DashVMC: Real-Time Discrete World Model Control in Geometry Dash

Florent Tariolle*
INSA Rouen Normandy

Website Code

Abstract

Fast deterministic games expose a practical world model problem: predictions must be visually sharp enough for control, action-conditioned over short timing windows, and cheap enough for live deployment. We present *DashVMC*, a real-time discrete Vision-Model-Controller pipeline for Geometry Dash. An FSQ tokenizer encodes 64×64 Sobel frames into 8×8 token grids; a block-causal transformer predicts next-frame tokens from action-conditioned context; and a lightweight actor-critic is warm-started by behavioural cloning and optimized with PPO in latent rollouts. The deployed agent runs at the 30 FPS cadence used by the captured training data, while measured optimized-path latency remains well within the 33 ms frame budget on an RTX 2060. Across 100 consecutive automated attempts, the frozen policy survives roughly ten seconds on each of the first two official levels; qualitative intervention runs show that it can clear most later sections when a small number of recurring blocking obstacles are skipped. The same action-conditioned dynamics can also be rolled forward from real gameplay prefixes to produce decoded visual continuations of plausible future Geometry Dash segments. We report the system design, latency profile, current model/controller metrics, and implementation diagnostics that motivated the final conservative architecture.

1 Introduction

Learning WMs that predict future observations from actions is central to model-based reinforcement learning [1]. Recent approaches tokenize visual frames into discrete codes and train autoregressive transformers to predict token sequences [2, 3]. Fast real-time games add a deployment constraint to this modeling problem: the agent cannot pause the environment while it plans, so perception, dynamics prediction, controller inference, and action dispatch must fit inside the frame budget. Geometry Dash is a useful stress case because it is visually simple but timing-sensitive: small prediction or action-delay errors can change survival.

Discrete tokenization is attractive in this setting because it gives sharp, compact states and turns next-frame prediction into classification. Finite Scalar Quantization [4] is especially convenient for a small real-time system. Each latent vector is independently rounded to one of L levels per dimension, yielding $\prod_d L_d$ implicit codes with no codebook collapse and no auxiliary losses. Each code maps to a point on a D-dimensional integer lattice, which gives the world model a compact discrete state space and a simple local notion of token neighbourhood when a loss-side prior is useful.

Our use of a discrete, reconstruction-anchored tokenizer is deliberate rather than incidental. In real-world video, much of the pixel stream is either unpredictable from the available context or irrelevant to control: lighting variation, sensor noise, sub-pixel texture, camera motion, and unobserved causes can make pixel-level prediction waste compute and even push the representation toward nuisance detail.

*Correspondence to florent.tariolle@insa-rouen.fr

This matches recent theory comparing reconstruction and joint-embedding SSL: when irrelevant components have large magnitude, latent-space prediction imposes weaker alignment requirements than reconstruction, whereas reconstruction is more competitive when such irrelevant components are weak [5]. For this reason, representation-space objectives such as JEPA are attractive in natural video because they can discard information that should not be predicted. Geometry Dash and Atari are different: they are rendered by deterministic, low-entropy state machines, so the pixels are a repeatable projection of the underlying discrete game state. In this setting, pixel reconstruction provides a useful ground-truth anchor for tokenizing the space, and preserving predictable pixel-level information can help rather than hurt downstream action.

We present the Geometry Dash Vision-Model-Controller (V-M-C) pipeline. The application is a fast-paced platformer that demands frame-accurate predictions under a 33 ms latency budget. Our system consists of three components: (1) an FSQ tokenizer that encodes 64×64 Sobel edge maps into 8×8 grids of discrete tokens, (2) a block-causal transformer that interleaves action tokens with frame tokens in a single sequence and predicts the next frame's tokens via parallel decoding, and (3) a lightweight CNN actor-critic on the current token grid, warm-started with Behavioural Cloning and fine-tuned with PPO [6] entirely in latent rollouts. Because the dynamics model is action-conditioned and autoregressive, it also acts as a procedural level-continuation generator: seeded with a short observed prefix and a candidate action sequence, it samples future token grids that decode into plausible continuations of the current Geometry Dash segment. The three components are trained sequentially – each on the frozen output of the previous one – and the deployed pipeline runs at the 30 FPS cadence used to capture the training data. The measured optimized loop averages 16.3 ms per frame on consumer hardware, so 30 FPS is a cadence choice rather than a compute ceiling.

The contribution is therefore task-specific systems co-design rather than a new generic WM component. FSQ provides a compact discrete representation with sharp reconstructions for the game's square-edged observations; action conditioning makes policy-dependent imagined transitions possible; BC reduces the cost of reaching a useful initial policy; and PPO moves expensive policy improvement into offline imagination, leaving a single reactive pass at deployment. Together these choices adapt recent WM techniques to consumer hardware on which inference-time planning would compete directly with the action-latency budget.

Our contributions are:

1. **A real-time discrete WM control system**: a complete V-M-C pipeline for Geometry Dash, combining FSQ tokenization, a transformer with action tokens interleaved in the sequence and an action-conditional contrastive auxiliary loss [7], and a CNN actor-critic trained via BC plus PPO in latent rollouts.
2. **Live deployment under a frame budget**: a screen-capture control loop configured at the 30 FPS training-data cadence, with measured end-to-end mean latency of 16.3 ms per frame and roughly 61 FPS of compute-equivalent throughput.
3. **Action-conditioned visual continuations and diagnostics**: decoded rollouts from real gameplay prefixes, plus implementation diagnostics for the tokenizer, action conditioning, controller, and scoped FSQ-neighbour smoothing choices.

2 Related work

Discrete WMs. The Vision-Model-Controller paradigm [1] introduced encoding frames with a VAE, modeling dynamics with an RNN, and training a controller in latent rollouts. IRIS [2] replaced the VAE with a discrete tokenizer and an autoregressive transformer, alternating tokenizer and dynamics updates. Our Geometry Dash system follows the same broad tokenizer-plus-transformer template with a frozen FSQ tokenizer and adds a real-time deployment loop for a non-paused game. DIAMOND [8] instead trains an agent inside a diffusion WM and shows that improved visual fidelity can benefit Atari control; our use case prioritizes a compact, single-pass discrete predictor under a strict live-action budget. DreamerV3 [3] demonstrated that categorical RSSM latents generalize across diverse domains; this approach was scaled further in [9]. TWISTER [7] added action-conditioned contrastive predictive coding to improve temporal coherence; we adopt their AC-CPC auxiliary loss. FSQ [4] is typically positioned as a frozen tokenizer for downstream tasks. We use its coordinate-lattice structure as a simple, deployment-friendly discrete state representation.

