Supervised Specialisation of Qwen3.5-4B for Three.js 3D Scene Generation

An exploratory study with synthetic data

ccore · Independent research · 28 October 2026

Abstract

This work presents the specialisation of Qwen3.5-4B for generating executable Three.js scenes from natural-language instructions. The corpus contains 4,146 synthetic examples, with 3,933 training and 213 validation examples. The model was developed with approximately 59 h 10 min of full fine-tuning of the language parameters and 45 min 20 s of LoRA, totalling about 59 h 55 min across the stages incorporated into the final weights. Each run used one GPU, without an additional reinforcement learning stage. An exploratory evaluation applied eight prompts to the base model and eight checkpoints. Checkpoint 8 met the acceptance criteria for all eight cases after 18 calls: six outputs retained their original code and two received local repairs. Visual comparisons show more recognisable scenes and better-organised compositions than the selected reference outputs. The protocol included retries and different stopping rules across versions. The results demonstrate the feasibility of supervised fine-tuning in this domain. The dataset and FP16 weights are publicly available for further experiments.

Keywords: supervised fine-tuning; language models; synthetic data; Three.js; 3D generation; LoRA

1. Introduction

Generating a 3D scene requires combining geometry, materials, lighting and camera placement in an executable program. Valid JavaScript alone does not guarantee a coherent composition. Objects may be disconnected, poorly scaled or outside the frame. Three.js provides an environment in which execution and appearance can be examined together [5].

The project began by exploring small models for this task. The initial intention was to establish a supervised fine-tuning (SFT) baseline and then investigate reinforcement learning (RL). Intermediate results motivated further dataset expansion and continued SFT. Development was exploratory, with configurations adjusted throughout the experiments.

This work presents the resulting model, its training trajectory and visual comparisons with the Qwen3.5-4B reference [1]. The study brings together executable code, example scenes and public artefacts. Specialisation used SFT without additional RL; this description applies to the present study, not to the original model’s post-training.

2. Related Work

LIMA investigated supervised alignment with a small set of selected demonstrations, without an additional RL stage [4]. That result motivates the study of data composition. The present work investigates this question in 3D scene generation, with a different task, model and training scale.

LoRA learns low-rank updates while keeping the original weights frozen [2]. QLoRA combines this adaptation with a quantised base [3]. A short stage of this kind was incorporated into the model. The LoRA experiments were part of development and are discussed in relation to the configurations used.

3. Dataset

The selected corpus was generated with MiniMax M3.1 Flash and expanded through three successive versions: 797, 2,128 and 4,146 examples. The final version contains 3,933 training and 213 validation examples. The original 80 validation examples remained in the expanded set; two examples from the initial training set were removed during the intermediate expansion [6].

Table 1. Characteristics of the final corpus.

Property	Description
Format	Instruction, reasoning and an HTML/JavaScript response
Coverage	41 annotated domains, including objects, interiors, architecture, landscapes, mechanisms and voxel scenes
Annotated complexity	Levels 1–10
Longest observed sequence	71,229 tokens, including the conversational template
Retained LoRA subset	178 complete examples of up to 16,384 tokens

Complexity levels are descriptive annotations of the corpus.

Training examples were kept complete, without truncation. Reasoning and response are separate fields, and validation examples were excluded from weight updates. The dataset used in this study is published on Hugging Face as ccore/ThreeJS-3D-Scenes [6], with fields for filtering examples by domain, complexity level and voxel style.

Initial exploration included DeepSeek data and 2B and 4B models. The trajectory producing the model in this study restarted from the official Qwen3.5-4B model with the MiniMax corpus.

4. Model and Training

4.1. Weight Lineage

The starting point was Qwen/Qwen3.5-4B [1]. Full fine-tuning here means updating the language parameters while keeping the visual encoder frozen. Checkpoints 1–4 resulted from this training. One LoRA epoch on 178 examples produced Checkpoint 5. The adapter was merged into the FP16 weights before full fine-tuning resumed.

