Why Houdini is the Only DCC Tool Structurally Ready for LLm Integration:
Naturalhoudini — A Chatbot Interface for the Future of 3d Production Pipelines

Prajyot Ugale
BSc Animation & Visual Effects (Final Year), Ajeenkya DY Patil University, Pune, India

***Abstract*—An AI chatbot interface for Houdini — one where a 3D artist describes what they want in plain language and the system builds the node network for them — is not a speculative future. It is the inevitable next step in 3D production pipeline evolution, and Houdini is the only major DCC tool whose architecture makes it structurally ready for this integration today. This paper argues that case. NaturalHoudini is the proposed system: a chatbot-style natural language interface that compiles artist intent into live, editable Houdini node networks via LLM-generated VEX, Python, and HScript code.**

The reason Houdini is uniquely positioned is architectural: Houdini's procedural, node-based directed acyclic graph (DAG) is structurally isomorphic to the computational graphs LLMs naturally generate. No other major DCC tool — not Blender, not Maya — shares this property. As LLMs improve at code generation generally, that improvement transfers automatically to VEX (Houdini's C-like graphics language), because VEX shares the same C-derived syntactic structure as languages already extensively represented in LLM training data. This paper does not benchmark which LLM is best for Houdini — that question becomes outdated within months. The question this paper answers is: why is Houdini the correct platform for LLM integration, and how does the NaturalHoudini architecture realise that potential?

The VEX feasibility claim is grounded in published research: LLMs have demonstrated successful real-time generation of GLSL [1] and HLSL [2] shaders from natural language — both C-like languages sharing VEX's syntactic structure. The paper draws on LLM code generation benchmarks [3][4], domain-specific language accuracy research [5], AI-driven 3D scene generation [6][7], Houdini's expanding native ML infrastructure [8][9], and a global AI-in-VFX market growing at 19.46% CAGR [10]. It concludes with a candid accessibility assessment and a future work roadmap for empirical validation.

***Index Terms*—Houdini, VEX, LLM, procedural generation, AI pipeline, NaturalHoudini, ComfyUI, Copernicus, domain-specific language, VFX, chatbot interface**

I. INTRODUCTION

The future of 3D production pipelines is a working artist describing what they want in plain language — and an AI building the Houdini node network for them. That is not a distant aspiration. It is the logical next step given where large language model (LLM) capabilities are today, and Houdini is the only major DCC tool whose architecture makes it structurally ready for this integration. This paper argues that case and proposes NaturalHoudini — a chatbot-style natural language interface that compiles artist intent into live, editable Houdini node networks via LLM-generated VEX, Python, and HScript code.

3D content creation — particularly at the level of simulation and procedural effects — has historically required years of technical specialisation before an artist can translate creative vision into production-quality output. SideFX Houdini, while unmatched in expressive power among 3D DCC tools, is the clearest example of this dynamic: it is the industry standard for VFX simulation at major studios worldwide, yet its node-based procedural architecture demands a depth of technical knowledge that limits the pool of practitioners who can use it effectively.

The emergence of LLMs capable of generating structured, executable code from plain language

descriptions offers a concrete path to bridging this gap. Existing research has demonstrated that LLM agents can drive 3D tools such as Blender through scripted interfaces [6][7]. However, these approaches produce object-level, baked outputs — they create assets, not parametric graphs. For production VFX workflows, this is insufficient: a simulation that cannot be non-destructively edited at the node level is not a professional deliverable.

This paper makes three core claims: (1) Houdini's DAG architecture is the only LLM-native compilation target among major DCC tools — no other platform shares this structural property; (2) VEX — Houdini's C-like graphics computation language — is a feasible LLM code generation target, as evidenced by published results for analogous C-like shader languages [1][2]; and (3) Houdini's native ML infrastructure already provides the technical foundation for a complete AI pipeline, requiring only the NaturalHoudini chatbot interface layer to become production-ready.

Critically, this paper does not benchmark which LLM is best for this task — that question becomes outdated within months. The argument is structural: as LLMs improve at code generation generally, VEX generation improves automatically, because VEX shares the same C-like syntactic structure that LLMs are already trained on. The improvement is directly proportional and does not require VEX-specific training. The right question is therefore not 'which LLM is best for VEX?' but 'why is Houdini the uniquely correct platform for an LLM-powered creative assistant?' — and that is the question this paper answers.

