Not Your Average Agent: Play-style Conditioned Decision Transformers

Sara Karimi[1, 2], **Sahar Asadi**[1], **Amir H. Payberah**[2]

[1]King AI Labs - Microsoft Gaming
[2]KTH Royal Institute of Technology
sara.karimi@king.com, sahar.asadi@king.com, payberah@kth.se

Abstract

Automated playtesting agents can help game designers evaluate balance, difficulty, and player experience by simulating diverse player behaviors. However, many existing reinforcement learning approaches collapse these behaviors into a single policy, limiting their ability to expose how different play styles interact with game mechanics and level design. In this work, we introduce STYLEDT, a style-conditioned Decision Transformer designed to generate controllable, diverse gameplay behaviors. STYLEDT models player strategies as a continuous latent style representation learned from full gameplay trajectories. A conditional prior maps designer-specified behavioral controls to distributions over latent styles, which are injected as prompt tokens into a Decision Transformer policy. This architecture enables inference-time control over agent behavior without requiring reference trajectories. We evaluate STYLEDT in two MiniGrid-based playtesting environments of increasing complexity featuring multiple viable strategies, including combat, stealth, adventure, and avoidance behaviors. Experimental results show that STYLEDT accurately reproduces different strategic behaviors and achieves strong style adherence, matching or exceeding specialized single-style policies while operating as a single unified model. Furthermore, the learned latent style representation enables reliable behavioral steering and generalizes to interpolated control configurations not present during training.

Introduction

Reinforcement Learning (RL) has become an increasingly attractive tool to build gameplay agents in modern game development, driven by advances in offline learning, large-scale behavioral datasets, and sequence modeling architectures (Xu et al. 2024). While recent research has demonstrated that RL-based agents can achieve strong performance in a wide range of games (Berner et al. 2019; Vinyals et al. 2019; Badia et al. 2020), their adoption in real-world production pipelines, particularly for game design and playtesting, remains limited by two persistent obstacles: (i) inability to capture diverse player styles and (ii) lack of behavioral controllability.

In practice, game level designers rely heavily on playtesting agents to evaluate balance, difficulty, and player experience under frequent content changes (Arifin, Mario, and Levina 2025; Jeon et al. 2023). Most Machine Learning based gameplay agents are trained to optimize a single objective or to mimic an average behavior across the dataset, resulting in a generic agent that represents no particular player group. This one-size-fits-all paradigm fails to capture the rich diversity of player behaviors observed in real games, such as risk-taking versus cautious play, aggressive versus stealthy strategies, or resource-driven versus avoidance-based decision-making. As a result, designers lack the ability to systematically probe how different play-styles interact with level layouts, item placements, and item distributions.

A natural solution is to build *style-aware* gameplay agents that can be steered to exhibit distinct behaviors at inference time. Prior work has explored style discovery and diversity in offline RL (Ingram et al. 2023; Sudhakaran and Risi 2023). However, these approaches typically yield a fixed set of discrete policies or skills, requiring either the explicit selection of a policy index or the use of reference trajectories at deployment. In playtesting settings, these requirements are often impractical, since designers may not have access to example trajectories and instead prefer to specify high-level behavioral preferences directly.

Recent advances in offline RL increasingly adopt transformer-based sequence models, framing RL as a conditional sequence modeling problem. Decision Transformer (DT) (Chen et al. 2021) learns policies from offline data by modeling trajectories autoregressively, conditioning actions on past states and desired returns. Extensions such as Prompting DT (Xu et al. 2022) and Skill DT (Sudhakaran and Risi 2023) improve adaptability by conditioning on task-specific trajectory prompts or discovered skills. While these methods enhance generalization and enable task-level control, they do not address the problem of *controllable, style-aware playtesting* driven by designer-specified preferences, nor do they provide a mechanism to generate such styles without reference trajectories at inference time.

In this work, we present STYLEDT, a controllable playtesting agent that enables designers to specify and sample diverse play-styles using a set of intuitive control configurations. STYLEDT combines three key components: (i) a full-trajectory encoder that learns a compact latent representation of long-horizon player strategies from offline gameplay data, (ii) a DT decoder conditioned on this latent as soft prompt tokens, and (iii) a learned conditional prior that maps

designer-provided style controls such as risk tolerance and resource-seeking behavior to distributions over latent styles. This design enables coherent, style-consistent behavior at inference time *without requiring demonstration trajectories*, while still preserving diversity through latent sampling.

We evaluate STYLEDT in two MiniGrid-based playtesting environments featuring multiple viable strategies to reach a goal, including combat, stealth, adventure, and avoidance routes. Our experiments demonstrate that STYLEDT enables fine-grained control over agent behavior, produces diverse yet consistent play-styles under identical control settings, and generalizes to interpolated style configurations not explicitly seen during training. By comparing against DT-based baselines, we show that both the full-trajectory style encoder and the conditional prior are essential for achieving controllable, realistic playtesting behavior.

Contributions: Our main contributions are as follows:

- STYLEDT: We introduce a style-conditioned DT architecture that models play-styles via a latent representation learned from full trajectories, enabling controllable and diverse gameplay behavior within a single unified model.
- *Conditional Prior*: We present a conditional prior over style latents that enables inference-time control using designer-specified behavioral controls without reference trajectories or discrete policy selection.
- *Empirical Evaluation*: Through experiments in two custom MiniGrid environments, we demonstrate style controllability and generalization to interpolated control configurations.

Related Work

This section reviews prior work most relevant to STYLEDT and highlights how our approach differs from these lines of research, particularly regarding inference-time controllability and applicability to playtesting.

Play-style Identification and Imitation

Identifying play styles from player data is a longstanding problem in game analytics and AI-assisted game design. One influential perspective is persona-based player modeling, which defines agents through designer-relevant behavioral criteria rather than solely cloning observed player data (Holmgård et al. 2014). Early work has also focused on clustering trajectories or player features to discover recurring behavioral patterns, which can inform game balancing and content iteration. Such analyses are often performed without access to explicit game variables, relying instead on temporal similarity measures or trajectory alignment methods, such as Dynamic Time Warping, to group player behaviors into distinct styles (Ferguson et al. 2020, 2022).