The final run started from Checkpoint 5 with a fresh Adam state and produced Checkpoints 6–8. Other LoRA branches started from the same checkpoint but were discarded (Figure 1). Appendix A gives the configurations of the retained stages.

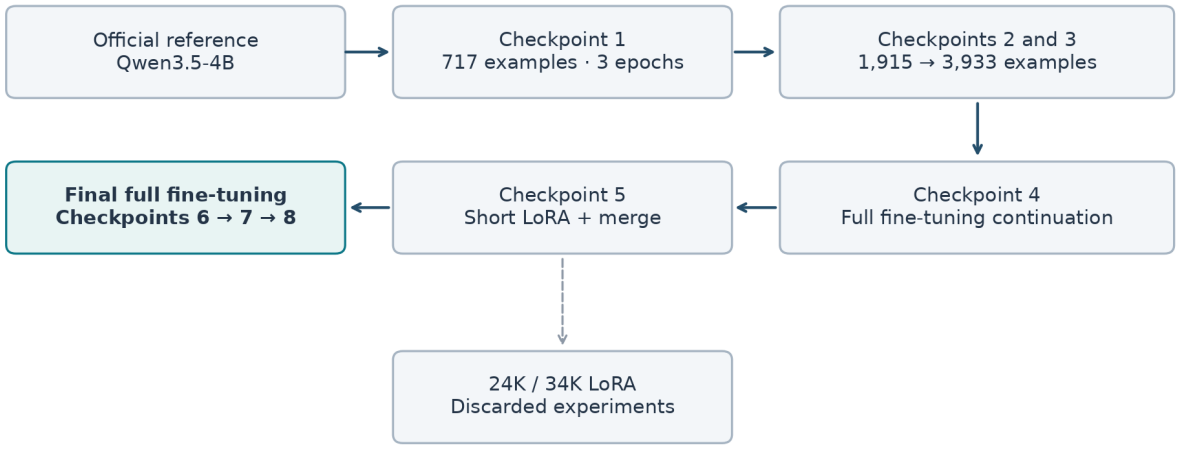


Figure 1. Lineage of the selected model. Solid arrows indicate stages incorporated into the final weights; the dashed branch contains discarded LoRA experiments.

4.2. Optimisation and Precision

In the final run, master weights, gradients and Adam states were kept in FP32. Computation used BF16 and FP8 matrix multiplications in selected linear layers. Optimiser states were offloaded to system memory. FP16 is the distribution format, not the precision of every training state.

The objective was next-token cross-entropy. Within each accumulation window, the summed loss was divided by the total number of valid supervised targets. With microbatch size 1 and accumulation factor 2, each window contained up to two examples. No further division by the accumulation factor was applied.

Table 2. Final-run and retained LoRA configurations.

Parameter	Final full fine-tuning	Short LoRA stage
Microbatch / accumulation	1 / 2	1 / 2
Learning rate	Peak: 10^{-5}; floor: 2×10^{-6}	Constant 5×10^{-5}
Schedule	59 warmup updates; cosine decay	Constant
Weight decay	0.01	0
Adaptation	Language parameters; vision frozen	NF4; double quantisation; r = 16; α = 32; dropout = 0
Computation	BF16 and selected FP8 GEMMs	BF16
Longest observed sequence	71,229 tokens	16,384 tokens
Planned examples	3,933	178

The final run used AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$. The adapter and its optimiser were initialised for the short LoRA stage.

4.3. Selected Checkpoint and Export

Checkpoint 8 corresponds to 1,956 updates in the final run. The run processed 3,912 of the planned 3,933 examples, totalling 112,903,599 input tokens and 112,431,961 supervised targets. Checkpoints 6 and 7 are intermediate points in this same run, not additional training runs.

Weights were published in FP16 and converted to GGUF Q4_K_M and Q8_0 using llama.cpp tools [7,8]. Conversion included a layer/MTP metadata adjustment without further training.