Observation-to-action bootstrapping. Recent autonomous-learning arguments distinguish observation-based learning from action-based learning and emphasize that useful initial structure can help bootstrap later interaction with the environment [10]. This view matches a long-standing practical pattern in RL from demonstrations: AlphaGo first trained policy networks from expert games before reinforcement learning through self-play [11], and DQfD used demonstrations to accelerate Atari reinforcement learning before and during environment interaction [12]. In a closer game-pretraining setting, Video PreTraining learned behavioral priors from human Minecraft videos and then fine-tuned them with imitation and RL for hard-exploration tasks [13]. Our Geometry Dash controller follows the same conservative recipe at smaller scale: expert trajectories provide a short BC warm-start, after which the policy is optimized by PPO in latent interaction with the learned world model.

Finite Scalar Quantization. FSQ [4] replaces learned codebooks with per-dimension rounding to fixed levels, eliminating codebook collapse and commitment losses. Each code is a point on a D-dimensional integer lattice, giving the codebook a natural coordinate structure. iFSQ [14] introduces a simple modification of the FSQ bounding function that improves bin utilization; we adopt it in our encoder. Recent Gaussian Quant (GQ) tokenizers [15] train a constrained Gaussian VAE and then convert posterior means into discrete codes with a fixed Gaussian codebook, reporting stronger image rate-distortion tradeoffs than FSQ and several VQ-style baselines on large-scale image benchmarks. Our use of FSQ should therefore not be read as a claim that FSQ is the strongest available image tokenizer. We use FSQ because it is simple, stable, fast enough for the real-time loop, and exposes an explicit coordinate lattice for defining local token neighborhoods in the Geometry Dash system. Replacing FSQ with a GQ-style tokenizer is an orthogonal and promising direction, but would require revalidating temporal prediction, control transfer, latency, and any token-neighbour metric used by the world model objective.

Joint-Embedding Predictive Architectures (JEPA). LeWorldModel [16] recently demonstrated a stable end-to-end JEPA WM from pixels using continuous latents and a SIGReg anti-collapse regularizer. We explored a JEPA-style joint-embedding variant of our system, in which context and target frames are encoded by a single online FSQ encoder into a shared discrete latent space and prediction gradients flow back into the encoder through a straight-through path, with the pixel decoder retained only as an anti-collapse regularizer. This was a constrained FSQ-token hybrid, not a faithful LeWorldModel reproduction: canonical SIGReg regularizes continuous Gaussian embeddings, whereas our dynamics model and controller consume bounded FSQ coordinates and hard token IDs. The variant did not improve on the sequentially-trained baseline reported here; we summarize the negative result and the FSQ-compatible anti-collapse surrogate we attempted in Section 5.5. We view this as a domain- and architecture-specific observation rather than as evidence against JEPA: in natural video, avoiding reconstruction of unpredictable pixels is often a feature [5], while in deterministic game domains the rendered pixels are a predictable supervision signal tied closely to the latent state.

Label smoothing. Label smoothing [17] proposed replacing one-hot targets with a uniform mixture to reduce overconfidence. While effective as a regularizer, uniform smoothing ignores class relationships. Knowledge distillation and soft-target methods distribute mass based on teacher predictions, but require a separate teacher model. For the Geometry Dash world model, we use the FSQ lattice to define a local smoothing target without an additional teacher model. Section 3.2 also reports the broader annealed variant as a diagnostic rather than as a general replacement for cross-entropy.

Adaptive conditioning. FiLM [18] introduced adaptive affine conditioning via scale and shift. DiT [19] applied this as AdaLN-Zero with zero-initialized gates for stable training of diffusion transformers, and QK-norm [20] (RMSNorm on queries and keys per head) was introduced to stabilize attention logits in deep AdaLN models. LeWorldModel [16] applied AdaLN to world model transformers. Our baseline keeps actions as tokens in the sequence; we treat AdaLN-Zero conditioning as an ablation (Section 5.5).

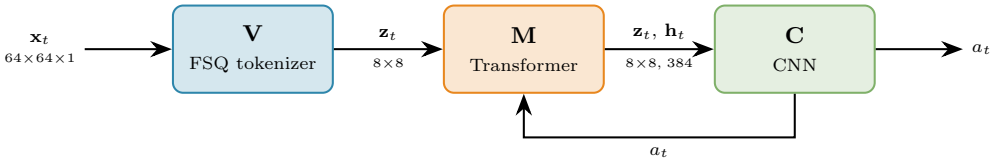

Figure 1: Vision-Model-Controller pipeline. The FSQ tokenizer (V) maps Sobel edge maps to a grid of discrete codes, the block-causal transformer (M) predicts next-frame tokens with action tokens interleaved in the sequence, and the CNN controller (C) reads the current token grid alongside the transformer hidden state to select actions. During live play the grid comes from the current observation; during autoregressive dreaming, grids after the seed context are model-generated.

3 Method

Our system follows the Vision-Model-Controller (V-M-C) paradigm [1] (Figure 1). We describe each component below, including Structured Label Smoothing (Section 3.2) as the FSQ-metric loss used in the Geometry Dash world model. The three components are trained sequentially – each on the frozen output of the previous one – a deliberate choice motivated by the ablation results in Section 5.5.

3.1 FSQ tokenization (Vision)

We encode 64×64 grayscale Sobel edge maps using an FSQ autoencoder with levels $[8, 5, 5, 5]$, producing 1000 implicit codes. The encoder applies three stride-2 convolutional stages with residual blocks, projecting to a 4-channel latent map of size 8×8. Each spatial position is independently quantized using the iFSQ bounding function [14], $2\sigma(1.6z) - 1$ in place of $\tanh(z)$, scaled to half-levels and rounded, yielding 64 tokens per frame. The encoder is trained on consecutive frame pairs with a pixel-MSE reconstruction loss and the two geometric regularizers from GRWM [21]: a temporal slowness term that penalizes large $\|\mathbf{z}_e^{t+1} - \mathbf{z}_e^t\|^2$ between adjacent frames, and a uniformity term that spreads pre-quantization features across the FSQ hypercube. The decoder mirrors the encoder with transposed convolutions and is discarded after training.

3.2 Structured Label Smoothing

Motivation. Standard label smoothing [17] replaces a one-hot target $\mathbf{e}_i$ with

$$q_{\text{uniform}}(j \mid i) = (1 - \varepsilon)\,\delta_{ij} + \frac{\varepsilon}{V - 1}(1 - \delta_{ij}), \tag{1}$$

where $V = 1000$ is the vocabulary size and ε controls the smoothing strength. This distributes ε uniformly across all alternatives, ignoring codebook geometry.