Practitioner Perspective: This paper is authored from a practitioner background — direct Houdini production experience combined with cited computer science literature. That cross-disciplinary vantage point is the paper's primary value: the architectural insight that Houdini's DAG makes it uniquely LLM-ready is not visible from a pure CS perspective. It requires knowing what working in Houdini actually feels like, and recognising the structural parallel to how LLMs generate code.

II. RELATED WORK

A. LLM-Generated Shader Code: The VEX Precedent

The most directly relevant precedents for VEX generation from natural language are published systems that generate C-like graphics shading languages — languages that share VEX's syntactic structure and domain — from user instructions.

AI Co-Artist (arXiv:2512.08951, 2024) [1] presents a system using GPT-4 to generate and iteratively refine GLSL shader code from natural language descriptions, producing complex real-time visual effects. GLSL shares C's core syntax — variable declarations, control flow, vector/matrix types, and function definitions — with VEX. Critically, the paper explicitly frames the problem in terms directly parallel to this proposal: the system demonstrates that LLMs can bridge exactly the kind of language-fluency barrier that Houdini artists face daily.

ShadAR (CHI 2026, ACM:10.1145/3772363.3799378) [2] demonstrates LLM-driven generation of HLSL shaders from spoken natural language commands in real time, deployed on AR hardware. HLSL is a Microsoft graphics shading language with C-like syntax, directly comparable in structure to VEX. Together, these two papers constitute the strongest existing evidence for the VEX feasibility claim: if LLMs can generate GLSL and HLSL from natural language, they can generate VEX — because all three languages share the same C-derived syntactic foundation, strong typing, and vector-oriented mathematical operations.

B. LLM-to-3D Code Generation

SceneCraft (Hu et al., ICML 2024) [6] demonstrated that LLM agents can generate Blender-executable Python scripts producing complex 3D scenes of up to 100 assets from text descriptions, using scene graph blueprints and vision-language model feedback for iterative refinement. 3Dify (Hayashi et al., 2025) [7] proposed a multi-DCC framework using Anthropic's Model Context Protocol (MCP) and Retrieval-Augmented Generation (RAG) to drive DCC tools — explicitly including Houdini — via LLM-generated code. BlenderGym (Zhang et al., 2024) [12] benchmarked LLM procedural modelling skills, finding that frontier models handle geometry well but struggle with complex simulation parameters — precisely the area where Houdini's deeper node stack and VEX access provide more precise generation targets than Blender's Python API.

C. LLM Code Generation Benchmarks

Chen et al. (2021) [3] introduced HumanEval, the primary LLM code generation benchmark. Frontier models (GPT-4o, Claude 3.5) achieve approximately 85% pass@1 on standard Python tasks [4] — establishing a strong baseline capability for code generation from natural language. Al Mazrouei (2024) [5] demonstrated that purpose-designed domain-specific languages with constrained syntax can achieve higher LLM generation accuracy than general-purpose languages even with zero training exposure: 99.9% parse success and a 40 percentage point accuracy advantage over Python on multi-step tasks. These results establish that LLM code generation quality is not solely a function of training data volume, but of language constraint and clarity — an advantage that benefits VEX. Importantly, these benchmarks track an upward trajectory across all LLM families: as code generation improves generally, VEX generation will improve proportionally.

D. Houdini's ML Infrastructure

SideFX's Copernicus context (Houdini 20.5, 2024) introduced a GPU-accelerated image processing pipeline with an ONNX inference node for native ML model execution [8][9]. Houdini 21 (2025) extended this to full supervised ML pipelines — data generation, preprocessing, training, and inference — described by SideFX as moving from R&D to deployable tools [11]. Neural Point Surface, an ML-based liquid surface reconstruction tool, shipped as a production-ready example of a learned node in H21 [13]. SideFX officially documents Houdini's capacity to generate large volumes of high-quality, labelled data for training models, including 3D geometry, simulations, and rendered imagery [14].