4.4. Computational Resources

Full fine-tuning ran on Google Colab with one NVIDIA RTX PRO 6000 Blackwell Server Edition per run, providing approximately 96 GiB of VRAM. The 16K LoRA stage used an NVIDIA L4. In the final run, about 190 GB of system RAM supported optimiser-state offloading.

Table 3. Training time in the Checkpoint 8 lineage.

Training	Accumulated time
Accumulated full fine-tuning	≈59 h 10 min
16K LoRA — one epoch, 178 examples	45 min 20 s
Total incorporated into the model	**≈59 h 55 min**

The total covers the stages incorporated into the final weights. The final full fine-tuning run, already included in this total, took 16 h 01 min 37 s. Data generation, evaluation and discarded experiments are outside this count.

5. Evaluation

The exploratory comparison applied eight prompts to the official model and eight checkpoints, yielding 72 model–scene pairs. Requests covered voxel ancient Rome, a Kyoto temple, a Venice canal, a Victorian library, an alpine castle, a container harbour, a regional airport and a pool table. These prompts were used during development rather than as an independent blind test.

Each case allowed up to five attempts. The first accepted output was selected, not the one with the highest visual score. The protocol allowed a separate copy with up to three local syntax or identifier repairs. Geometry, composition and camera changes were not permitted. Inference parameters and detailed rules appear in Appendix B.

The official model stopped at the first technical render. Checkpoint acceptance required a recognisable scene and the requested controls. The comparison is qualitative and exploratory, with attempts and repairs reported separately in the results; it is not a uniform first-attempt success benchmark.

Visual assessment considered instruction adherence, spatial relations, geometry, appearance and composition. Agent/GPT-assisted analysis supported manual inspection rather than serving as a quantitative ranking. Checkpoint 8 was selected on the basis of this comparison and additional scene-generation tests.

6. Results

6.1. Execution and Acceptance

Checkpoint 8 met the acceptance criteria for all eight cases after 18 calls. Six selected outputs retained their original code and two received minimal repairs. Two scenes were accepted on the first attempt without editing; four were accepted on the first attempt when repairs are included. Eight calls were interrupted by reasoning loops before delivering HTML and remain in the count.

Table 4. Selected Checkpoint 8 outputs.

Scene	Attempt	Selected code	Calls
Voxel ancient Rome	A1	Minimal repair	1
Kyoto temple	A1	Minimal repair	1
Venice canal	A3	Original	3
Victorian library	A1	Original	1
Alpine castle	A1	Original	1
Container harbour	A5	Original	5
Regional airport	A3	Original	3
Pool table	A3	Original	3
Total	**8 cases**	**6 original; 2 repaired**	**18**

A1–A5 indicate the attempt order for each scene. Acceptance required a recognisable scene and functioning controls as requested.

6.2. Visual Comparisons

Checkpoint 8 produced more recognisable structures and better-organised compositions in the evaluated scenes. Figures 2–4 show six comparisons with the base model under the protocol in Section 5.