Token metric. SLS assumes a distance $d(i, j)$ between target token i and alternative token j. For FSQ, a codebook with levels $\mathbf{L} = (L_1, \ldots, L_D)$ assigns each code i a coordinate vector $\mathbf{c}(i) = (c_1(i), \ldots, c_D(i))$ on a D-dimensional integer lattice, giving the squared lattice distance

$$d^2(i, j) = \sum_{d=1}^{D} \left(c_d(i) - c_d(j)\right)^2 . \tag{2}$$

For a learned VQ codebook or an IRIS-style tokenizer, the same construction can use codebook-vector or embedding distance instead. This makes the construction portable in form, but the present paper validates it only as an FSQ-neighbour design choice inside the Geometry Dash system.

Kernel-family study. Given $d(i, j)$, we compare three kernels:

$$k_{\text{Gauss}}(d) = e^{-d^2/2\sigma^2}, \qquad k_{\text{Laplace}}(d) = e^{-d/\sigma}, \qquad k_{\text{Cauchy}}(d) = \frac{1}{1 + d^2/\sigma^2}. \tag{3}$$

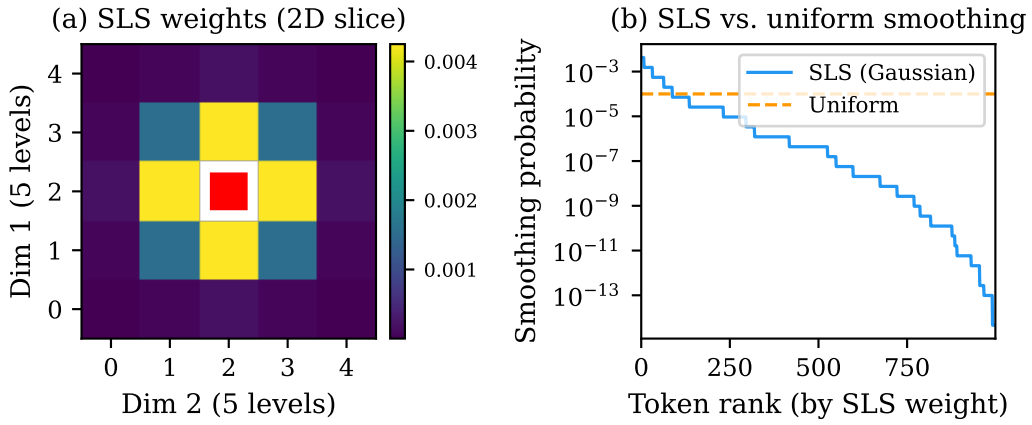

Figure 2: Illustrative SLS target distribution for an FSQ token near the centre of the codebook. **(a)** 2D slice over two lattice dimensions with the remaining two held at the target's values: weights decay as a Gaussian kernel over lattice distance, red square marks the target. **(b)** All $V - 1$ alternative tokens sorted by SLS weight (log scale) vs. uniform label smoothing. SLS concentrates mass on near-miss tokens while suppressing distant ones.

The three families differ in tail behaviour. Cauchy has polynomial tails and leaks meaningful mass to distant neighbours; Laplace has a cusp at $d = 0$ and an exponential tail; Gaussian has the thinnest tails, decaying faster than either alternative once d exceeds the bandwidth. We take a Gaussian over the full codebook as the current default: thin tails ensure distant neighbours receive negligible mass while the nearest alternatives still receive meaningful target probability.

Metric calibration. A natural alternative is per-dimension weights α_d calibrated from an empirical sensitivity probe: perturbing each FSQ dimension by ± 1, decoding, and measuring reconstruction MSE. The probe gives data-dependent weights, but ties the smoothing target to a particular trained codebook and varies across training runs. For the FSQ Geometry Dash system, we use an isotropic kernel and fix σ by a geometric rule: choose the bandwidth so that the immediate integer lattice neighbour sits near the kernel's half-maximum. For embedding-distance baselines, the corresponding hyperparameters are the smoothing mass, kernel temperature or width, and, if used for scalability, top-k truncation. The sensitivity probe is retained as an analysis tool to characterize a learned codebook's anisotropy and cross-dimension coupling, but not as a hyperparameter source.

Structured smooth target. We replace the uniform distribution in Equation 1 with a kernel over token distances:

$$q_t(j \mid i) = \begin{cases} 1 - \varepsilon_t & \text{if } j = i, \\ \varepsilon_t \cdot \dfrac{k(d(i,j))}{\sum_{l \neq i} k(d(i,l))} & \text{otherwise.} \end{cases} \tag{4}$$

For the $V = 1000$ vocabulary used here, we precompute the full dense target matrix $\mathbf{Q} \in \mathbb{R}^{V \times V}$ once per kernel choice; top-k truncation is only a possible scalability option for larger codebooks (Figure 2).

Annealing diagnostic. The smoothing mass ε_t is scheduled during training. Our IRIS/Pong diagnostic schedule uses $\varepsilon_t = 0.1$ through the early training phase, cosine-anneals from 0.1 to 0 during the middle phase, and then trains with pure CE. This makes the optimization a curriculum: early targets encode neighbourhood structure and reduce brittleness, while the final objective is exactly the hard CE baseline. That broader benchmark test did not produce a robust positive result, so annealing is reported only as an auxiliary diagnostic. For Geometry Dash, we keep SLS fixed: exact token identity is a stricter target than decoded or control-relevant equivalence, and the application is precisely the setting where visually similar FSQ neighbours motivated the loss.

Integration with focal loss. To handle class imbalance (background tokens dominate over obstacle and spike tokens), we apply focal modulation to the SLS soft-target cross-entropy [22]. For target token i, define

$$\ell_{\text{SLS}}(i) = -\sum_{j=1}^{V} q_t(j \mid i) \log p_j, \qquad \bar{p}_i = \exp[-\ell_{\text{SLS}}(i)], \tag{5}$$

and compute

$$\mathcal{L}_{\text{SLS-focal}}(i) = (1 - \bar{p}_i)^{\gamma}\, \ell_{\text{SLS}}(i), \tag{6}$$

where $p_j = \text{softmax}(\mathbf{z})_j$ and $\gamma = 2.0$; the training objective averages this quantity over target positions. Thus focal modulation is applied once to the aggregate soft-target cross-entropy, not independently to each class term.

3.3 Transformer WM (Model)

The WM is a block-causal transformer (384 embedding dim, 8 heads, 8 layers) that predicts the next frame's tokens in parallel given a context of $K = 4$ past frames and their associated actions, on top of a frozen FSQ tokenizer (Section 3.1).