E. Gap in Existing Literature

No existing work has proposed or evaluated a natural language chatbot interface specifically targeting Houdini's procedural node graph — one that generates VEX code, HScript expressions, and Python hou API calls as compilation outputs for a live Houdini session. This gap, and the practitioner-level insight into why Houdini's architecture makes it the right target for this interface, is the primary contribution of this paper.

III. VEX AS AN LLM CODE GENERATION TARGET: A CITED FEASIBILITY ARGUMENT

This section constitutes the paper's primary original contribution: a structured argument that VEX is a suitable LLM code generation target, grounded entirely in published research without requiring original experiments.

A. VEX is a C-Like Graphics Domain-Specific Language

VEX (Vector Expression) is described in SideFX's official documentation as a high-performance expression language loosely based on the C language, taking ideas from C++ and the RenderMan shading language [15]. It supports standard C control flow (for, while, if/else, break, continue), strong static typing (int, float, vector, string, matrix, struct), and a context-specific execution model. VEX code uses C-identical syntax for all control structures and expressions; the differences from C are limited to the @ attribute access convention, the context system, and VEX-specific built-in functions [16].

This places VEX in the same syntactic family as GLSL and HLSL — both C-like graphics shading languages with strong typing, vector/matrix types, and context-specific built-in functions. The structural similarity is not superficial: VEX, GLSL, and HLSL all compile to GPU-executable code, operate on typed geometric or image data, and share C's operator precedence, scoping rules, and function definition syntax.

B. Precedent: LLMs Successfully Generate GLSL and HLSL

The fundamental insight of this paper is that LLMs do not need to be specifically trained on VEX to generate it well — they need only to be good at C-like code in general. Because VEX shares the syntactic structure of C, C++, GLSL, and HLSL — languages that are extensively represented in LLM training data — any improvement in LLM coding ability automatically transfers to VEX. This is the relationship this paper seeks to establish: not a static benchmark of today's models, but a structural argument that the LLM-to-Houdini pathway improves as the field improves.

AI Co-Artist [1] demonstrates that GPT-4 generates functional GLSL shader code from natural language. ShadAR [2] demonstrates real-time HLSL generation from spoken commands on deployed AR hardware.

Both systems work precisely because GLSL and HLSL share C's syntactic structure — the same structure VEX shares. As frontier LLMs continue to advance in coding reasoning, longer context, and instruction-following, VEX generation will advance with them. The question is not whether — it is when, and how to build the integration layer that makes this accessible to artists.

C. The DSL Constraint Advantage

Al Mazrouei (2024) [5] demonstrated that constrained DSL syntax significantly reduces LLM code generation errors compared to general-purpose languages. VEX's constraints — smaller vocabulary than C++, no dynamic dispatch, no undefined behaviour — are precisely the kind of constraints that produce this accuracy advantage. A grammar-constrained generation layer applied to LLM token output for VEX would reduce syntax errors further, as demonstrated by grammar-augmented code generation research [17].

D. The Low-Resource Risk and RAG Mitigation

VEX has significantly less public training data than Python, C, or GLSL. Research documents that the performance of LLMs worsens when dealing with low-resource programming languages [18]. This is the primary risk for VEX generation quality. The mitigation is a Houdini-specific RAG corpus — following the approach of 3Dify [7] — constructed from SideFX VEX documentation, community wrangle repositories, and annotated .hip files. RAG supplies domain-specific vocabulary at generation time, compensating for low training data volume without requiring model fine-tuning.

E. Summary Comparison

Table 1. Comparison of VEX with GLSL and HLSL across factors relevant to LLM code generation.

Factor	GLSL	HLSL	VEX	Implication for VEX
C-like syntax	Yes	Yes	Yes	Full C syntax transfer applies
Strong typing	Yes	Yes	Yes	Reduces LLM ambiguity
Vector/matrix types	Yes	Yes	Yes	Shared mathematical vocabulary
Graphics/compute domain	Yes	Yes	Yes	Similar operation semantics
LLM generation shown	Yes [1]	Yes [2]	By analogy	Analogical feasibility established
Low training data risk	Medium	Medium	High	Mitigated by RAG corpus

IV. FUTURE WORK: VALIDATION EXPERIMENT

This paper establishes the architectural and feasibility argument for NaturalHoudini. The natural next step is empirical validation. The following evaluation framework is proposed for future researchers, developers, or the author as a follow-up study.

