Scaffolding Human Learning by Shaping Visual Environment

Anonymous CVPR submission

Paper ID *****

Abstract

Learning is an interactive process in which timely guidance plays a crucial role in helping learners overcome difficulties. In now ubiquitous online learning environments, however, such guidance is often unavailable or requires learners to explicitly request help, interrupting their cognitive flow. In our work, we propose a novel paradigm, environment shaping, which proactively supports learners by adaptively modifying the learning environment based on learner behavior. We formulate this problem as a Markov decision process (MDP), where learner states are inferred from multimodal behavioral signals and actions correspond to environment-level interventions such as adaptive visual highlighting. We implement the MDP framework in a block-based coding environment (VEX VR) using vision-language models (VLM) to estimate learner engagement and guide learning. To enable controlled and reproducible evaluation, we further introduce a simulation framework with VLM-simulated agent learners. Preliminary experiments with simulated agents show that environment shaping significantly improves early-stage learning performance and guides learners toward more productive trajectories. These results demonstrate the great promise of proactive environment shaping as a scalable complement to traditional teacher assistance.

1. Introduction

Learning is an interactive process in which students actively explore, test ideas, and refine their understanding [25, 26]. In traditional classroom-based learning settings, teacher assistance plays an important role in supporting students when they encounter difficulties. By providing timely hints or guidance, teachers can help learners overcome obstacles, regain focus, and continue making progress [7].

However, in the now ubiquitous online learning environments, on-the-fly teacher supervision is often unavailable. Many existing online tutoring systems therefore rely on help-on-demand mechanisms, where learners must explicitly request assistance when they feel stuck. While such approaches can be effective, they require learners to interrupt their own learning process in order to ask for help. This interruption can disrupt learners' cognitive flow and may reduce the effectiveness of the learning experience.

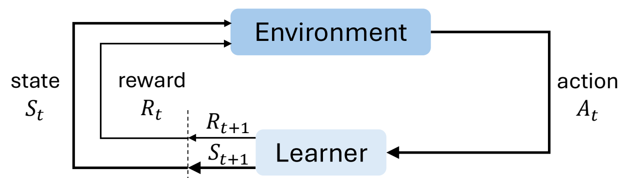


Figure 1. Our algorithm: shaping the environment action to scaffold human learning modeled in Markov Decision Process (MDP).

We explore a novel idea: *proactively supporting learners by shaping the learning environment itself.* Instead of requiring learners to explicitly request help, the system would continuously monitor the learner's behavior and adaptively modify the environment to provide just-in-time guidance whenever needed. By proactively shaping the environment when learners encounter difficulties, the system can guide learners to productive learning states without requiring them to interrupt their cognitive flow (Figure 1). Prior work has shown that learning environments integrated with pedagogical design can significantly influence learners' performance [2, 19, 20]. Our work built on this insight enables an intelligent system to dynamically adapt the environment in response to learner behavior. Importantly, such environment-level support is not intended to replace human instructors. Instead, it serves as a complementary mechanism that provides timely assistance when direct teacher supervision is unavailable, helping maintain productive learning trajectories in self-directed learning environments.

However, designing such an adaptive learning environment introduces two key challenges. First, the system must model the learner's current learning state from observable behavioral signals. Second, based on this inferred learner state, the system must decide when and how to perform an adaptive intervention that guides the learner without introducing unnecessary distraction. To address these two challenges in a general way, we frame the problem of environment shaping as a *Markov decision process (MDP)*. In this formulation, the state represents the current learning context, including the learner's recent interactions and the configuration of the environment. The actions correspond to possible environment-level interventions, such as highlighting relevant objects or modifying visual cues to guide the learner's attention and action. The reward reflects the learner's progress toward solving the task, such as improvements in program correctness or successful task completion. This formulation provides a general solution framework for proactive environment assistance for learners, where the system would iteratively infer the current state of the MDP and choose an optimal action based on the inferred state to shape the learning environment as needed. The general solution framework can be instantiated with a specific learning platform to develop operational algorithms for environment shaping in a specific learning application.