Sequence construction. Each frame contributes 64 visual tokens (from the 8×8 FSQ grid) plus one status token (ALIVE or DEATH), for 65 positions per frame block. Actions are embedded into the same 384-dimensional space and inserted as single tokens between consecutive frame blocks, giving the input sequence

$$[f_0\, a_0\, f_1\, a_1\, \ldots\, f_{K-1}\, a_{K-1}\, f^{\text{tgt}}]$$

of length $K \cdot (65 + 1) + 65 = 329$. All target-frame positions are replaced with a learnable `[MASK]` embedding so the model never sees the ground-truth target tokens during training.

3D Rotary Position Embedding. We apply factored RoPE [23] encoding three positional axes: spatial row, spatial column, and temporal frame index. The head dimension is partitioned into three bands (rows, columns, time) with separate base frequencies ($\theta_{\text{spatial}} = 10$, $\theta_{\text{time}} = 10{,}000$). Status and action tokens receive the grid centre coordinates to represent whole-frame summaries.

Block-causal attention. Within each frame block, attention is bidirectional (tokens attend to all positions in the same frame). Across frame and action blocks, attention is strictly causal: each block attends only to current and past blocks. At inference, the next frame's 64 visual tokens and its status position are decoded in the same parallel forward pass from the masked target block. The death probability is the two-class softmax over the `ALIVE` and `DEATH` logits at that final status position. After normalizing the encoded context, we define $\mathbf{h}_t$ as the hidden state at its final position, which for the reported in-sequence design is the most recent interleaved action token.

Action-conditional contrastive auxiliary loss. Following TWISTER [7], we add an action-conditional contrastive predictive coding (AC-CPC) auxiliary loss: for each $k \in \{1, \ldots, K\}$, an MLP head predicts the future hidden state $\mathbf{h}_{t+k}$ from the current $\mathbf{h}_t$ and the intervening actions $a_{t:t+k-1}$, scored against in-batch negatives with InfoNCE. The auxiliary loss enters the total objective with weight $\lambda_{\text{cpc}} = 0.1$.

Context-token noise. The reported transformer uses two context-noise branches. Each visual context token is independently eligible for 5% uniform random-token replacement and for 5% FSQ-neighbour replacement, where the latter perturbs one lattice dimension by ± 1 subject to codebook boundaries. Noise is applied only to context visual tokens, not target or status tokens.

Output projection and total loss. The output projection shares weights with the input embedding (weight tying). The total training loss is

$$\mathcal{L} = \mathcal{L}_{\text{SLS-focal}} + \lambda_{\text{cpc}}\, \mathcal{L}_{\text{AC-CPC}}, \tag{7}$$

applied to the transformer parameters with the FSQ tokenizer frozen.

3.4 PPO controller (Controller)

The controller is a lightweight CNN actor-critic, in the spirit of IRIS [2] and DIAMOND, that reads the current token grid alongside the transformer's hidden state and outputs a jump probability and a state value estimate. During live play, the grid is encoded from the current observed frame; during latent rollouts, the first grid comes from the recorded seed context and subsequent grids are sampled from the frozen world model.

Architecture. Token IDs are embedded into a small learnable space and reshaped into the 8×8 spatial grid; two stride-1 convolutions with max-pooling produce a 256-dimensional feature vector, which is concatenated with the transformer's hidden state $\mathbf{h}_t \in \mathbb{R}^{384}$ and passed to zero-initialized linear actor and critic heads. A third zero-initialized multi-token-prediction head predicts the next eight binary actions from the same features; during PPO, its mean binary cross-entropy enters the loss with coefficient 0.1. The full controller has ~ 45K parameters.

Training in latent rollouts. The controller is first pretrained via Behavioural Cloning (BC) on expert demonstrations, then fine-tuned with PPO [6] in latent rollouts. We use BC as an initialization mechanism rather than as the final learning rule: demonstrations supply a non-random policy from observation, and PPO then performs action-based improvement in the model's imagination [10–13]. At each iteration, several hundred rollouts of up to 45 steps are generated by autoregressively sampling from the frozen WM. The 45-step cap approximately matches the time required for the side-scrolling scene to traverse the 64-pixel observation width, so content inherited from the real seed has scrolled out by the horizon. Longer diagnostic dreams can remain coherent for hundreds of steps, but they sometimes enter an absorbing empty-screen failure mode in which the locally most likely continuation contains no new obstacle; we therefore do not use those longer horizons for policy optimization. Rollouts terminate when the predicted death probability exceeds 0.5. Rewards are $+1$ per survival step with a -0.25 penalty per jump to discourage gratuitous jumping (an idle agent dies immediately upon reaching an obstacle, while a jumping agent survives longer by spending time airborne, creating a deceptive local optimum). We use GAE [6] with $\gamma = 0.995$, $\lambda = 0.95$, and clip ratio $\varepsilon_{\text{PPO}} = 0.2$. A Dreamer-style λ-return actor-critic is a natural replacement: on-policy imagined rollouts make PPO's off-policy clipped surrogate unnecessary. We leave this to future work.

4 Experimental setup

Environment. Geometry Dash is a rhythm-based platformer where the player controls a cube moving at constant horizontal speed. The only action is a binary jump. Colliding with any obstacle causes instant death. Levels are deterministic: the same action sequence always produces the same outcome, but frame-accurate timing is required.

Data collection. We collect roughly 4,200 gameplay episodes ($\sim$250K frames) including both intentional-death episodes (covering diverse failure modes) and 36 expert trajectories/segments. The expert subset contains 33,153 frames spanning 1,137.1 seconds (18.95 minutes) of recording; individual segments contain 91–2,601 frames and last 3.13–87.90 seconds (median 530 frames and 18.59 seconds). Per-recording metadata preserves a numeric capture-level identifier, target and measured frame rates, and a timestamp, but does not annotate the in-level start state or an explicit completion status. We therefore do not treat every segment as a verified full-level clear. We use one deterministic 90/10 train/validation split (seed 42), stratified separately over intentional-death and expert sources and shared by the tokenizer, WM, and BC stages. Splitting is performed on base episode identifiers before augmentation; all shifted versions inherit the assignment of their base episode. Vertical shift augmentation ($\pm 2, \pm 4$ pixels) expands the dataset $5\times$. Frames are preprocessed with Sobel edge detection and resized to 64×64. Before transformer training, all 4,264 base episodes are retokenized with the selected frozen FSQ checkpoint; the base data and four shifted variants yield exactly 21,320 tokenized episodes. The archived world-model job reports consuming the same 21,320 episodes. We retain the checkpoint hash, tokenization and training timestamps, job identifier, and relevant log extracts with the released artifacts.