A. Experiment Rationale

The goal of the validation experiment is not to compare one LLM against another — such comparisons become outdated within months as models advance. The goal is to establish a baseline for VEX generation quality using any capable frontier LLM, and to confirm in practice that the structural analogy to GLSL/HLSL holds. The experiment tests the core claim: that a general-purpose LLM, with no VEX-specific training, can produce syntactically and functionally correct VEX code from natural language descriptions when provided with relevant documentation via RAG.

B. Proposed Methodology

The experiment would proceed in three phases:

Phase 1 — Prompt Set Construction: A set of 30–50 representative natural language prompts spanning three difficulty tiers would be created. Tier 1 (basic): simple per-point attribute operations (e.g., 'set the colour of each point based on its height'). Tier 2 (intermediate): wrangle logic involving conditionals and loops (e.g., 'delete every other point in the selection'). Tier 3 (advanced): simulation-level parameter control (e.g., 'create a noise-driven velocity field for a pyro solver').

Phase 2 — Generation and Evaluation: Each prompt would be sent to a frontier LLM (e.g., GPT-4o, Claude 3.5, or equivalent) with and without a RAG corpus constructed from SideFX VEX documentation. Generated VEX code would be evaluated on three metrics: (a) syntactic validity — does it parse without errors in Houdini's VEX context?; (b) functional correctness — does it produce the described output when executed?; (c) editability — can a working Houdini artist understand and modify the generated node?

Phase 3 — RAG Delta Analysis: The difference in metrics between RAG-augmented and non-RAG-augmented generation would quantify the benefit of the documentation corpus, validating the mitigation strategy proposed in Section III.D.

C. Expected Outcome

Based on the GLSL and HLSL precedents [1][2], the hypothesis is that a frontier LLM will achieve acceptable syntactic validity (>70% parse success) on Tier 1 and Tier 2 prompts without RAG, with RAG augmentation improving all tiers by a meaningful margin. Confirming that the VEX-generation pathway works without fine-tuning would validate that the NaturalHoudini architecture is buildable today with publicly available LLM APIs. A preliminary version of this evaluation is intended using Houdini Apprentice and a public LLM API.

Table 3. Summary of the feasibility argument and its future validation roadmap.

Claim / Dimension	Evidence / Basis	Relevance to NaturalHoudini
Natural language → node compilation	LLM code generation research [3][4]	Foundation for NaturalHoudini compiler
VEX generation feasibility	GLSL [1] and HLSL [2] analogy	VEX shares C-like syntax with both
DAG structural fit	Houdini node graph architecture	Isomorphic to LLM computational graphs
Low-resource language risk	RAG from SideFX documentation	Compensates for limited VEX training data
Artist accessibility	NaturalHoudini chatbot interface	Core motivation of this proposal

V. NATURALHOUDINI: PROPOSED SYSTEM ARCHITECTURE

Based on the feasibility argument in Section III, we propose NaturalHoudini — a three-layer architecture that translates natural language artist intent into live, editable Houdini node networks. The interface is designed as a chatbot: the artist types or speaks a description of what they want, and NaturalHoudini builds the corresponding node network in their active Houdini session. Each layer is defined with its inputs, outputs, existing technology dependencies, and the engineering work remaining.

A. Layer 1 — Natural Language Interface (The Chatbot Layer)

The NLI receives free-form natural language from the artist — through a panel inside Houdini or an external chat interface — and produces a structured intermediate representation (JSON) specifying node types, connection topology, parameter values, and VEX snippets required. It operates in two modes: initial construction mode (full network from a description) and delta mode (targeted update from a follow-up instruction such as 'make the smoke thicker'). Delta mode preserves artist edits to the existing graph and enables the conversational iteration loop that distinguishes NaturalHoudini from full-regeneration approaches.