As an initial study of the proposed MDP framework, in our paper, we focus on instantiating it with **VEX VR**, a widely adopted block-based programming environment in which learners control a virtual robot by composing visual code blocks. The platform supports rich visual interactions among code blocks, robot behaviors, and environmental state transitions. We selected VEX VR for two reasons. First, VEX VR are extensively used in programming education and have reached over 1.45 million users across 151 countries [28]. Second, visual interventions are among the most intuitive and effective ways to shape learning environments; VEX VR's visually rich interface thus provides an ideal testbed for studying how the visual understanding and generation capabilities of modern AI systems can support environment shaping.

Specifically, as shown in Figure 2, our system monitors multimodal behavioral signals, including eye gaze, mouse interactions, and coding activity, to estimate learner engagement and task progress. Using vision-language models, the system identifies situations in which a learner may be losing focus and introduces adaptive visual highlighting to guide the learner's attention toward relevant objects or program elements. Note that although our system was implemented in the context of VEX VR, the whole system architecture is generally applicable to any visual learning environment to support environment shaping.

Evaluating environment shaping presents a new research challenge. Since this problem has not been studied before, no established datasets or benchmarks currently exist. While one possible approach is to conduct user studies with human learners, such studies make it difficult to perform controlled and reproducible comparisons across different algorithms. In particular, when using human learners, it is inherently difficult to ensure fair comparisons: once a learner has completed a task under one system, their cognitive state has already changed, making it impossible to evaluate another system with the same initial conditions. To enable systematic evaluation, we explore using large language models to simulate learners interacting with the environment. This setup allows us to conduct controlled experiments and compare multiple environment-shaping strategies under identical conditions. In particular, it enables us to evaluate key components of our framework, including learner state estimation and adaptive visual guidance. Using simulated learners, our experiment results demonstrate that adaptive visual interventions can effectively guide learners toward more productive learning trajectories.

To evaluate our approach, we conduct experiments with both automated GUI agents and human learners in the VEX VR environment. For agents, our assistant improves performance from 1 kg to 55.2 kg (after five assistive tips) in the Coral Reef Cleanup playground, demonstrating over 50× gain in solution quality, but further guidance leads to degradation, revealing concentration loss in iterative reasoning. For human learners, preliminary observations of 14 students show that most achieve minimal improvement without assistance, indicating substantial difficulty in unguided exploration. These results show that environment-level interventions can significantly improve early-stage learning performance while exposing a key limitation of VLM-driven learners in maintaining long-horizon guidance.

We summarize our contributions as follows:

- We introduce **environment shaping** as a novel proactive paradigm for AI-assisted learning. Instead of relying on help-on-demand interactions, our approach supports learners by adaptively modifying the visual learning environment, allowing guidance to be provided without interrupting the learner's cognitive flow.
- We formulate environment shaping as a **Markov decision process (MDP)**, providing a general framework for modeling the interaction between learners and adaptive learning environments. This formulation enables systematic exploration about when and how environmental interventions should be applied.
- We instantiate this framework in a visually rich programming learning environment (**VEX VR**) by leveraging large visual models to detect learner disengagement and difficulties and generate adaptive visual interventions. To evaluate this new problem setting, we introduce a simulation-based evaluation framework using VLM-based agent learners, enabling reproducible comparison of environment-shaping strategies. We show the great promise of environment shaping by presenting encouraging preliminary experiment results.

1.1. Problem Formulation

We study how an intelligent system can support human learning by adaptively shaping the visual learning environment. Instead of requiring learners to explicitly request assistance, the system monitors learner behavior and modifies the environment to provide guidance when needed.

We model the interaction between the learner and the environment as a **Markov Decision Process (MDP)**. At each time step t, the system observes the learner state s_t, selects an intervention action a_t, and receives a reward r_t reflecting the learner's progress.

Formally, the MDP is defined as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R)$, where $\mathcal{S}$ denotes the learner state space, $\mathcal{A}$ denotes possible environment interventions, $P(s_{t+1}|s_t, a_t)$ denotes models learner behavior transitions, and $R(s_t, a_t, s_{t+1})$ measures learning progress. The goal of the system is to select interventions that guide learners toward productive learning trajectories while preserving natural exploration. The MDP formulation naturally provides a general solution framework for solving the environment shaping problem computationally, where the system would continuously infer the current learner state and decide whether and how to intervene by choosing an optimal intervention action based on the inferred state. The general solution framework can be instantiated with any specific learning environment to derive an operational algorithm for environment shaping in a specific application system. Below, we discuss such an instantiation with the VEX VR visual learning environment.