Training details. The FSQ autoencoder and transformer are trained sequentially. The FSQ autoencoder is trained for 1,000 epochs with Adam (batch 2,048, initial LR 10^{-3}, cosine decay to 10^{-5}, no

warmup), with the slowness and uniformity regularizers weighted at $\alpha_{\text{slow}} = 0.1$ and $\alpha_{\text{uniform}} = 0.01$. Validation and best-checkpoint selection occur every ten epochs. The selected encoder is then frozen and used to retokenize the corpus. The transformer is trained for 200 epochs with AdamW (batch 512, 500 optimization steps per epoch, initial LR $2 \cdot 10^{-3}$, five-epoch linear warmup followed by cosine decay to $5 \cdot 10^{-5}$, weight decay 0.01, dropout 0.1, and gradient clipping at 1.0). Its objective uses fixed Gaussian SLS ($\varepsilon = 0.1$, $\sigma = 0.9$, focal $\gamma = 2$), the two 5% context-noise branches described above, AC-CPC weight 0.1, and $5\times$ death-frame oversampling. The Geometry Dash transformer uses fixed SLS with smoothing mass 0.1, rather than the Atari annealing schedule, because its role is to preserve a local semantic neighbourhood over FSQ codes rather than to provide a late-stage exact-CE benchmark comparison. BC is trained for 50 epochs with AdamW (batch 512, LR 10^{-3}, weight decay 10^{-4}, positive jump-class weight 1.5) and selected by validation loss. PPO uses 512 parallel imagined episodes per iteration, four update epochs with minibatches of 512, Adam at LR 10^{-4}, entropy coefficient 0.01, critic coefficient 0.5, gradient-norm clipping at 0.5, and a fixed 15,000-iteration budget. The remaining rollout parameters are $\gamma = 0.995$, GAE $\lambda = 0.95$, clip ratio 0.2, jump penalty 0.25, MTP coefficient 0.1, 45-step horizon, and death threshold 0.5. All Geometry Dash stages use seed 42. BF16 mixed-precision training is used throughout; BF16 requires no loss scaling and is numerically more stable than FP16. Training runs on a single NVIDIA A100 GPU.

Training throughput. We benchmarked `torch.compile` modes under BF16 on our joint-training workload using subprocess isolation to prevent CUDA allocator fragmentation across runs. All `torch.compile` modes are within a few percent of each other; we adopt `reduce-overhead` with `fullgraph=True` as the default: it is faster than `default` mode while avoiding the lengthy autotune phase that `max-autotune` adds at startup in exchange for marginal additional steady-state throughput. This configuration is applied to the world model and controller training scripts. A secondary NVIDIA RTX 2060 (Turing, SM 7.5) was used in parallel for auxiliary experiments. Turing lacks native BF16 (software-emulated and much slower than FP32 on this card) and has no TF32 support; Inductor's channels-last layout heuristic additionally triggers a stride assertion in the convolution backward, which we disable by setting `torch._inductor.config.layout_optimization = False`. Under these constraints the best local configuration is `torch.compile` in `default` mode with FP16 and a `GradScaler`.

Real-time inference. The deployed agent processes each frame at a fixed 30 FPS cadence, with a per-iteration sleep padding any spare time so the capture rate stays constant rather than racing the GPU. Within each iteration we minimize stalls: the captured image is staged into a pinned-memory tensor and uploaded with `non_blocking=True`, so the CPU returns from the H2D copy immediately and queues the subsequent CUDA dispatches (FSQ encode, transformer, controller) without waiting for the transfer to physically complete. The transformer's context window is held in a GPU-resident buffer of FSQ token indices, eliminating a GPU→CPU→GPU round-trip per frame. The FSQ encoder is compiled with `torch.compile`, while transformer context encoding is captured in a CUDA Graph after a short warm-up. Under these optimizations, 5,000 wall-clock samples from capture through the keyboard action update average 16.3 ms per frame on a single RTX 2060 (median 16.3 ms; p95 18.7 ms), well within the 33 ms 30-FPS budget; reciprocal mean latency corresponds to approximately 61 FPS of compute-equivalent throughput. This latency budget also determines the scope of downstream comparisons. We do not treat inference-time planning agents, including full LeWorldModel-style planning systems, as direct baselines for live Geometry Dash control: the game cannot be paused, so any online search or trajectory optimization would have to fit inside the same 33 ms loop as capture, preprocessing, model inference, and action dispatch. Our architecture therefore follows a reactive deployment regime: expensive optimization happens offline in imagined rollouts, while the deployed controller executes one fast policy step per frame. Planning-based WMs remain relevant in settings where additional online computation is available, but they answer a different control question from this real-time deployment setting.

BC pretraining ablation. To verify that BC pretraining is a convergence accelerator rather than a stability requirement, we train one PPO run from random initialization. Cold-start PPO converges to the same plateau as BC-initialized PPO on all metrics (eval survival, entropy, jump ratio) but requires roughly 2,000 additional iterations (≈6 hours on A100); BC itself takes ∼5 minutes, giving a large compute ROI, consistent with prior demonstration-based RL results where demonstrations mainly improve early learning and sample efficiency [12, 13]. BC is retained in the pipeline on this basis.

Table 1: Selection-aligned frozen DashVMC system metrics. Model rows report metrics at the model-selection point rather than independent maxima across training; each live row contains 100 consecutive automated attempts.

Component	Metric	Value
FSQ tokenizer	Selected validation MSE / PSNR	3.894×10^{-4} / 34.10 dB
World model	Selected validation token accuracy	29.74%
World model	Selected validation death precision / recall	$0.7337/0.8654$
World model	Selected validation death F1	0.7941
BC controller	Selected-epoch validation accuracy	79.76%
PPO controller	Selected latent development survival	$29.71/45$ steps
Live Level 1	Mean survival (95% CI)	279.6 $[270.5, 289.2]$ frames
Live Level 2	Mean survival (95% CI)	263.3 $[239.8, 287.2]$ frames
Live Level 3	Mean survival (95% CI)	64.3 $[59.3, 70.0]$ frames

Evaluation metrics. We report: (1) next-token prediction accuracy and cross-entropy on held-out episodes, (2) death prediction precision/recall/F1, (3) BC action prediction accuracy, (4) PPO survival on fixed latent-rollout development contexts, (5) live-loop deployment latency, and (6) acted frames survived in the real game. For PPO checkpoint development, we pre-sample 512 fixed contexts from the nominal validation episodes and reevaluate them every ten iterations. Because PPO rollout seeds are drawn from the full episode collection, including those episodes, and the best checkpoint is selected adaptively on this fixed set, we report this quantity as a development metric rather than held-out validation. For the controlled deployment evaluation, we freeze the selected checkpoints and run 100 consecutive attempts on each of the first three official levels at 30 FPS with Auto-Retry enabled. The evaluator uses game memory only to delimit alive/dead episodes; the reported outcome is acted frames survived rather than level percentage. We compute each confidence interval of the mean using a percentile bootstrap over attempts with 50,000 resamples. For the application section, we retain qualitative generated rollouts: the WM is seeded with real gameplay frames and action tokens, then rolled forward to visualize level continuations and failure/survival branches in token space.