The NLI is backed by an LLM augmented with a Houdini-specific RAG corpus — constructed from SideFX VEX documentation, the Python hou module API reference, and annotated community node networks. This follows the RAG approach of 3Dify [7] extended specifically to Houdini's language stack.

B. Layer 2 — Graph Compiler

The compiler translates the structured node specification into three code targets selected by operation type:

Houdini Python API (hou module) — node creation, connection, and network-level operations.

VEX — per-point attribute computation in Attribute Wrangle nodes; the primary target for simulation and geometry operations.

HScript — parameter linking and conditional expressions within node parameter fields.

Targeting all three layers — rather than Python alone — is the architectural distinction from SceneCraft and

3Dify. VEX gives NaturalHoudini access to per-point operations with no equivalent in object-level scripting, enabling simulation-level customisation essential for production VFX.

C. Layer 3 — Live Session Integration

The session layer executes compiled scripts against an active Houdini instance via the hou Python module, maintaining state tracking (AI-generated vs. artist-created nodes), version history, and error interception — routing Python/VEX errors back to the LLM for self-correction, following SceneCraft's iterative refinement pattern [6].

D. Copernicus and ComfyUI Extension

Beyond node-network generation, NaturalHoudini extends to generative AI image capabilities via two components. Houdini's Copernicus ONNX inference node [9] enables ML model execution (denoising, upscaling, segmentation) as native Houdini nodes — the artist requests 'denoise this render' and the compiler emits an ONNX COP node. A proposed ComfyUI bridge — a Python operator communicating with the ComfyUI REST API — would extend this to the full ecosystem of Stable Diffusion community models as Houdini graph operations, without modifying either application.

Table 2. NaturalHoudini system layers, data flow, and implementation status.

Layer	Input	Output	Key Technology	Status
NLI + RAG	Natural language	Node spec (JSON)	LLM + Houdini doc corpus	Requires development
Compiler	Node spec (JSON)	Python / VEX / HScript	LLM code generation	Requires development
Session Integration	Code scripts	Live Houdini nodes	hou Python API (ships with Houdini)	API exists; connector needed
Copernicus Bridge	ONNX model + render	ML inference as COP node	ONNX runtime (built into H20.5+)	Infrastructure exists
ComfyUI Bridge	Workflow JSON	Diffusion output as layer	ComfyUI REST API	Connector required

VI. INDUSTRY CONTEXT

A. Market Scale and AI Investment

The global VFX market was valued at $10.54 billion in 2024 and is projected to reach $24.34 billion by 2033 at a CAGR of 9.75% [19]. The AI-in-VFX sub-segment is projected to grow from $4.87 billion in 2025 to $28.66 billion by 2035 at a CAGR of 19.46% [10]. AI-assisted tooling has already reduced rotoscoping turnaround time by 30–50% at major studios in 2024–2025 (Omdia, 2025) [20]. Framestore launched a dedicated AI lab in London in 2024; DNEG merged with Metaphysic in 2024 to accelerate AI-driven content workflows [21].

B. Houdini's Institutional Position

Houdini is the dominant simulation and effects platform at tier-1 VFX studios globally — ILM, Weta FX, Framestore, DNEG, MPC, and Scanline VFX all operate Houdini-centric effects pipelines. NaturalHoudini would integrate into existing studio infrastructure rather than requiring platform migration — the critical adoption factor for pipeline technologies at this scale. SideFX's Houdini 21 ML release confirms institutional alignment: AI integration is a production roadmap priority, not a speculative research direction.

C. The Learning Curve as Design Motivation

Houdini's steep learning curve is its primary adoption barrier, and it is precisely the barrier that NaturalHoudini is designed to reduce. This motivation is not unique to Houdini: AI Co-Artist [1] was built explicitly to address the same problem in GLSL shader programming — the steep learning curve and requirement for programming fluency pose substantial barriers for newcomers and even experienced artists.

NaturalHoudini extends this principle to a more powerful and more technically demanding tool, where the gap between novice capability and expert capability is largest, and therefore the potential productivity gain from a chatbot interface is greatest.