Beyond analysis, several methods leverage discovered styles to train gameplay agents that imitate specific player groups. For example, play-style-centric policy generation methods train multiple behavioral cloning agents, each specializing in a particular style cluster (Ingram et al. 2023). While effective at reproducing representative behaviors, these approaches typically result in a fixed set of discrete agents and require explicit policy selection at inference time.

On the topic of human-like play-style generation, de Woillemont et al. (De Woillemont, Labory, and Corruble 2022) propose CARMI, a configurable RL agent conditioned on relative summary metrics that define play-styles with respect to the distribution of human players. Their approach trains a single agent to generate a continuum of play-styles by providing target play-mode values as inputs and shaping the reward to match these objectives, enabling generalization to previously unseen levels while relying only on summarized human data rather than full trajectories. While this work demonstrates the effectiveness of metric-driven conditioning for automated testing, it relies on reward-based optimization and careful specification of target metrics.

SORL (Mao et al. 2024) extends this paradigm by combining trajectory clustering with offline policy optimization. SORL alternates between assigning trajectories to latent style indices and learning one policy per style using advantage-weighted updates, yielding multiple high-quality policies from heterogeneous datasets. This approach is well-suited for discovering and extracting diverse behaviors when styles are unknown a priori, but, like other multi-policy approaches, it does not support inference-time interpolation or control (Mao et al. 2024).

Recent work has also explored learning controllable and diverse player behaviors in multi-agent environments (Cilan and Özgövde 2025). These approaches demonstrate that explicit mechanisms for behavioral control and diversity can be learned jointly within a single model. However, they typically rely on reward shaping or environment-specific training procedures, and do not provide a mechanism for inference-time control.

While prior style-centric methods focus on discovering or training separate policies for each style, STYLEDT addresses a complementary problem: enabling *controllable generation* of play styles at inference time. Instead of producing a discrete set of policies or requiring reference trajectories, our method learns a continuous latent style space and a conditional prior that maps designer-specified behavioral preferences to distributions over style latents. This allows a single agent to generate diverse, coherent behaviors without explicit policy selection.

Conditioned Offline RL and Transformer Policies

Recent advances in offline RL increasingly adopt transformer-based architectures, framing decision-making as a conditional sequence modeling problem. DT models trajectories autoregressively and conditions action prediction on past states and desired returns, enabling effective policy learning from offline data (Chen et al. 2021). Several extensions of DT introduce conditioning mechanisms to improve adaptability and generalization. For example, PDT (Xu et al. 2022) prepends prompt embeddings to guide behavior in multi-task and few-shot settings, while Skill-DT (Sudhakaran and Risi 2023) discovers discrete latent skills and conditions action generation on these skill embeddings and predicted skill transitions. These methods demonstrate that transformer decoders can effectively

leverage both hard and soft conditioning signals to modulate behavior.

STYLEDT builds on these conditioning mechanisms by introducing a probabilistic latent style representation inferred from full trajectories and injected into a DT as soft prompt tokens. Unlike prior conditioned DT variants, which rely on discrete skills or explicit prompts provided at inference time, STYLEDT learns a conditional prior over style latents that enables control using continuous, interpretable designer inputs. This design allows STYLEDT to generate diverse play styles consistent with high-level preferences while maintaining behavioral coherence.

In contrast to prior work on style discovery, discrete skill learning, or multi-policy generation, STYLEDT focuses on inference-time controllability for playtesting. By combining full-trajectory style encoding, a soft-prompted DT decoder, and a conditional prior driven by designer controls, our method enables controllable, diverse, and coherent gameplay behavior within a single unified model.

Preliminaries

This section briefly reviews some basic concepts from the DT and the Prompting DT (PromptDT).

Decision Transformer

A DT (Chen et al. 2021) is a sequence model-based approach to RL that treats trajectory optimization as a supervised sequence prediction task. Instead of estimating value functions or learning an explicit policy, DT conditions action selection on desired return-to-go. In a Markov Decision Process (MDP), a trajectory is defined as a sequence of tuples $\tau = (s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_T, a_T, r_T)$, where $s_t \in \mathcal{S}$ is the state at timestep t, $a_t \in \mathcal{A}$ is the action taken, $r_t \in R$ is the reward received, and T is the trajectory length. Instead of modeling a standard policy $\pi(a_t|s_t)$, DT conditions action selection on past states, actions, and future return-to-go $R_t = \sum_{t'=t}^{T} r_{t'}$. DT models the probability of the next action using a causal transformer $\pi(a_t|s_{\leq t}, a_{<t}, R_{\leq t})$, where inputs are tokenized as sequences $(R_1, s_1, a_1, R_2, s_2, a_2, \ldots)$. This allows the model to generate actions autoregressively by conditioning on past trajectory data and future expected return, enabling goal-directed behavior in an offline RL setting.

Prompting Decision Transformer

PromptDT (Xu et al. 2022) extends DT to a multi-task setting by incorporating task-specific demonstrations as additional inputs that guide the agent's behavior. These contextual inputs are commonly referred to as *prompts*, which serve as high-level behavioral guides, enabling PromptDT to control generalization to new tasks and improving flexibility compared to standard DT. PromptDT modifies the transformer input sequence as $(\rho, R_1, s_1, a_1, R_2, s_2, a_2, \ldots)$, where ρ is composed of task-specific prompts of form $(R_1^*, s_1^*, a_1^*, R_2^*, s_2^*, a_2^*, \ldots)$. PromptDT randomly samples trajectory segments from a set of expert demonstration trajectories $\mathcal{P}_i$ that constitute the prompt ρ for each training task $\mathcal{T}_i \in \mathcal{T}^{train}$.

Method

We present STYLEDT, a controllable playtesting agent that generates style-consistent gameplay behavior conditioned on designer-specified controls. STYLEDT combines (i) a full-trajectory style encoder, (ii) a conditional latent prior, and (iii) a DT decoder with soft prompt conditioning. An overview of the architecture is shown in Figure 1.