5 Results

5.1 Geometry Dash system metrics

Table 1 summarizes the frozen DashVMC checkpoints and controlled live evaluation. The selected tokenizer checkpoint (epoch 920, minimum validation reconstruction loss) has held-out reconstruction SSE 1.595 per frame on pixels normalized to $[0, 1]$. Because squared errors are summed over each 64×64 frame and averaged over samples, this is an aggregate per-pixel MSE of 3.894×10^{-4}, or 34.10 dB PSNR computed from that mean MSE. The selected transformer checkpoint (epoch 139, maximum validation death F1) reaches 29.74% held-out token accuracy, with death precision 0.7337, recall 0.8654, and F1 0.7941. The BC checkpoint selected by minimum validation loss at epoch 9 reaches 79.76% held-out action accuracy, and the PPO checkpoint selected at iteration 3250 reaches fixed-context latent-rollout development survival of 29.71 out of 45 steps. On Levels 1 and 2, the frozen policy survives 279.6 and 263.3 acted frames on average, with overlapping 95% bootstrap CIs of $[270.5, 289.2]$ and $[239.8, 287.2]$. Mean survival on Level 3 is 64.3 frames (95% CI $[59.3, 70.0]$), and 75% of attempts end by frame 93. This reduction is expected to some degree: the official levels increase in difficulty, and Level 3 introduces yellow orbs that require an additional timed jump input while airborne. We therefore report each level separately rather than interpreting the values as a controlled generalization comparison. In separate qualitative intervention runs, bypassing a small number of recurring blocking obstacles allows the same policy to clear most later sections; this observation is not included in the controlled survival statistics.

5.2 Real-time deployment

Table 2 reports the deployment timing of the full screen-capture loop. The system is configured at 30 FPS because the training data were captured at that cadence and because a fixed cadence avoids

Table 2: Per-stage means and measured wall-clock latency over 5,000 optimized deployment frames on an RTX 2060.

Stage	Mean (ms)
Screen capture	2.2
Crop	0.5
Grayscale	0.5
Sobel	2.5
Downscale	2.2
FSQ encode	2.1
Transformer	4.9
Controller	0.5
Component-sum mean	15.3
Measured end-to-end	**16.3**

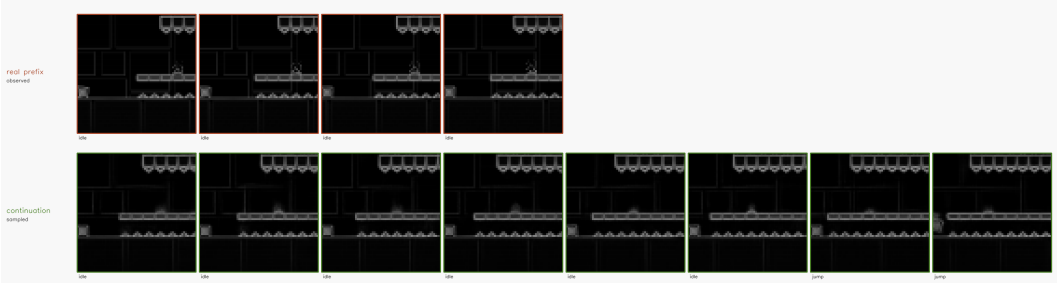

Figure 3: Decoded visual continuation from a real gameplay prefix. The first four frames are recorded Sobel observations; the remaining frames are sampled autoregressively from the selected world model under the recorded future action sequence and decoded through the FSQ decoder.

racing the capture stream. The measured end-to-end latency averages 16.289 ms per frame on an RTX 2060, including screen capture, Sobel preprocessing, FSQ encoding, transformer inference, controller inference, and the keyboard action update when required; the deliberate cadence sleep is excluded. Median latency is 16.308 ms and p95 is 18.722 ms; only 2 of 5,000 frames exceed the 33.3 ms budget. Thus the 30 FPS setting is not the compute ceiling: reciprocal mean latency corresponds to approximately 61 FPS of compute-equivalent throughput, rather than a measured 61-FPS deployment.

These optimized-path samples are distinct from the diagnostic timings emitted by the controlled evaluator: that script transfers token context through CPU/NumPy and synchronizes CUDA after every GPU stage to isolate component times, operations that are absent from deployment. We therefore use the evaluator only for survival outcomes and the optimized live path for latency.

5.3 Action-conditioned visual continuations

Because the transformer is action-conditioned and autoregressive, it can be seeded with real gameplay prefixes and sampled forward under candidate action sequences. Decoded rollouts provide a qualitative inspection tool: generated futures should preserve the local Geometry Dash layout, obstacle timing, and survival/death branches implied by the action sequence. The decoder is used only for visualization in this setting; policy optimization operates on token grids and hidden states.

5.4 Auxiliary SLS diagnostic

To test whether the FSQ-neighbour smoothing idea transfers beyond Geometry Dash, we ran a matched IRIS/Pong comparison where the world model loss target is the only intended change. The five-seed result does not support using annealed SLS as a general discrete-world-model objective.

Condition	n	Final return	Tail 500–600	Failure tail < 10
CE	5	15.68 ± 4.96	15.09 ± 5.59	20%
Annealed SLS	5	12.14 ± 10.82	12.55 ± 8.63	40%
Fixed SLS	2	10.06 ± 14.05	8.53 ± 16.21	50%

Annealed SLS shows partial stability signals: its average max drawdown and return-difference volatility are lower than CE, and its mean return is slightly higher in the 250–420 epoch window. Those benefits do not translate into better asymptotic robustness: annealed SLS has lower final and tail returns and twice the catastrophic tail-failure rate of CE in this run set. We therefore treat IRIS/Pong as an auxiliary diagnostic. It supports only the narrower claim that FSQ-neighbour smoothing is a plausible Geometry Dash prior; it does not support a broader method claim for annealed SLS across discrete world models.