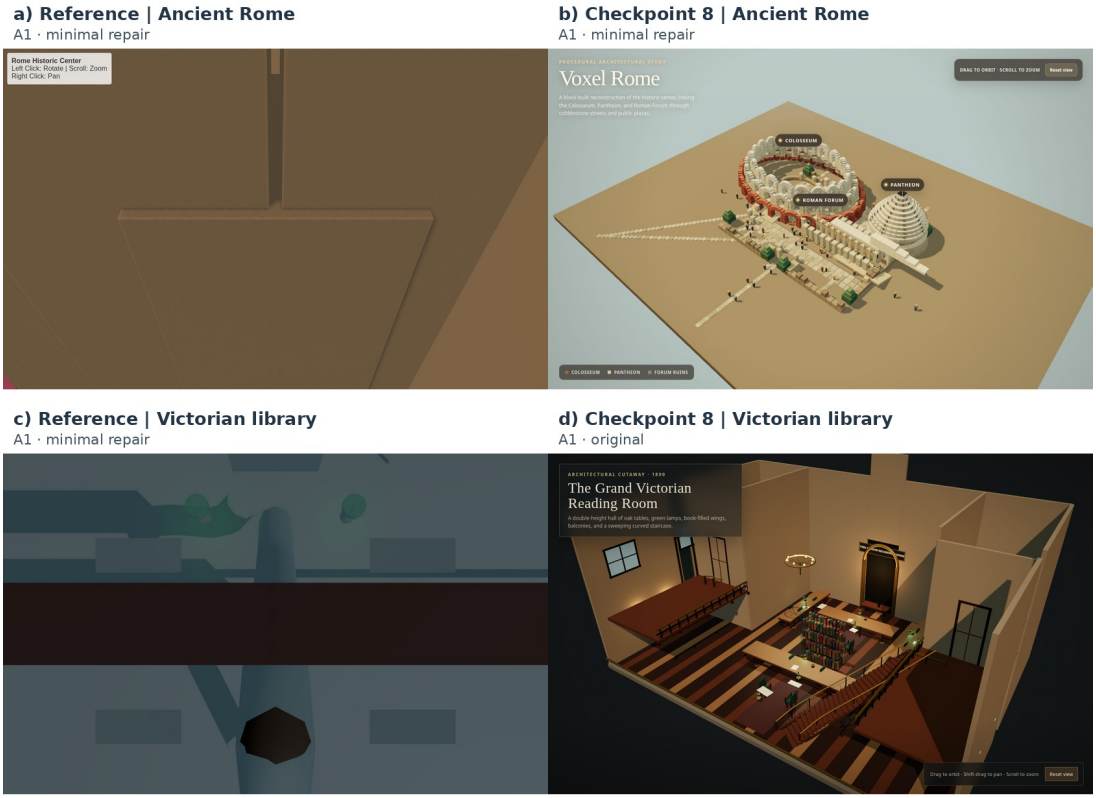


Figure 2. Ancient Rome and Victorian library. Reference on the left; Checkpoint 8 on the right. Rome uses repaired first attempts in both versions. For the library, the reference was repaired and the checkpoint retained its original code.