1.2. Learner State Representation

To determine whether learners need assistance, the system constructs a representation of the current learning context from multimodal behavioral signals.

The learner state is defined as $s_t = (g_t, m_t, c_t)$, where g_t represents **eye gaze activity**, m_t represents **mouse interaction activity**, and c_t represents **coding activity**.

Eye Gaze Activity Eye gaze signals capture the learner's visual attention in the environment. We compute several gaze-based features, including fixation duration on objects, gaze entropy across regions of interest, and the frequency of gaze shifts. These signals help identify situations learners may be searching for relevant information or getting lost.

Mouse Interaction Activity Mouse interaction reflects how learners manipulate code blocks in the programming interface. Extracted features include drag operations on code blocks, mouse trajectory length, and idle time between interactions. Frequent rearrangement or prolonged inactivity may indicate confusion or difficulty.

Coding Activity Activity Coding activity measures the learner's progress toward solving the programming task. Relevant signals include the number of blocks in the current program, syntactic correctness, and partial task completion status. These signals provide a direct estimate of task progress.

The combined state representation captures both behavioral engagement and problem-solving progress.

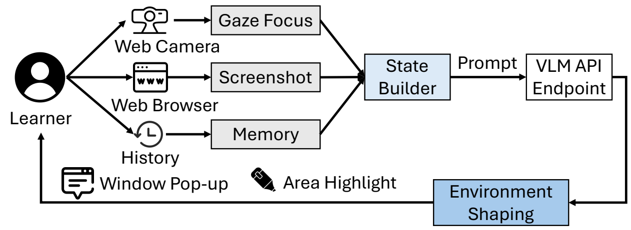


Figure 2. Our system: a general MDP-based implementation.

1.3. Action Space: Environment Interventions

The system assists learners by modifying visual elements of the environment. The intervention action space is defined as $a_t \in \mathcal{A}$, where actions correspond to different forms of visual environment shaping.

In our block-based programming environment, the action set includes

- **Highlight Code Blocks:** emphasize code blocks relevant to the current task.
- **Highlight Environment Objects:** highlight objects in the virtual playground related to the task.
- **Visual hints:** introduce visual cues such as pop-up windows with hints for next steps that guide learners.
- **No Intervention:** allow the learner to continue without assistance.

These actions modify the visual environment while preserving the learner's autonomy.

2. A Markov Decision Process Framework for Environment Shaping

2.1. Transition Model

After an intervention is applied, the learner may adjust their behavior in response to the modified environment, resulting in a transition to a new learner state.

This process is modeled as $s_{t+1} \sim P(s_{t+1} \mid s_t, a_t)$, where P represents the transition dynamics of learner behavior.

Since human learning behavior is difficult to model analytically, we estimate the transition probabilities empirically from interaction data. Let $N(s, a, s')$ denote the number of observed transitions from state s to state s' after applying action a.

The transition probability is estimated as

$$P(s'|s, a) = \frac{N(s, a, s')}{\sum_{s''} N(s, a, s'')} \tag{1}$$

This empirical transition model allows the system to approximate how environment interventions influence learner behavior.

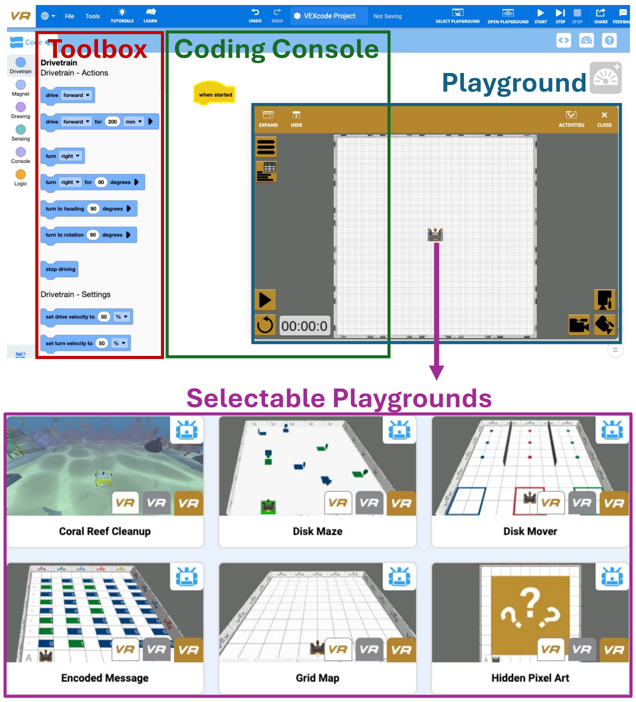


Figure 3. Illustration of Vex VR Testbed. It is generally divided into three components of toolbox, coding console, and playground.