Problem Setting

We consider an offline dataset $\mathcal{D} = \{(\tau_i, c_i)\}_{i=1}^{N}$, where each trajectory $\tau = (s_1, a_1, \ldots, s_T, a_T)$ is generated by a particular play-style, and $c \in R^d$ is a vector of domain-relevant behavioral controls that can provide a high-level description of different styles. Examples of such control variables include risk tolerance, stealth preference, and resource collection preference. The controls c are provided during training and are assumed to be available at inference time, reflecting the designer's intent.

We aim to learn policy $\pi(a_t \mid s_{\leq t}, a_{<t}, R_{\leq t}, z)$ that: (i) produces coherent behavior consistent with specific styles represented by the latent variable z, (ii) allows inference-time control via c without requiring reference trajectories as input to the model, and (iii) supports diversity by sampling multiple behaviors consistent with the same controls. To achieve this, STYLEDT factorizes behavior generation into three components:

1. *Style encoder* $q_\phi(z \mid \tau)$ that maps a full trajectory to a latent style representation.
2. *Conditional prior* $p_\psi(z \mid c)$ that maps specified controls to a distribution over the style latents.
3. *DT decoder* $p_\theta(a_t \mid s_{\leq t}, a_{<t}, R_{\leq t}, z)$ that generates actions conditioned on the style latent.

In the rest of this section, we explain these components.

Style Encoder

The style encoder compresses an entire trajectory of states and actions into a latent variable that captures strategic behaviors, referred to as *styles* (part B in Figure 1). Given a full trajectory τ, we first embed states and actions using learned embedding layers and arrange them as a sequence of tokens augmented with positional (timestep) embeddings. This sequence is then processed by a Transformer encoder, producing token representations that capture temporal dependencies across the trajectory.

Since trajectories may vary in length, we apply mean pooling to obtain a fixed-length representation. This representation is passed through linear projection layers that output the parameters of a latent distribution. In our implementation, we use a Gaussian posterior:

$$q_\phi(z \mid \tau) = \mathcal{N}(\mu_\phi(\tau), diag(\sigma_\phi^2(\tau))),$$

where q_ϕ denotes the encoder model, μ_ϕ and σ_ϕ^2 are the mean and variance of the Gaussian $\mathcal{N}$. The latent variable z is sampled using the "reparameterization trick", $z = \mu_\phi(\tau) + \sigma_\phi(\tau) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$. Encoding full trajectories ensures that z captures global strategy choices such as route commitment or risk management that are not recoverable from a short trajectory context window.

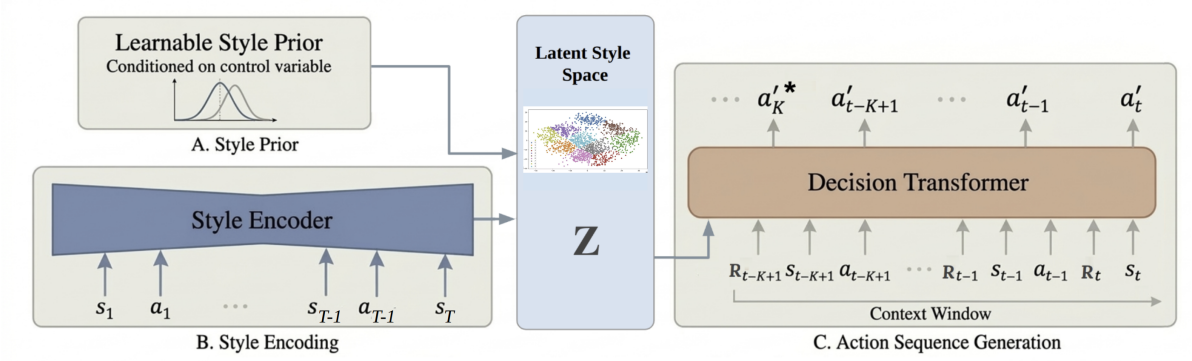

Figure 1: Overview of StyleDT. A trajectory encoder maps full gameplay trajectories to a latent style variable z. A conditional prior learns to map control variables to a distribution over styles. The sampled style embedding is injected as a prompt into a DT, enabling the generation of style-consistent behavior.

Conditional Prior over Styles

To enable inference-time control without requiring demonstration trajectories as input, we learn a conditional prior that maps designer-specified behavioral controls to a distribution over latent styles. Let $c \in R^d$ denote a vector of high-level control variables describing the desired play-style (e.g., risk tolerance, stealth preference, or resource-seeking behavior). The conditional prior is defined as:

$$p_\psi(z \mid c) = \mathcal{N}(\mu_\psi(c), diag(\sigma^2_\psi(c))),$$

where $\mu_\psi(c)$ and $\sigma^2_\psi(c)$ are produced by a lightweight multilayer perceptron (MLP) parameterized by ψ. The MLP takes the control vector c as input and outputs the mean and log-variance parameters of the latent Gaussian distribution.

During training, the conditional prior is aligned with the encoder posterior $q_\phi(z \mid \tau)$ by minimizing the Kullback–Leibler (KL) divergence:

$$D_{\mathrm{KL}}\big(q_\phi(z \mid \tau) \,||\, p_\psi(z \mid c)\big),$$

encouraging the prior to produce latent style representations consistent with those inferred from trajectories. This alignment allows the model to learn a mapping between interpretable control variables and regions of the latent style space.

At inference time, designers specify a control vector c to the learned prior $p_\psi(z \mid c)$ from which a style latent z is sampled. This sampled latent variable is then used to condition the policy, thereby enabling direct control over the agent's behavior without requiring reference trajectories.

DT Decoder with Style Prompts

We adopt the DT as the policy decoder. Following the DT formulation, trajectories are modeled as sequences of tokens $(R_1, s_1, a_1, R_2, s_2, a_2, \ldots)$, where R_t denotes the return-to-go at timestep t. The model predicts actions by conditioning on past states, actions, and desired returns. To incorporate play-style information, we condition the Transformer on the latent style variable z. Specifically, we project the latent style embedding into the token embedding space and prepend it to the input sequence as a set of *style prompt tokens*. These tokens act as global conditioning signals that influence the action generation. The resulting decoder defines a style-conditioned policy as follows:

$$p_\theta(a_t \mid s_{\le t}, a_{<t}, R_{\le t}, z).$$

Injecting the style embedding as prompts to the model provides the Transformer with style information without modifying the core DT architecture. This mechanism encourages the policy to generate behavior that remains consistent with the specified play-style over long horizons.