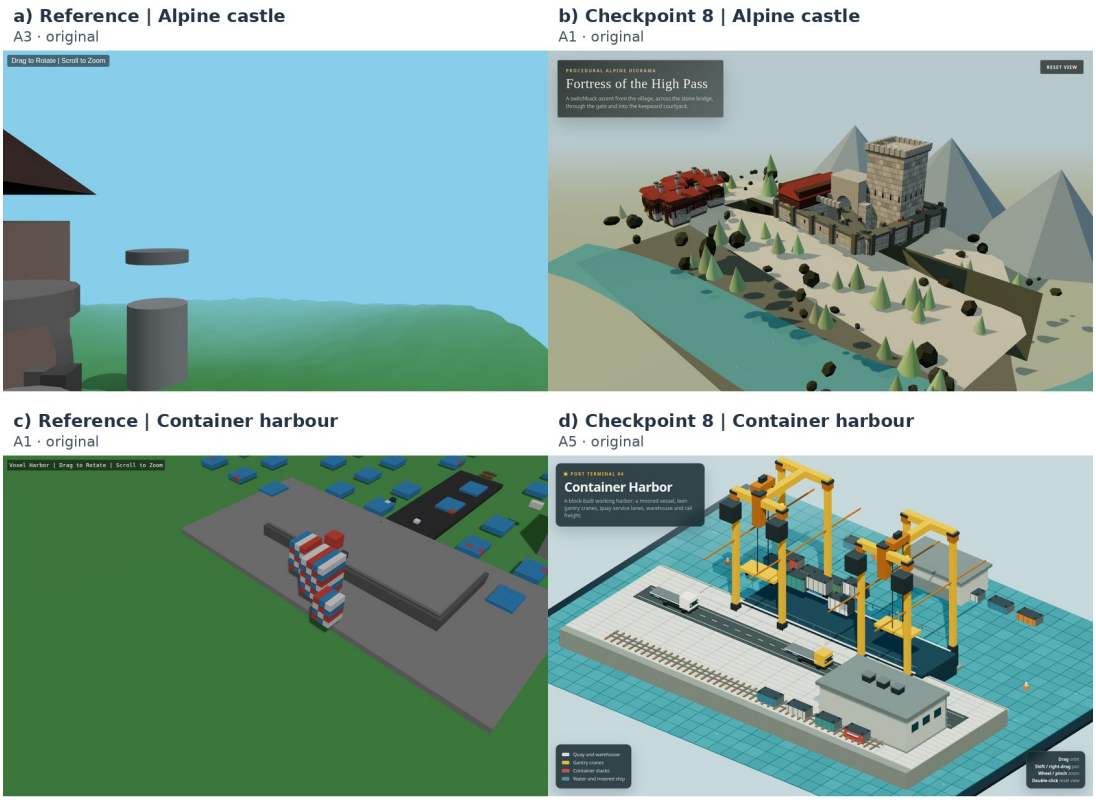


Figure 3. Alpine castle and container harbour. All selected code is original. Castle: reference A3 and checkpoint A1. Harbour: A1 and A5, respectively. The checkpoint castle retains the geometry warnings observed during execution.

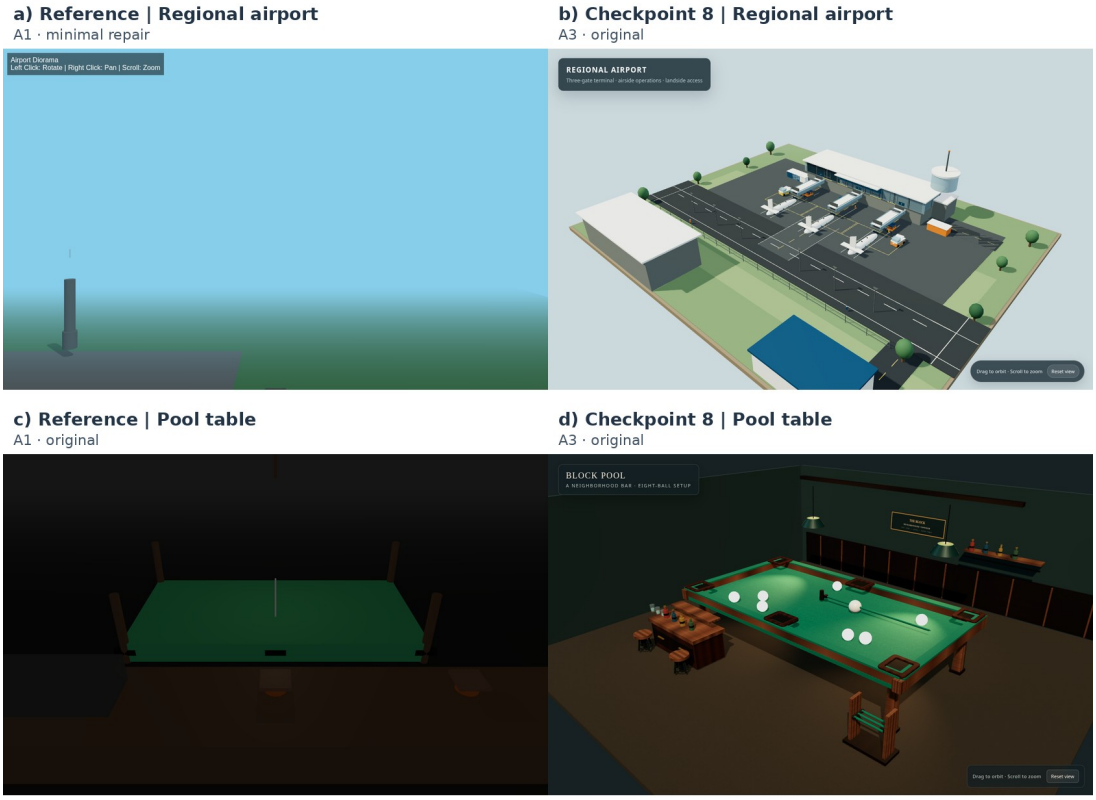


Figure 4. Regional airport and pool table. Reference on the left; Checkpoint 8 on the right. Airport: repaired A1 and original A3. Pool table: original A1 and A3. Screenshots preserve the generated cameras and scenes.