VII. LIMITATIONS

This paper presents no working implementation and no original experimental results. All feasibility claims are grounded in structural analogy to published results (GLSL/HLSL generation) and cited LLM code generation research. The VEX generation argument is reasoned, not empirically demonstrated — which is why Section IV proposes a concrete validation experiment.

The analogical argument from GLSL/HLSL to VEX, while structurally sound, does not account for VEX's lower internet training data volume. RAG mitigation is proposed but not evaluated. Actual VEX generation accuracy may differ from the GLSL/HLSL precedent due to this data gap.

The paper is authored from an Animation & VFX practitioner background rather than a computer science research background. The architectural proposal draws on direct Houdini production experience; the ML systems implementation complexity is assessed from published descriptions rather than engineering experience.

Houdini's commercial licensing restricts access for independent research and prototyping. The free Apprentice tier limits commercial use and render resolution, constraining the scope of community-driven development of NaturalHoudini.

ComfyUI integration introduces a dependency on an external server with variable latency. Real-time interactive use requires further latency characterisation.

VIII. CONCLUSION

This paper has argued that SideFX Houdini is the optimal platform for AI integration in 3D production pipelines, grounded in three complementary claims: its DAG architecture is structurally isomorphic to LLM-generated computational graphs; VEX, its core language, is a feasible LLM code generation target by analogy to demonstrated GLSL and HLSL generation [1][2]; and its native ML infrastructure already provides the technical foundation for a complete AI pipeline.

The paper's central architectural argument is this: as LLMs improve at code generation generally, that improvement automatically benefits VEX generation, because VEX shares the same C-like syntactic structure as languages extensively represented in LLM training data. Benchmarking specific models against each other is therefore beside the point — the trajectory of the field is what matters, and that trajectory points clearly toward an LLM that can reliably generate Houdini node networks from natural language.

We have proposed the NaturalHoudini architecture that realises these properties as a chatbot-style natural language interface to Houdini's node graph and situated this proposal within a $28.66 billion projected AI-in-VFX market [10]. We have further proposed a concrete validation experiment that the research community or future developers can execute to empirically confirm the feasibility argument.

The industrial motivation is real and growing. Studios are investing in AI pipeline integration at scale, Houdini occupies the dominant position in exactly the workflows where AI assistance would have the greatest impact, and the structural case for Houdini as the optimal AI integration target is strong. The artist who can direct an AI to construct a pyro solver setup in plain language and then edit it at the node level is not replaced by AI — they are amplified by it.

A. On Accessibility: The Work That Remains

It is important to be honest about a significant practical barrier that this paper's architectural proposal does not yet resolve: integrating LLMs with Houdini today is not accessible to the average 3D artist. Connecting an LLM to a live Houdini session, building a RAG corpus from SideFX documentation, writing a VEX compiler, and maintaining a stable Python API bridge requires substantial software engineering expertise that most animation and VFX artists — including the author of this paper — do not currently possess. The gap between the architecture proposed here and a tool that a working Houdini artist could install and use in a single afternoon remains wide.

This limitation is not permanent — it is a function of where we are in the maturity curve of LLM tool integration. As AI-assisted coding tools improve, as SideFX potentially develops first-party LLM

integration features within Houdini, and as the open-source community builds accessible bridges between LLM APIs and the hou Python module, the engineering barrier will lower. The precedent is visible: GLSL shader generation went from a research paper [1] to a deployed real-time AR system [2] within a matter of months. The same trajectory for VEX and Houdini is not only plausible — given SideFX's documented ML investment in Houdini 21 [11] — it is likely.

The hope expressed by this paper is therefore not just that NaturalHoudini can be built — it is that in the near future, it will not require a software engineer to build it. A working Houdini artist who understands what they want to create should be able to describe it in plain language, have the AI build the node network for them, and iterate on it conversationally — without needing to understand what happens under the hood. That future is the real goal of this proposal, and it is closer than it has ever been.

REFERENCES

[1] AI Co-Artist. (2024). AI Co-Artist: A LLM-Powered Framework for Interactive GLSL Shader Animation Evolution. arXiv:2512.08951. [GPT-4 generates GLSL shaders from natural language; addresses steep learning curve for shader programming.]