Training Objective

StyleDT is trained using an offline objective that combines behavior cloning and latent regularization. For a trajectory τ with controls c, we minimize:

$$\mathcal{L} = E_{z\sim q_\phi(z|\tau)}\left[\sum_t \mathcal{L}_{BC}(a_t, \hat{a}_t)\right] + \beta\,\mathrm{KL}\big(q_\phi(z \mid \tau) \,\|\, p_\psi(z \mid c)\big),$$

where $\mathcal{L}_{BC}$ is a supervised imitation loss (cross-entropy for discrete actions or mean squared error for continuous actions), and β controls the strength of the KL regularization.

We train the decoder using fixed-length context windows of size K sampled from trajectories, while the encoder always observes the full episode. This mirrors inference-time usage, where the policy operates with limited history but is conditioned on a global style latent.

Inference and Playtesting

At inference time, designers specify a control vector c that reflects the desired behavioral preferences. A style latent is sampled from the learned prior $p_\psi(z \mid c)$, and used as style prompt tokens to condition the DT during rollout. Sampling multiple latents for the same c yields diverse yet style-consistent playthroughs, enabling robust playtesting under controlled behavioral variations. This sampling mechanism also enables exploration of interpolated control configurations; for example, blending high-risk and high-resource-seeking preferences to generate more stochastic yet style-consistent behaviors.

Experiments

We evaluate STYLEDT in controlled experiments measuring task performance, control fidelity, and generalization. We describe the environment, offline dataset, implementation, baselines, and evaluation metrics, then present quantitative and qualitative analyses of style consistency, success rate, and behavior under interpolated and out-of-distribution controls. Code, hyperparameters, and reproduction instructions are available at: https://github.com/sarakarimi/StyleDT.

Environments and Datasets

We evaluate STYLEDT in custom MiniGrid-based environments (Chevalier-Boisvert et al. 2023) (Figure 2) designed to capture multiple viable play strategies. Two environments are presented: a simpler, smaller map and a larger, more complex map. The environments consist of a grid world containing an agent, a goal location, an enemy, and some optional items: a weapon, a camouflage, and an iron boot. At each environment reset, the collectible items are randomly placed on a different tile within a predefined region of the map. Additionally, in the smaller map, the agent and goal locations are perturbed slightly between resets to introduce further variability in the environment. The agent's objective is to reach the goal while avoiding death. The enemy has a fixed position and a region of sight, and the agent is terminated if it enters that region while the enemy is alive. The weapon can be collected to eliminate the enemy in combat, enabling safe traversal through its region-of-sight, while the camouflage allows the agent to remain undetected even when entering those visible tiles. In the hard map, other possibilities are teleporting using the portals or using the iron boots to enable safe traversal through the risky lava river at the bottom of the map. Moreover, the environment includes alternative routes that allow the agent to bypass the risky scenarios at the cost of longer paths to the goal. The episode is terminated upon entering the lava without boots, entering the enemy region of sight without elimination, or upon timeout (100 steps).

The offline dataset consists of gameplay trajectories generated using pre-trained online RL models under distinct strategic behaviors: (i) *weapon* strategy, where the agent first collects the sword item and uses it to eliminate the enemy, and then proceeds through the enemy's region of sight to reach the goal, (ii) *camouflage* strategy, where the agent collects the camouflage item, allowing it to traverse the enemy's region of sight without being detected, (iii) *lava* strategy (only present in the hard map) where the agent collects a boot to enable safe traversal through the lava tunnel, (iv) *portal* strategy (only present in the hard map) where the agent uses the portals to teleport across the map, and (v) *bypass* strategy (only present in the Easy map), where the agent avoids interacting with all items and instead follows an alternative route that circumvents the enemy's region of sight entirely, typically resulting in a longer but safer path to the goal. Each trajectory records the sequence of states, actions, and termination signals, and is labeled with a high-level behavioral control vector provided by the environment.

We define $\mathbf{c}$ as a behavioral control vector whose dimensions reflect the mechanics available in each environment.

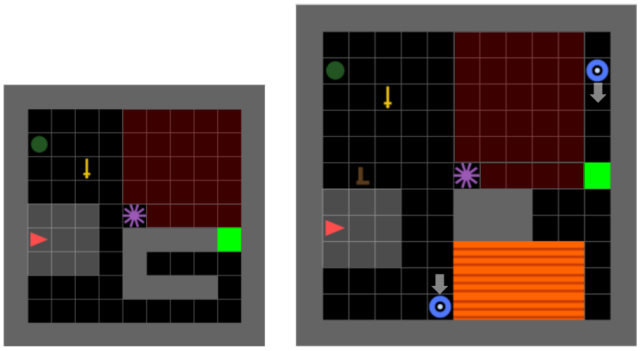

Figure 2: **Two MiniGrid environments simulating simple stealth games**. The red triangle represents the player, who has a fixed field of view. The purple star denotes the enemy, and the green square marks the goal. The simpler variant introduces a collectible weapon marked by a yellow cross that allows the player to eliminate the enemy, and a camouflage item marked by a green circle that helps the player avoid detection. The more complex variant additionally introduces the brown boots, which help the agent traverse the lava field at the bottom, and portals that move the agent from one side of the map to the other. The highlighted red area indicates the enemy's region of sight.

Both environments include (i) *risk_tolerance*, measuring the agent's willingness to approach enemies (normalized minimum enemy distance), and (ii) *commitment*, measuring how efficiently the agent progresses toward the goal instead of turning or backtracking (ratio of forward movement steps). The third dimension is environment-specific: in the Easy environment, *resource_preference* measures the tendency to collect collectible items like weapon or camouflage, while in the Hard environment, *stealth_preference* measures the tendency to avoid the enemy by using camouflage or crossing the lava tunnel instead of confronting it.