2.2. Reward Signal

The reward function measures the learner's progress toward solving the programming task.

Formally, the reward is defined as

$$r_t = R(s_t, a_t, s_{t+1}) \tag{2}$$

In our environment, the reward is derived from indicators of task progress, including

- improvements in program correctness,
- successful task completion,
- resumed interaction after periods of inactivity.

The reward signal encourages interventions that help learners return to productive learning trajectories.

2.3. Intervention Policy

While the interaction between the learner and the environment is formulated as an MDP, our goal is not to learn a reinforcement learning policy. Instead, the MDP formulation provides a principled framework for reasoning about intervention timing and effects.

In practice, the intervention policy is implemented using a rule-based decision mechanism based on the estimated learner state.

Specifically, the system evaluates a progress function $f(s_t)$ derived from behavioral signals and coding activity.

The intervention policy is defined as

$$a_t = \begin{cases} \text{intervention}(s_t), & f(s_t) < \tau \\ \text{no action}, & \text{otherwise} \end{cases} \tag{3}$$

where τ is a threshold representing insufficient learning progress.

When the learner appears to be struggling, the system selects an intervention that highlights relevant elements in the environment.

2.4. Environment Shaping

Once an intervention action is selected, the system modifies the visual environment accordingly.

Unlike traditional tutoring systems that provide explicit hints, our approach focuses on subtle visual guidance embedded directly in the environment. These interventions influence the learner's perception of the task space and guide attention toward relevant information.

Because the guidance is integrated into the environment itself, the learner can continue interacting naturally without interrupting the learning process.

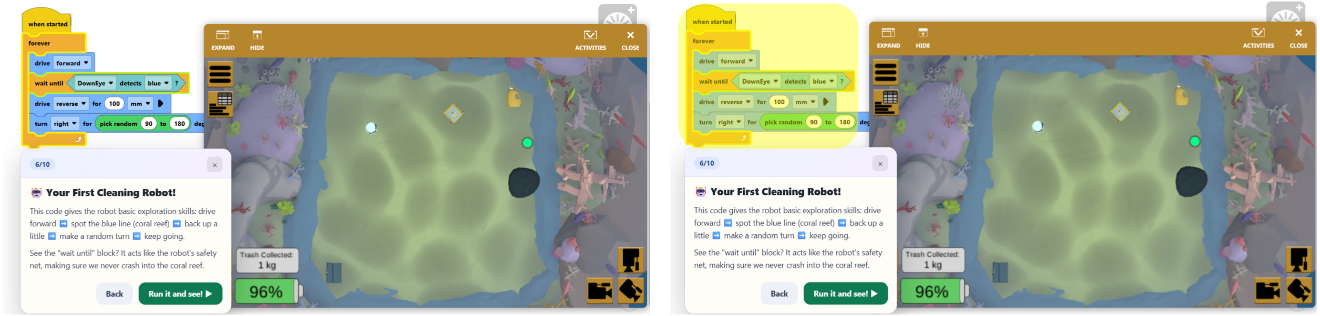
3. Implementation

Testbed. We employ the VEX platform as our programming testbed, as illustrated in Figure 3. The interface contains three main components. The *toolbox* on the left provides functional blocks for program construction. The *coding console* in the center allows users to drag and compose these blocks into an executable program. On the right is the *playground*, a floating window where the program is executed. The generated program controls a robot (e.g., a vehicle) within the playground to perform tasks such as collecting garbage or navigating the environment.

Playground Setting. We evaluate our system using the *Coral Reef Cleanup Playground* in VEX VR [28]. This playground simulates an underwater environment where a programmable robot collects trash from a coral reef. The objective is to remove as much waste as possible while operating under limited battery energy. The task requires a combination of robot navigation, sensing, and strategy design, and naturally involves programming concepts such as sequencing, control flow, and efficient path planning.