6.3. Additional Model-Generated Examples

Figure 5 presents further scenes generated by Checkpoint 8: voxel Rome, a rural landscape, a zoo and an interior with an emissive television. Rome, the landscape and the interior were each produced in a single generation without code editing. The three HTML files accompany the supplement so that these examples can be executed and inspected.



Figure 5. Additional scenes generated by Checkpoint 8: (a) voxel Rome; (b) rural landscape; (c) zoo; (d) interior with an emissive television.

6.4. Training Loss

Figure 6 shows a 100-update moving average for the final run, equivalent to 200 examples per complete window. Loss rises initially and then declines, with local fluctuations. Each observation is already weighted by the supervised targets in its update. No additional length normalisation or concatenation with earlier runs is applied.

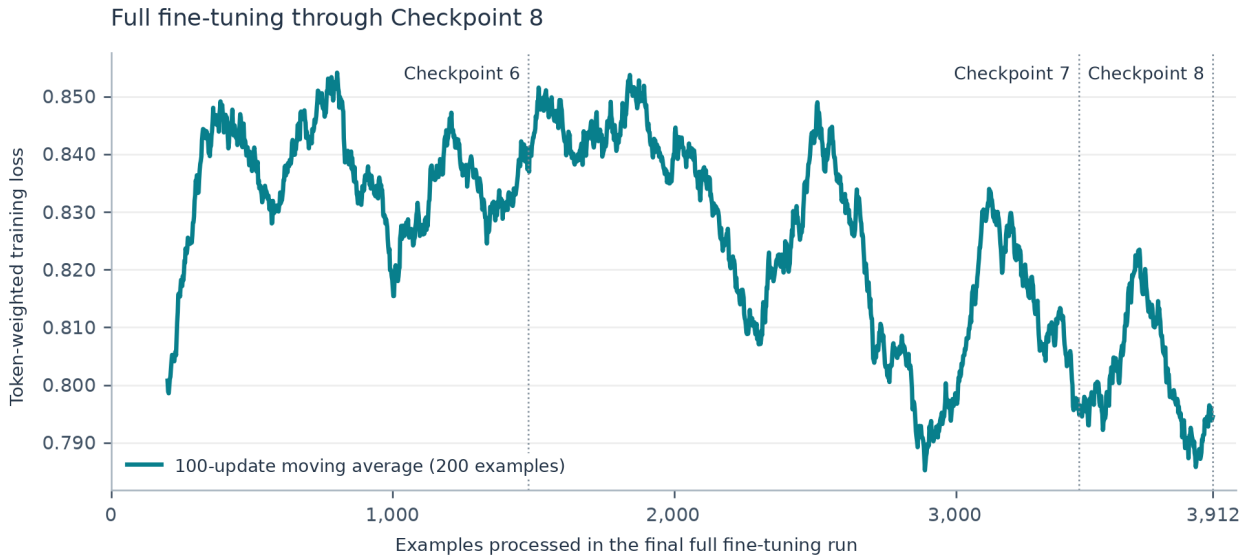


Figure 6. Training loss in the final run, smoothed with a 100-update moving average. The curve ends at example 3,912; vertical lines mark Checkpoints 6–8.

The final-run analysis uses training loss, without a validation curve for that stage. Validation losses from earlier stages were obtained with different datasets and implementations and are treated separately.

7. Discussion and Limitations

The central result is the generation of executable, recognisable scenes through supervised specialisation. The thematic and structural diversity of the data is a possible explanation for the observed gains. Because the corpus, training time and optimisation evolved during development, ablations can investigate the contribution of each factor.