Baselines

We compare STYLEDT with four baseline approaches, each representing alternative strategies for incorporating style information into offline policy learning.

Prompting Decision Transformer (PromptDT). As a first baseline, we implement PromptDT (Xu et al. 2022) adapted to the play-style setting by using short trajectory segments corresponding to the target strategy as prompts. During training, prompts are sampled from trajectories of the same style as the rollout window, while at inference, a representative trajectory segment is provided. This baseline evaluates whether direct trajectory-based prompting is sufficient to recover coherent style-conditioned behavior.

Control-Prompt Decision Transformer (ControlDT). Our second baseline replaces trajectory prompts with the high-level control vector c. Instead of prepending demonstration tokens, we prepend an embedding of the control vector to the DT input sequence as a soft prompt. ControlDT

provides the model with explicit behavioral descriptors, but does not learn a latent style representation or a conditional prior. In contrast to StyleDT, ControlDT must directly map control variables to low-level actions without an intermediate trajectory-level encoding. ControlDT serves as a combined ablation of the trajectory encoder and conditional prior, testing whether direct control-to-action conditioning can replace the proposed latent-style mechanism.

Behavioral Cloning (BC) per Style. As a non-Transformer baseline, we train independent BC policies, one for each style. Each model is trained using supervised imitation learning on trajectories corresponding to a single strategy. While this approach avoids the need for conditioning mechanisms, it requires training and maintaining multiple separate policies and does not support interpolation between styles. This baseline evaluates whether StyleDT can match or exceed specialized single-style agents while maintaining controllability in a unified model.

Stylized Offline Reinforcement Learning (SORL). We include SORL (Mao et al. 2024) as an additional baseline. SORL learns multiple policies corresponding to different behavioral styles by clustering trajectories and optimizing one policy per cluster. At inference time, a specific style is selected by choosing the corresponding policy. This baseline evaluates whether explicitly learning separate style-specific policies can match the controllability and performance of a unified style-conditioned model.

Evaluation Metrics

We evaluate the methods across three complementary dimensions: *online rollout performance*, *behavioral control fidelity*, and *latent space quality*.

Rollout Metrics. For each method and target style $s \in \{portal, weapon, camouflage, lava, bypass\}$, we execute rollouts in the corresponding environments with randomized layout and report the following per-style statistics, aggregated over episodes:

- *Success Rate (SR)*: the fraction of episodes in which the agent reaches the goal before the time limit.
- *Style Achievement Rate (SAR)*: proportion of episodes in which the agent reaches the goal and the realized behavioral style matches the intended target style s, as determined by the environment's episode summary.
- *Weapon/Camouflage Usage Rate (WUR and CUR)*: the fraction of successful episodes in which the weapon or camouflage item was collected.

Control Fidelity. Control fidelity measures how faithfully each control dimension drives its intended behavioral outcome. Specifically, we report the mean signed Spearman rank correlation $\bar{\rho} = \frac{1}{K}\sum_k \rho_k$ as a summary of control fidelity, where ρ_k is the Spearman correlation (Spearman 1961) formulated as $\rho_k = 1 - \frac{6\sum_{i=1}^{N} d_i^2}{N(N^2-1)}$, and $d_i = rank(c_k^{(i)}) - rank(b_k^{(i)})$ measuring the correlation between the control values $c_k^{(i)}$ and the corresponding behavioral measurements $b_k^{(i)}$ over the N samples.

Generalization to Interpolated Controls. To evaluate generalization, we test the model on interpolated control vectors (e.g., blending weapon and camouflage preferences) that often lie in regions of the control space not observed during training. We study generalization by analyzing shifts in the distribution of generated behaviors when the model is conditioned on unseen control configurations.

Results & Analysis

Model Performance

Tables 1 and 2 report rollout metrics for different target strategies on each of the environments and across all evaluated models. Overall, StyleDT achieves performance comparable to or exceeding specialized baselines while maintaining strong style adherence, demonstrating that a single style-conditioned model can reproduce diverse behaviors without requiring separate policies.

Bypass Strategy. In the bypass setting (Table 1), all models achieve near-perfect success rates, indicating that the task itself is relatively easy once the correct route is chosen. However, differences emerge in *style adherence*. StyleDT achieves a high style achievement rate (SAR=0.96), closely approaching the oracle BC baseline (SAR=1.0), while achieving comparable or higher style adherence than PromptDT (0.94) and ControlDT (0.93). PromptDT exhibits occasional style leakage, as evidenced by a higher weapon usage rate (0.06) despite the bypass strategy ideally avoiding item interactions. In contrast, StyleDT maintains near-zero weapon usage. These results indicate that StyleDT captures the long-horizon avoidance behavior characteristic of bypass strategies, rather than simply reaching the goal by any means.

Weapon Strategy. The weapon-focused strategy (Table 1) requires the agent to collect the sword and eliminate the enemy before proceeding. In this setting, StyleDT achieves strong performance with high style fidelity and resource usage (WUR=0.98), approaching the oracle BC baseline (1.0) and comparable to or exceeding PromptDT and ControlDT. While BC achieves slightly cleaner behavior due to its single-style specialization, StyleDT maintains comparable success rates and style adherence while operating as a unified multi-style model. PromptDT shows weaker style consistency, reflected in lower SAR and higher camouflage usage, suggesting that short prompt segments may not always provide sufficient context to enforce a specific strategic pattern.