[2] ShadAR. (2026). ShadAR: LLM-Driven Shader Generation to Transform Visual Perception in Augmented Reality. CHI 2026, ACM. doi:10.1145/3772363.3799378. [Real-time HLSL shader generation from spoken natural language on AR hardware.]

[3] Chen, M., Tworek, J., Jun, H., et al. (2021). Evaluating Large Language Models Trained on Code. arXiv:2107.03374. [HumanEval benchmark — foundational LLM code generation evaluation.]

[4] EvoEval / COLM 2024. (2024). Can LLMs Replace Human Evaluators? COLM 2024. openreview.net/pdf?id=zZa7Ke7WAJ. [Frontier models GPT-4o, Claude-3.5 achieve ~85% pass@1 on HumanEval.]

[5] Al Mazrouei, S.K. (2024). Anka: A Domain-Specific Language for Reliable LLM Code Generation. arXiv:2512.23214. [DSL with constrained syntax: 99.9% parse success, +40pp over Python on multi-step tasks with zero prior training.]

[6] Hu, Z., Iscen, A., Jain, A., Kipf, T., Yue, Y., Ross, D.A., Schmid, C., & Fathi, A. (2024). SceneCraft: An LLM Agent for Synthesizing 3D Scene as Blender Code. ICML 2024. arXiv:2403.01248.

[7] Hayashi, S., Mukunoki, D., Hoshino, T., Ohshima, S., & Katagiri, T. (2025). 3Dify: A Framework for Procedural 3D-CG Generation Assisted by LLMs Using MCP and RAG. arXiv:2510.04536.

[8] SideFX. (2024). Copernicus Context — What's New in Houdini 20.5. sidefx.com/docs/houdini/news/20_5/copernicus.html

[9] SideFX. (2024). ONNX Inference Copernicus Node. sidefx.com/docs/houdini/nodes/cop/onnx.html

[10] SNS Insider. (2025). AI in VFX Market: USD 4.87B in 2025, projected USD 28.66B by 2035, CAGR 19.46%. snsinsider.com/reports/ai-in-vfx-market-9394

[11] SideFX. (2025). Building Machine Learning Pipelines — Houdini 21 SIGGRAPH Talk. sidefx.com/houdini-hive/siggraph-2025/

[12] Zhang, Z., et al. (2024). BlenderGym: Evaluating the Procedural 3D Modelling Skills of Large Language Models. arXiv:2504.01786.

[13] SideFX. (2025). Houdini 21: VFX — Neural Point Surface. sidefx.com/products/whats-new-in-h21/vfx/

[14] SideFX. (2024). Machine Learning & AI — Houdini Pipeline & AI. sidefx.com/products/houdini/pipeline-ai/machine-learning-ai/

[15] SideFX. (2024). VEX Overview. sidefx.com/docs/houdini/vex/index.html

[16] Tomori, J. (2021). VEX Tutorial — Syntax and Capabilities. github.com/jtomori/vex_tutorial [C-like control flow, strong typing, attribute access syntax in VEX.]

[17] Grammar-Coder Research Group. (2025). Grammar-Based Code Representation: Is It a Worthy Pursuit for LLMs? arXiv:2503.05507.

[18] Casamayor Pujol, V., et al. (2026). A Framework for Assessing LLM Capabilities in Constraint Domain-Specific Languages. arXiv:2603.05278.

[19] SkyQuestT. (2025). Visual Effect Market Forecast. skyquestt.com/report/visual-effects-market
[20] Omdia. (2025). Media Technology Report 2025. [As cited in: vitrina.ai/blog/global-film-monetization-strategies.]
[21] Global Growth Insights. (2025). VFX Market Report 2025–2033. globalgrowthinsights.com [Framestore AI Lab 2024; DNEG-Metaphysic 2024.]
[22] Sun, C., et al. (2023). 3D-GPT: Procedural 3D Modelling with Large Language Models. arXiv:2310.12945.
[23] Sudhakaran, S., et al. (2024). Procedural Content Generation via Generative AI: A Survey. arXiv:2407.09013.
[24] Digital Production. (2025). Houdini 21: SideFX Pushes for Production Readiness. digitalproduction.com/2025/08/13/houdini-21-sidefx.