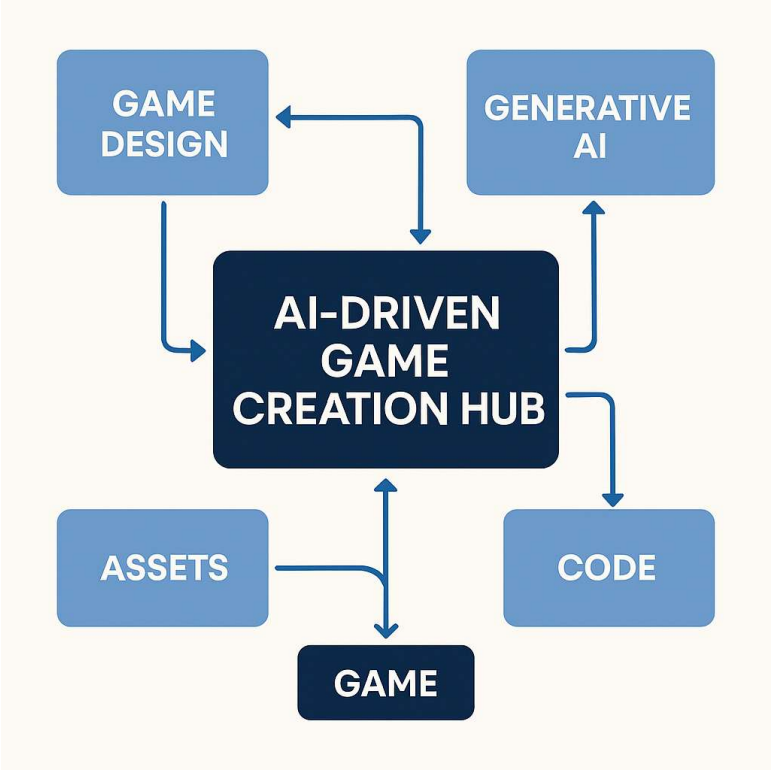
From a technical standpoint, the hub's reliance on pre-trained language models introduces potential for hallucination and factual errors. The schema validation layer mitigates many of these issues for domain-specific facts, but open-ended claims or references to real-world events require additional verification mechanisms.

Finally, the computational footprint of AI models, even when optimized, has environmental and accessibility implications. The hub supports edge-friendly configurations and encourages developers to consider energy-efficient deployment strategies.

8. Future Work

Several avenues exist to extend and strengthen the hub. First, integrating stronger planning components can improve long-term narrative coherence by enabling the system to reason about multi-step goals and consequences. Second, richer player modeling that captures preferences, playstyle, and emotional state can enable more personalized narratives. Third, expanding schema expressiveness to include probabilistic constraints and soft preferences will allow more nuanced trade-offs between creativity and rule adherence.

Integration with AR and VR platforms is another promising direction, where adaptive narratives can respond to physical context and embodied interactions. Finally, collaborative storytelling features that allow multiple players to co-author narratives in shared worlds present both technical and design challenges worth exploring.



9. Conclusion

The AI-Integrated Game Creation Hub demonstrates that semi-runtime, player-dependent narrative generation is both feasible and valuable when combined with schema-driven validation and modular procedural systems. The hub improves replayability, reduces developer workload, and produces narratives that are coherent and engaging. While challenges remain in schema design, dataset bias, and computational efficiency, the hybrid approach presented here offers a practical path toward adaptive storytelling in games and interactive simulations. Continued research into planning, personalization, and efficient model deployment will further expand the capabilities and applicability of such systems.

References

1. JSON Schema tutorial (validation best practices) — Complete guide to JSON Schema validation. https://betterjsonviewer.com/json-schema-tutorial
2. Godot procedural generation tutorial (video walkthrough) — Practical Godot procedural generation guide. https://www.youtube.com/watch?v=oB1xsCcO9wI
3. Godot docs: Procedural geometry — Official Godot procedural geometry reference and examples. https://docs.godotengine.org/en/stable/tutorials/3d/procedural_geometry/index.html
4. IFDB (Interactive Fiction Database) — corpus and resources for interactive fiction datasets. https://ifdb.org/
5. Plans and Planning in Narrative Generation (review) — classic plan-based narrative generation survey. https://www.justusrobertson.com/papers/Young%20Ware%20Cassell%20and%20Robertson%202013%20-%20Plans%20and%20Planning%20in%20Narrative%20Generation.pdf
6. Player-Centered AI for Automatic Game Personalization (open problems) — player modeling & personalization overview. https://arxiv.org/pdf/2102.07548
7. Masked Generative Story Transformer (recent work on story transformers) — transformer approaches for story generation. https://arxiv.org/abs/2403.08502
8. PCG in Games: Survey with LLM integration insights (arXiv) — up-to-date PCG survey including LLMs. https://arxiv.org/abs/2410.15644
9. Procedural content generation for games: a survey (VU Amsterdam) — foundational PCG survey. https://research.vu.nl/en/publications/procedural-content-generation-for-games-a-survey
10. Improving Speech Recognition with Vosk / Kaldi (paper) — practical Vosk/Kaldi guidance for on-device ASR. https://arxiv.org/pdf/2503.21025
11. MediaPipe Hands: On-device Real-time Hand Tracking (paper) — authoritative hand-tracking pipeline. https://arxiv.org/abs/2006.10214
12. QuestVille: Procedural Quest Generation Using NLP Models (FDG/ACM) — quest generation using NLP techniques. https://dl.acm.org/doi/fullHtml/10.1145/3582437.3587188