ControlDT performs moderately well but shows less consistent behavior, with slightly lower style adherence. This indicates that directly conditioning the policy on control variables without a structured latent representation makes it

Metric	StyleDT	BC	PromptDT	ControlDT	SORL
SR Bypass	**1.00±0.00**	**1.00±0.00**	0.99±0.01	**1.00±0.00**	**1.00±0.00**
SAR Bypass	0.96±0.08	**1.00±0.00**	0.93±0.02	0.93±0.08	0.96±0.08
WUR Bypass	0.01±0.02	0.00±0.00	0.06±0.02	0.00±0.00	0.00±0.00
CUR Bypass	0.04±0.08	0.00±0.00	0.00±0.00	0.06±0.08	0.04±0.08
SR Weapon	0.96±0.00	0.96±0.08	0.93±0.07	0.93±0.04	0.94±0.03
SAR Weapon	0.93±0.02	**0.96±0.08**	0.84±0.09	0.86±0.05	0.94±0.03
WUR Weapon	**0.98±0.02**	**1.00±0.00**	0.90±0.07	0.94±0.04	**1.00±0.00**
CUR Weapon	0.02±0.02	0.00±0.00	0.09±0.07	0.05±0.05	0.00±0.00
SR Camouflage	**1.00±0.00**	0.92±0.10	0.99±0.01	0.99±0.01	0.96±0.08
SAR Camouflage	**1.00±0.00**	0.92±0.10	0.98±0.01	0.77±0.05	0.96±0.08
WUR Camouflage	0.00±0.00	0.00±0.00	0.01±0.01	0.78±0.01	0.00±0.00
CUR Camouflage	**1.00±0.00**	**1.00±0.00**	0.99±0.01	0.78±0.05	**1.00±0.00**

Table 1: Per-style rollout metrics of Easy Env (mean ± std over 5 random seeds) — Bypass, Weapon and Camouflage.

Metric	StyleDT	BC	PromptDT	ControlDT	SORL
SR Portal	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**
SAR Portal	**1.00±0.00**	**1.00±0.00**	0.78±0.19	0.98±0.05	**1.00±0.00**
SR Weapon	0.81±0.09	0.80±0.13	**0.86±0.04**	0.29±0.08	0.81±0.04
SAR Weapon	**0.81±0.09**	0.80±0.13	0.79±0.09	0.28±0.08	0.77±0.04
SR Camouflage	**1.00±0.00**	**1.00±0.00**	**1.00±0.00**	0.97±0.03	**1.00±0.00**
SAR Camouflage	**1.00±0.00**	**1.00±0.00**	0.86±0.14	0.95±0.04	**1.00±0.00**
SR Lava	**0.97±0.02**	0.60±0.18	0.97±0.03	0.97±0.03	0.80±0.01
SAR Lava	**0.92±0.06**	0.60±0.18	0.67±0.08	0.72±0.04	0.80±0.01

Table 2: Per-style rollout metrics of Hard Env (mean ± std over 5 random seeds) — Portal, Weapon, Camouflage, Lava.

harder for the model to learn coherent long-horizon strategies.

The weapon strategy becomes more challenging in the hard environment (Table 2) due to the additional routing decisions available. Despite this, STYLEDT achieves better style adherence (SAR=0.81) than the baselines. Although PromptDT attains a slightly higher success rate, its lower SAR indicates less consistent execution of the intended strategy. In contrast, ControlDT suffers a substantial drop in success rate, highlighting the importance of the learned latent style representation.

Camouflage Strategy. The camouflage setting (Table 1) highlights the advantages of STYLEDT. The model achieves perfect success and style adherence (SR=1.0, SAR=1.0) while consistently using camouflage items (CUR=1.0). This performance matches the Oracle BC baseline while surpassing both PromptDT and ControlDT. Notably, ControlDT struggles significantly with this style, exhibiting a lower SAR (0.77) and incorrectly collecting weapons in a large fraction of episodes. This behavior suggests that the model fails to consistently associate control inputs with the intended strategic behavior. In contrast, STYLEDT maintains coherent stealth-oriented navigation.

In the Hard environment (Table 2), STYLEDT maintains perfect success and style adherence in the camouflage setting, matching the BC baseline and surpassing the baselines, which suggests that STYLEDT better preserves the intended stealth behavior under more complex navigation choices.

Portal Strategy. The portal strategy requires the agent to exploit environment-specific navigation mechanics rather than simply collecting resources. As shown in Table 2, STYLEDT achieves perfect success and style adherence (SR=1.00, SAR=1.00), matching BC while substantially outperforming PromptDT in style consistency (SAR=0.78).

Lava Strategy. The lava strategy requires a multi-stage behavior in which the agent first acquires the boots before traversing the lava shortcut. STYLEDT achieves a high success rate (0.97) together with strong style adherence (SAR=0.92), substantially outperforming all baselines in behavioral consistency.

Across all strategies, the results demonstrate four key findings. First, STYLEDT matches or approaches the performance of specialized single-style policies while retaining the flexibility of a unified model. Second, PromptDT provides some behavioral guidance but cannot enforce consistent strategies as it depends on the informativeness of the trajectory prompt. Third, SORL achieves competitive performance on some styles but relies on a separate policy for each behavior, preventing interpolation and unified inference-time control. The lower performance of SORL on some styles compared to STYLEDT can be attributed to the limitations of clustering-based multi-policy approaches, which rely on clear separability in the action distribution.

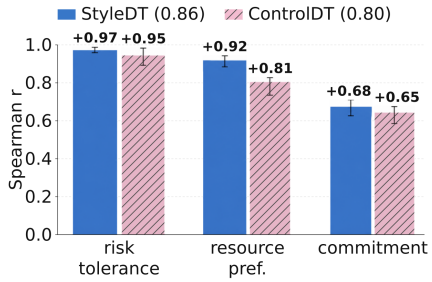

Figure 3: Control Fidelity on the Easy game based on Spearman correlation.

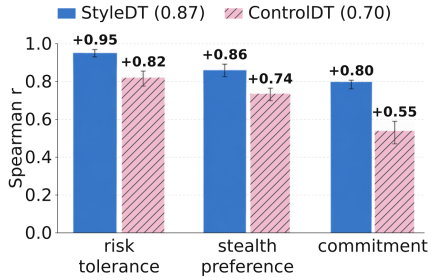

Figure 4: Control Fidelity on the Hard game based on Spearman correlation.

As a result, behaviors that differ only in a small number of decisions, are difficult to disentangle into distinct policies. Finally, ControlDT struggles to reliably translate discrete high-level behavioral controls into coherent policies, highlighting the importance of the learned latent style representation. Together, these results indicate that combining a trajectory encoder with a conditional latent prior enables StyleDT to capture the underlying structure of player strategies, generating style-consistent behavior without multiple specialized agents.