5.5 Ablations and negative results

We report a set of architectural changes which, on paper, were expected to improve over the baseline of Section 3, but did not yield real-game gains in our experiments. Together they motivate the deliberately conservative system design of the current paper.

Joint training of FSQ and transformer. We trained a JEPA-style variant in which the FSQ encoder and the transformer are optimized together in a single AdamW group, with prediction gradients routed back into the encoder through a learnable straight-through projection (`fsq_grad_proj`) inserted between the pre-rounding continuous latents and the transformer input. A light pixel-MSE loss is retained as an anti-collapse regularizer on the encoder, and a single global gradient-norm clip is applied to the combined parameter set following [16]. This setup trains stably and produces lower validation cross-entropy than the sequentially-trained baseline, but the resulting FSQ codebook drifts toward a representation that is harder to act on: real-game level progress consistently dropped relative to the sequentially-trained baseline of identical capacity. Removing the pixel-MSE anchor (pure joint embedding) causes the encoder to collapse. We did not run canonical SIGReg in this discrete variant: LeWorldModel regularizes continuous embeddings toward an isotropic Gaussian, while our tokenizer exposes bounded FSQ coordinates and hard token IDs. Instead, we tried an FSQ-compatible distribution-matching surrogate inspired by SIGReg, using a Cramér-von-Mises-style test against factorized code utilization; it was insufficient to prevent joint-distribution degeneracy when the prediction target comes from the online encoder. We therefore keep the reconstruction-anchored tokenizer for the reported system. This choice should not be read as a faithful comparison to LeWorldModel or as a claim that pixel prediction is generally preferable to JEPA: in natural videos, forcing a model to spend capacity on unpredictable pixel detail can waste compute and degrade the representation, consistent with theoretical and empirical evidence favoring joint-embedding objectives when irrelevant input components are high-magnitude [5]. The comparison is further shaped by our observation space: the tokenizer reconstructs grayscale Sobel edge maps rather than raw RGB frames, so preprocessing already removes much of the color, texture, and background variation that joint-embedding objectives are often meant to ignore. This makes reconstruction a less wasteful anchor here than it would be from raw natural video, and gives the JEPA-style variant less room to differentiate itself within this constrained architecture. Our claim is narrower: in deterministic discrete games, where pixels are predictable outputs of the simulator and small pixel errors often correspond to state errors, reconstruction is a useful anchor for a token space that the dynamics model and controller can share. Finding a faithful decoder-free discrete anti-collapse signal, or a full continuous-latent LeWorldModel-style Geometry Dash system, remains open work.

AdaLN-Zero action conditioning. Rather than inserting actions as tokens, we conditioned each transformer block on the action history via Adaptive Layer Normalization [19, 18]. The K-step action sequence was projected through an MLP to per-layer scale, shift, and gate parameters modulating each sublayer. To stabilize training we needed both QK-norm [20] (RMSNorm on queries and keys per head) and gate-bias initialization to 1 rather than DiT's standard zero: zero-initialized gates suppressed both conditioning and learning, and without QK-norm, attention logits diverged. Even with both fixes, the AdaLN variant matched but did not surpass the in-sequence-action baseline on real-game level progress, while adding architectural complexity. The token-in-sequence design is what we ship.

Larger transformer ($384 \to 512$). Scaling the embedding dimension from $384 = 384$ to 512 at otherwise matched capacity improves dream-quality metrics (validation cross-entropy and AC-CPC score) but does not translate into real-game gains, with extra training time per epoch. We retain the smaller model.

MLP controller on $\mathbf{h}_t$ alone. We tested a controller that consumes only the transformer hidden state $\mathbf{h}_t$, with no CNN over the current token grid, motivated by the fact that $\mathbf{h}_t$ already encodes spatial and temporal structure through 8 transformer layers with 3D-RoPE. This MLP variant converges 2–$5\times$ faster in PPO and has the smallest train-eval gap of any controller we trained, but the CNN-on-tokens variant from Section 3.4 dominates it on real-game level progress, suggesting the explicit token-grid signal carries information that is washed out in the global hidden-state summary.

6 Limitations

The current benchmark evidence for annealed SLS does not support a broad method claim: on IRIS/Pong with five CE and five annealed-SLS seeds, CE has better final/tail return and a lower catastrophic-failure rate. The result leaves open whether a different tokenizer metric, schedule, or game could make SLS useful, but that is outside the scope of this paper. The full Atari controller path did not become a reliable evaluation vehicle and is explicitly out of scope for this paper. Geometry Dash remains a single-game application with a binary action space and deterministic level layouts. The controlled live evaluation covers 300 attempts of one frozen policy across three deterministic levels of increasing difficulty. It measures deployment variability, not uncertainty over training seeds, and the current evaluator measures survival duration rather than level percentage. Because Level 3 introduces a timed mid-air yellow-orb mechanic, its lower survival cannot by itself be interpreted as a generalization failure. The level-generation claim is likewise scoped to autoregressive visual continuations sampled from the learned dynamics model, not to exporting editor-valid Geometry Dash levels with guaranteed playability. The generalization of SLS to environments with higher-dimensional actions, stochastic dynamics, or RGB observations remains to be validated. We also do not claim a downstream control comparison against planning-based WMs under relaxed inference budgets; our Geometry Dash system targets a non-paused 30-FPS setting where online planning must compete with the action latency budget. The system is trained sequentially – FSQ, then transformer, then controller – which is operationally simple but forfeits the possibility of prediction gradients shaping the codebook. Our preliminary attempts at a JEPA-style joint-training variant inside the existing FSQ-token architecture did not produce real-game gains over this sequential baseline (Section 5.5); whether a faithful continuous-latent LeWorldModel adaptation, a more careful version of joint FSQ training, or a stronger decoder-free discrete anti-collapse signal would close this gap is open work. The reconstruction anchor is also domain-specific: it is appropriate here because the rendered observations are predictable projections of a discrete game state, but the same objective may be wasteful or counterproductive in real-world video where much of the pixel information is not predictable from the agent's context. We also do not claim that the FSQ tokenizer used here is optimal. Recent GQ-style tokenizers improve image-tokenizer rate-distortion benchmarks, but swapping tokenizer families is outside the scope of this paper because it would change the token metric, world model training distribution, controller inputs, and deployment latency profile.