The short LoRA stage was incorporated into the model. Longer branches initially improved, then showed lower scene quality and more repeated variable declarations. Full fine-tuning therefore resumed from Checkpoint 5. Context length, data, quantisation and optimisation also varied; this outcome does not establish a general limitation of LoRA. A relevant extension is to repeat the entire specialisation with LoRA from the base model, varying rank and context length to investigate whether comparable quality can be achieved.

The evaluation covers eight prompts used during development and different stopping rules for the reference and checkpoints. Its results are exploratory. An independent evaluation should use new prompts, the same attempt budget and separate criteria for execution, adherence and visual quality. Animation and temporal consistency also require examination beyond static screenshots.

The public dataset supports experiments with filtering by domain, complexity, style or sequence length, as well as comparisons between training subsets. These experiments can clarify the role of example distribution. Comparisons with larger models, between FP16 and Q8_0, and between SFT and SFT followed by RL are also useful extensions. Training and evaluation splits should remain separate in each configuration.

8. Conclusion

Specialising Qwen3.5-4B with synthetic data produced more recognisable and better-organised Three.js scenes than the selected reference outputs. The model resulted predominantly from full fine-tuning, with a short LoRA stage, in approximately 59 h 55 min of training incorporated into the final weights. The results show the potential of SFT for this task without additional RL. Releasing the data and weights enables broader evaluation and exploration of other adaptation strategies.

Artefact Availability

The dataset used in this study is available on Hugging Face at https://huggingface.co/datasets/ccore/ThreeJS-3D-Scenes [6]. FP16 weights and GGUF distributions are available at ccore/Qwen3.5-4B-3D-Coder [7]. The published versions use the MIT licence for the dataset and Apache-2.0 for the model.

Public artefacts associated with the study.

Artefact	Documented version
Dataset	4,146 examples: 3,933 training and 213 validation
FP16 weights	`b77dc455605e674d2478010dda4188ef5864a4be`
GGUF distributions	`15f353a67b292997a02a687b6c2d4e8751e25eff`

The supplement includes the figures and HTML files for the additional examples. Screenshots show the generated scenes, with presentation adjustments limited to labels and proportional sizing. The data-generation and curation pipeline remains separate from the public release.

AI tools supported manuscript organisation and language editing.

References

[1] Qwen Team. Qwen3.5-4B: official model card and configuration. 2026. https://huggingface.co/Qwen/Qwen3.5-4B

[2] Hu, E. J. et al. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685, 2021. https://arxiv.org/abs/2106.09685

[3] Dettmers, T.; Pagnoni, A.; Holtzman, A.; Zettlemoyer, L. QLoRA: Efficient Finetuning of Quantized LLMs. arXiv:2305.14314, 2023. https://arxiv.org/abs/2305.14314

[4] Zhou, C. et al. LIMA: Less Is More for Alignment. arXiv:2305.11206, 2023. https://arxiv.org/abs/2305.11206

[5] Three.js contributors. Three.js documentation. https://threejs.org/docs/

[6] ccore. CCORE Three.js 3D Scenes. Dataset and documentation, 2026. https://huggingface.co/datasets/ccore/ThreeJS-3D-Scenes

[7] ccore. Qwen3.5-4B-3D-Coder. Weights and documentation, 2026. https://huggingface.co/ccore/Qwen3.5-4B-3D-Coder

[8] ggml-org and contributors. llama.cpp. Conversion and quantisation tools. Revision 848bbda9041decd87904f7eb5a2990863ecce03b. https://github.com/ggml-org/llama.cpp/tree/848bbda9041decd87904f7eb5a2990863ecce03b

Appendix A. Training Configuration and Lineage

The tables describe the training stages incorporated into Checkpoint 8. Preliminary experiments and discarded LoRA branches were conducted separately.

Table A1. Stages incorporated into the selected weights.