Control Fidelity

Figure 3 compares how faithfully StyleDT and ControlDT translate control variables into behavioral outcomes in the Easy environment. StyleDT consistently achieves higher Spearman correlations across all evaluated dimensions, indicating stronger alignment between the prompted controls and the generated behavior, suggesting that the learned latent style representation steers behavior more reliably than directly mapping control variables to actions. Figure 4 shows that the advantage persists and widens in the Hard environment. Although both methods lose some correlation as task complexity increases, the gap grows on every dimension (e.g., commitment: 0.80 vs. 0.55), suggesting that the latent representation matters more, not less, as the planning problem becomes more challenging.

Style Embeddings

Figure 5 shows the t-SNE projection of the learned Style embeddings. As seen from the plots, in both environments, the encoder learns a structured latent representation. Although the Hard environment exhibits greater intra-style variability, the styles remain well separated, indicating that the representation scales to more complex gameplay. Moreover, the spread within each cluster reflects the natural behavioral diversity present in the dataset, providing a latent space from which diverse yet style-consistent behaviors can be sampled.

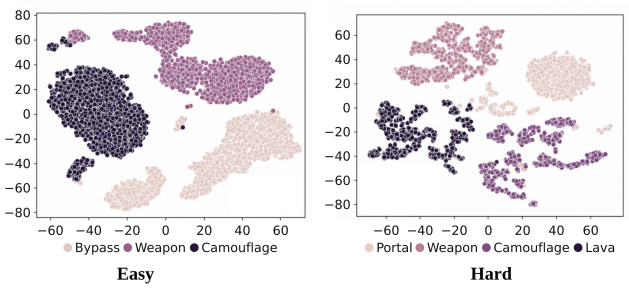

Figure 5: t-SNE projection of style embeddings learned from a random subset of offline dataset trajectories. Colors indicate ground-truth play styles.

Generalization Ability

To evaluate whether the learned style representations generalize beyond the control distribution observed during training, we designed an *interpolation* experiment in which agents are conditioned on control vectors that are linear blends of two canonical style prototypes rather than any single training vector. These blended vectors may not appear in the training dataset and therefore represent out-of-distribution inputs for the model. We constructed five interpolation sequences: bypass ↔ camouflage, weapon ↔ camouflage, and weapon ↔ bypass on the Easy environment and weapon ↔ portal and portal ↔ camouflage of the Hard one, each containing four steps ranging from pure style A (100%/0%) through two intermediate mixtures (60%/40% and 40%/60%) to pure style B (0%/100%). For each control vector, we executed rollouts across five random seeds. We report the distribution of achieved styles.

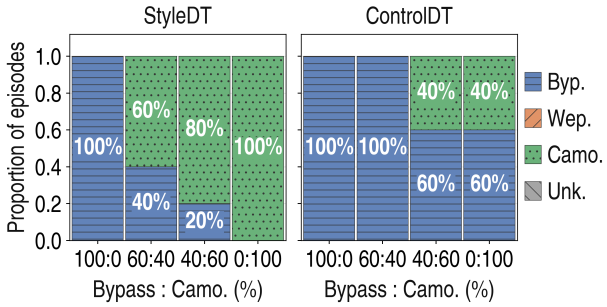

Figure 6: Style interpolation: **bypass ↔ camouflage.**

As shown in Figures 6, 7, 8, 9, and 10, StyleDT successfully captures this behavioral shift in a meaningful way across most interpolation sequences; except for the weapon↔ bypass sequence (Figure 8) which collapses to the camouflage style at intermediate blends due to the proximity of these styles in the dataset (also visible from Figure 5

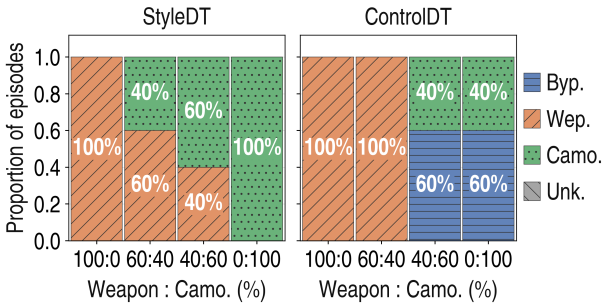

Figure 7: Style interpolation: **weapon** ↔ **camouflage.**

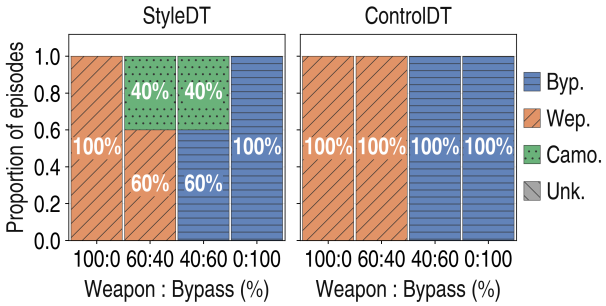

Figure 8: Style interpolation: **weapon** ↔ **bypass.**

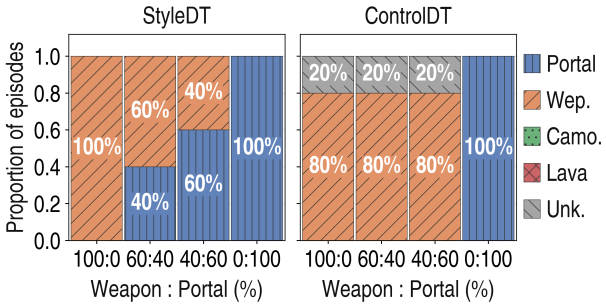

Figure 9: Style interpolation: **weapon** ↔ **portal.**

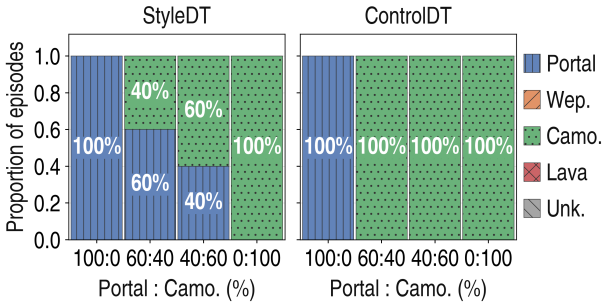

Figure 10: Style interpolation: **portal** ↔ **camouflage.**

Easy). ControlDT largely fails to respond to the interpolated inputs, collapsing to a dominant style regardless of the blend ratio. We attribute this disparity to the qualitative difference in how the two models represent style: ControlDT is trained to map a fixed set of discrete control vectors to corresponding behaviors and therefore has no mechanism to generalize to unseen regions of control space. STYLEDT, by contrast, learns a continuous latent space through its trajectory encoder, embedding styles as a structured manifold rather than a lookup table; interpolated control vectors therefore map naturally to coherent intermediate points on this manifold. These findings demonstrate that the encoder component is essential. It is the encoder's role in structuring the latent space during training that enables the model to generalize.