7 Conclusion

We presented DashVMC, a real-time discrete-WM system for Geometry Dash: an FSQ tokenizer, a block-causal transformer with action tokens interleaved in the sequence and a TWISTER-style action-conditional contrastive auxiliary loss, and a CNN actor-critic on the current token grid trained via BC and PPO entirely in latent rollouts. The same action-conditioned dynamics can be used generatively: from a real gameplay prefix, the model rolls out plausible future token grids and decoded frames, providing procedural level-continuation samples in addition to control-oriented latent rollouts. The three components are trained sequentially, and the full pipeline is deployed at 30 FPS on consumer hardware, with measured mean end-to-end latency of 16.3 ms. Across 100 controlled attempts per level, the frozen policy averages 9.32 s and 8.78 s of survival on Levels 1 and 2 and 2.14 s on the harder Level 3 with its additional yellow-orb timing mechanic. A set of architectural variants (joint FSQ+M training, AdaLN-Zero action conditioning, scaling to a larger transformer, an MLP controller on the hidden state) is reported as ablations: each is more elegant on paper but did

not improve real-game level progress in our experiments, motivating the deliberately conservative baseline of this paper. Structured Label Smoothing is reported as a scoped Geometry Dash design choice and auxiliary diagnostic: it is motivated by FSQ-neighbour equivalence in the application, but the IRIS/Pong transfer test does not support a broad method claim. A natural extension is to make the tokenizer-neighbour objective adaptive, but this would require a controlled validation target and safeguards against degenerate soft targets.

Acknowledgments and Disclosure of Funding

We thank Maël Planchot for contributing gameplay data. The NVIDIA A100 GPU used for all primary training runs was provided by MesoNet (CRIANN, Rouen) through the Representation Learning course at INSA Rouen Normandy. Geometry Dash is developed by RobTop Games.

References

[1] David Ha and Jürgen Schmidhuber. World models. In *Neural Information Processing Systems*, 2018.

[2] Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *International Conference on Learning Representations*, 2023.

[3] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 2025.

[4] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: VQ-VAE made simple. In *International Conference on Learning Representations*, 2024.

[5] Hugues Van Assel, Mark Ibrahim, Tommaso Biancalani, Aviv Regev, and Randall Balestriero. Joint embedding vs reconstruction: Provable benefits of latent space prediction for self supervised learning. *arXiv preprint arXiv:2505.12477*, 2025.

[6] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[7] Maxime Burchi and Radu Timofte. TWISTER: Learning transformer-based world models with contrastive predictive coding. In *International Conference on Learning Representations*, 2025.

[8] Eloi Alonso, Adam Jelley, Vincent Micheli, Anssi Kanervisto, Amos Storkey, Tim Pearce, and François Fleuret. Diffusion for world modeling: Visual details matter in atari. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[9] Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025.

[10] Emmanuel Dupoux, Yann LeCun, and Jitendra Malik. Why AI systems don't learn and what to do about it: Lessons on autonomous learning from cognitive science. *arXiv preprint arXiv:2603.15381*, 2026.

[11] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[12] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Deep q-learning from demonstrations. In *AAAI Conference on Artificial Intelligence*, 2018.

[13] Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (VPT): Learning to act by watching unlabeled online videos. *arXiv preprint arXiv:2206.11795*, 2022.

[14] Mohammad Hassan Vali, Tom Bäckström, and Arno Solin. iFSQ: Improving FSQ for image generation with 1 line of code. In *International Conference on Learning Representations*, 2026.

[15] Tongda Xu, Wendi Zheng, Jiajun He, José Miguel Hernández-Lobato, Yan Wang, Ya-Qin Zhang, and Jie Tang. Training-free vector quantization via gaussian VAEs. *arXiv preprint arXiv:2512.06609*, 2025.

[16] Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorld-Model: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

[17] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception architecture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[18] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, 2018.

[19] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *International Conference on Computer Vision*, 2023.

[20] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorber, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning*, 2024.

[21] Zaishuo Xia, Yukuan Lu, Xinyi Li, Yifan Xu, and Yubei Chen. Clone deterministic 3d worlds with geometrically-regularized world models. *arXiv preprint arXiv:2510.26782*, 2025.

[22] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *International Conference on Computer Vision*, pages 2980–2988, 2017.

[23] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Paper Checklist

1. **Claims**

   Answer: [Yes]

   Justification: The abstract and introduction state the main contribution as DashVMC, a real-time discrete world model control system for Geometry Dash, with SLS reported as a scoped design choice and diagnostic study.

2. **Limitations**

   Answer: [Yes]

   Justification: Section 6 discusses the auxiliary IRIS/Pong SLS diagnostic, the retired Atari controller path, Geometry Dash's single-environment scope, binary action space, and tokenizer-metric transferability.

3. **Theory assumptions and proofs**

   Answer: [N/A]

   Justification: The paper does not include theoretical results. SLS is an empirical method.

4. **Experimental result reproducibility**

   Answer: [Yes]

   Justification: Section 4 provides the data split, full training details, and controller hyperparameters. Code and checkpoints are available with the submission artifacts.

5. **Open access to data and code**

   Answer: [Yes]

   Justification: Code and dataset are publicly available at `https://github.com/Tariolle/dash-vmc`. Data collection procedure is described in Section 4.

6. **Experimental setting/details**

   Answer: [Yes]

   Justification: All hyperparameters, optimizers, schedules, and data augmentation are specified in Section 4.

7. **Experiment statistical significance**

   Answer: [No]

   Justification: The draft reports 100 controlled live attempts on each of three levels with bootstrap confidence intervals and an auxiliary five-seed IRIS/Pong diagnostic, but the Geometry Dash models themselves are frozen single-seed systems.

8. **Experiments compute resources**

   Answer: [Yes]

   Justification: All experiments run on a single NVIDIA A100 GPU. Training times are reported in Section 4.

9. **Code of ethics**

   Answer: [Yes]

   Justification: The research involves no human subjects, sensitive data, or dual-use concerns.

10. **Broader impacts**

    Answer: [N/A]

    Justification: This work applies to a single-player video game with no direct societal impact.

11. **Safeguards**

    Answer: [N/A]

    Justification: The released model is a small world model for a video game and poses no misuse risk.

12. **Licenses for existing assets**

Answer: [Yes]

Justification: All referenced works are properly cited. Geometry Dash is a commercial game by RobTop Games; gameplay data was collected from a legally purchased copy.

13. **New assets**

    Answer: [Yes]

    Justification: Code and model checkpoints are released under an open-source license with documentation.

14. **Crowdsourcing and research with human subjects**

    Answer: [N/A]

    Justification: No crowdsourcing or human subjects research was conducted.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Answer: [N/A]

    Justification: No human subjects research was conducted.

16. **Declaration of LLM usage**

    Answer: [N/A]

    Justification: LLMs were not used as a component of the core methodology.