Version	Stage	Train / validation	Updates
Checkpoint 1	Full fine-tuning; three epochs	717 / 80	270
Checkpoint 2	One epoch; Adam reset	1,915 / 213	240
Checkpoint 3	One epoch; Adam reset	3,933 / 213	492
Checkpoint 4	Continuation; Adam and RNG retained	3,933 / 213	640 additional
Checkpoint 5	LoRA; one epoch; adapter merged	178 / not used	89
Checkpoint 6	Final run; Adam reset	3,933 / 213	741 cumulative
Checkpoint 7	Final-run continuation	3,933 / 213	1,718 cumulative
Checkpoint 8	Selected endpoint	3,933 / 213	1,956 cumulative

Counters restart across runs. Checkpoints 6–8 have cumulative counts within one run and must not be added together. Validation examples were excluded from weight updates.

Table A2. Stage-specific hyperparameters.

Stage	Learning rate and regime	Microbatch / accum.	Seed / max. tokens
Checkpoint 1	10^{-5}; three epochs	1 / 8	20261021 / 68,808
Checkpoint 2	10^{-5}; one epoch	1 / 8	20261021 / 70,635
Checkpoint 3	10^{-5}; one epoch	1 / 8	20261021 / 71,229
Checkpoint 4	WSD: peak 10^{-6}; floor 10^{-7}	1 / 8	20261021 / 71,229
Checkpoint 5	Constant 5×10^{-5}; r = 16; α = 32	1 / 2	2026 / 16,384
Checkpoints 6–8	Cosine: peak 10^{-5}; floor 2×10^{-6}	1 / 2	2026 / 71,229

Sequence lengths are observed maxima, not the model's nominal context limit.

Checkpoint 4 used a warmup–stable–decay (WSD) schedule with 20 warmup updates, 492 stable updates and 128 decay updates. The final run used 59 warmup updates. Checkpoint 3 resumed with its training state preserved; the continuation producing Checkpoint 4 also retained the optimiser and random state.

Checkpoint 1 was selected at the end of the third epoch. Checkpoint 8 was selected through visual and functional assessment of the scenes, together with the additional tests presented in Section 6.3.

Appendix B. Protocol and Supplementary Results

B.1. Inference Configuration

Table B1. Exploratory comparison parameters.

Parameter	Configuration
Evaluated distribution	GGUF Q8_0
Attempts per case	Up to five; first accepted output
Seeds	1729, 1730, 1731, 1732 and 1733, in that order
Temperature / top-p / top-k	0.6 / 0.9 / 20
Min-p / repetition penalty	0 / 1
Context / KV cache	65,536 tokens / F16
Reasoning budget	32,768 tokens
Configured generation limit	64,512 tokens
Allowed repairs	Up to three local corrections in a separate code copy
Prohibited changes	Redesigning geometry, composition or camera

B.2. Execution Criteria and Observations

Checkpoint acceptance required a recognisable scene and the requested controls. The official reference stopped at the first technical render. Composition was assessed from the initial camera; orbit and zoom supported geometry and functionality checks. Figure 2 includes repairs, while Figures 3 and 4 identify selected attempts in their captions.

Table B2. Observations during evaluation.

Case	Observation
Rome — Checkpoint 8	The first attempt required a local const-to-let correction.
Harbour — Checkpoint 8	An earlier output received a better visual assessment but failed interaction checks. The fifth attempt was the first accepted.
Castle — Checkpoint 8	Controls worked despite NaN geometry warnings.
Reasoning loops	Eight of the 18 Checkpoint 8 calls were interrupted before delivering HTML and remain included in the total.

During development, some intermediate checkpoints received additional exploratory attempts, and the Checkpoint 7 assessment included a human exception. Cross-checkpoint analysis was therefore qualitative. For Checkpoint 8, all attempts and corrections followed the stated protocol.

B.3. Additional Scenes and Example Files

The supplement includes roma_voxels.html, paisagem_voxels.html and interior_televisao.html, corresponding to additional scenes generated by Checkpoint 8. Figure 5 presents screenshots of these examples and the zoo. The HTML files allow the scenes to be executed and the generated code to be examined.