Conclusion

In this work, we introduce STYLEDT, a style-conditioned Decision Transformer (DT) for controllable, diverse gameplay behavior in automated playtesting. Unlike prior approaches that rely on discrete policies, demonstration prompts, or fixed behavior clusters, STYLEDT models play styles as a continuous latent representation learned from full trajectories. Combining a trajectory encoder, a conditional prior over style latents, and a prompt-conditioned DT decoder, it allows game designers to control agent behavior through interpretable high-level variables without reference trajectories at inference time. Experiments across environments of increasing complexity show that STYLEDT matches or, in some settings, exceeds the style adherence of specialized single-style policies while operating as a single unified model. Unlike multi-policy approaches such as SORL, STYLEDT supports interpolation across styles and remains reliable where overlapping styles defeat clustering-based extraction.

Compared to direct control conditioning, the latent representation steers behavior more reliably, with the advantage widening as task complexity grows. STYLEDT responds systematically to interpolated controls unseen in training, and the single failure case is traced to overlapping style prototypes in the latent space, suggesting that the learned latent style space captures meaningful structure in player strategies. Overall, these findings show that combining trajectory-level style encoding with conditional latent priors enables effective controllable behavior generation in offline RL, supporting scalable playtesting across diverse player behaviors.

Limitations and Future Work

The current evaluation is limited to relatively simple games. An important next step is to test STYLEDT in more complex games and using real human player data.

Acknowledgments

Generative AI tools were used to assist with figure styling and editing parts of the Introduction and Related Work sections. This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

Arifin, Y.; Mario, M.; and Levina, A. 2025. A Review of the Trends, Methods, and Impacts of Dynamic Game Balancing. *JOIV: International Journal on Informatics Visualization*, 9(6): 2467–2480.

Badia, A. P.; Piot, B.; Kapturowski, S.; Sprechmann, P.; Vitvitskyi, A.; Guo, Z. D.; and Blundell, C. 2020. Agent57: Outperforming the atari human benchmark. In *International conference on machine learning*, 507–517. PMLR.

Berner et al., C. 2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Chen, L.; Lu, K.; Rajeswaran, A.; Lee, K.; Grover, A.; Laskin, M.; Abbeel, P.; Srinivas, A.; and Mordatch, I. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34: 15084–15097.

Chevalier-Boisvert, M.; Dai, B.; Towers, M.; Perez-Vicente, R.; Willems, L.; Lahlou, S.; Pal, S.; Castro, P. S.; and Terry, J. K. 2023. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *Advances in Neural Information Processing Systems*, 36: 73383–73394.

Cilan, A.; and Özgövde, A. 2025. Learning Controllable and Diverse Player Behaviors in Multi-Agent Environments. *arXiv preprint arXiv:2512.10835*.

De Woillemont, P. L. P.; Labory, R.; and Corruble, V. 2022. Automated play-testing through RL based human-like playstyles generation. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 18, 146–154.

Ferguson, M.; Devlin, S.; Kudenko, D.; and Walker, J. A. 2020. Player style clustering without game variables. In *Proceedings of the 15th International Conference on the Foundations of Digital Games*, 1–4.

Ferguson, M.; Devlin, S.; Kudenko, D.; and Walker, J. A. 2022. Imitating playstyle with dynamic time warping imitation. In *Proceedings of the 17th International Conference on the Foundations of Digital Games*, 1–11.

Holmgård, C.; Liapis, A.; Togelius, J.; and Yannakakis, G. N. 2014. Personas versus clones for player decision modeling. In *International Conference on Entertainment Computing*, 159–166. Springer.

Ingram, B.; Van Alten, C.; Klein, R.; and Rosman, B. 2023. Creating diverse play-style-centric agents through behavioural cloning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 19, 255–265.

Jeon, H.-C.; Baek, I.-C.; Bae, C.-m.; Park, T.; You, W.; Ha, T.; Jung, H.; Noh, J.; Oh, S.; and Kim, K.-J. 2023. RaidEnv: Exploring new challenges in automated content balancing for boss raid games. *IEEE Transactions on Games*, 16(3): 645–658.

Mao, Y.; Wu, C.; Chen, X.; Hu, H.; Jiang, J.; Zhou, T.; Lv, T.; Fan, C.; Hu, Z.; Wu, Y.; et al. 2024. Stylized offline reinforcement learning: Extracting diverse high-quality behaviors from heterogeneous datasets. In *The Twelfth International Conference on Learning Representations*.

Spearman, C. 1961. The proof and measurement of association between two things.

Sudhakaran, S.; and Risi, S. 2023. Skill decision transformer. *arXiv preprint arXiv:2301.13573*.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *nature*, 575(7782): 350–354.

Xu, M.; Shen, Y.; Zhang, S.; Lu, Y.; Zhao, D.; Tenenbaum, J.; and Gan, C. 2022. Prompting decision transformer for few-shot policy generalization. In *international conference on machine learning*, 24631–24645. PMLR.

Xu, X.; Wang, Y.; Xu, C.; Ding, Z.; Jiang, J.; Ding, Z.; and Karlsson, B. F. 2024. A survey on game playing agents and large models: Methods, applications, and challenges. *arXiv preprint arXiv:2403.10249*.