Environment Assistant. We implement an intelligent environment-assistant browser extension based on the principles of scaffolding human learning. As shown in Figure 2, at fixed decision intervals, the system collects multimodal signals during web-based learning. A webcam captures the learner's facial image, and WebGazer [21] estimates the on-screen gaze location. Meanwhile, the extension captures screenshots of the active webpage and records the browsing history as a memory log.

These signals are aggregated by a learner state builder, which constructs a structured prompt containing the gaze



(a) (b)

Figure 4. The demonstration of the environment assistant (highlight and pop-up window).

information (which is the green dot shown in Figure 4), the current webpage screenshot, and the interaction history. The prompt is then sent to a VLM API to estimate the current learner state s_t. At each time step, the system re-evaluates the learner state, updates the progress estimate $f(s_t)$, and computes the reward $r_{t-1} = R(s_{t-1}, a_{t-1}, s_t)$ via an interpolation between the consecutive progress estimates $f(s_{t-1})$ and $f(s_t)$, yielding a temporally smoothed signal of learning progress. Based on the estimated state and reward, the system selects an intervention action as shown in Figure 4. The browser extension executes the selected intervention by displaying textual hints in pop-up windows or highlighting relevant regions of the webpage, thereby providing real-time feedback.

GUI Agent Learner. For reproducibility and to emulate automated learner behavior, we use Google Antigravity [9] as a graphical user interface (GUI) agent that interacts with the VEX platform. The agent is prompted to open the VEX website and access the course interface (the prompt is shown in the listing). During the experiment, the agent also receives guidance tips generated by the VLM. Upon receiving each environmental visual supports such as pop-up windows, the agent searches for the corresponding functional blocks, drags them into the coding console, and executes the resulting program. We implement the agent using Gemini 3 Pro [10], and each experiment is repeated five times. For the baseline setting without our system, the agent explores the VEX interface independently and constructs programs based solely on its internal reasoning.

4. Preliminary Experiment

Our evaluation is designed to validate the proposed MDP-based formulation for adaptive learning assistance. Recall that we model the learning process as a Markov Decision Process, where the system observes the learner state and selects intervention actions to guide learning. In our implementation, these interventions are realized by *reshaping the learning environment*, e.g., highlighting relevant regions or providing contextual prompts, rather than requiring explicit queries from the learner.

The goal of the evaluation is to examine whether such environment-level interventions can effectively guide learners toward more productive trajectories. In particular, we aim to answer the following research questions:

- **RQ1:** Can the environment assistant improve the solution quality achieved by the GUI agent learner?
- **RQ2:** Do AI-simulated GUI agent learners maintain consistent improvement when receiving iterative guidance, or do they exhibit signs of losing concentration?
- **RQ3:** How do novice human learners approach the programming task without assistance?
- **RQ4:** What differences exist between human learners and GUI agent learners in utilizing generated guidance?

4.1. Evaluation Challenges and Solution

Evaluating AI-assisted learning systems in our setting is challenging for several reasons. First, the Coral Reef Cleanup task in VEX VR is a newly introduced task that has not been systematically studied in prior work. As a result, there are no existing datasets or standardized benchmarks that directly support our evaluation. Therefore, we build a real-time system to analyze this novel task. Second, while user studies with human learners provide realistic insights, they make it challenging to conduct controlled and reproducible comparisons across different algorithms. To enable systematic and scalable evaluation, we adopt large

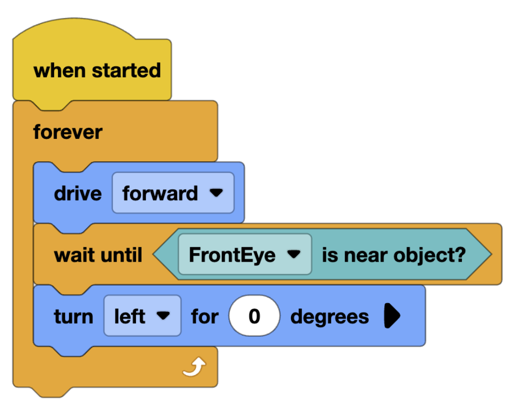
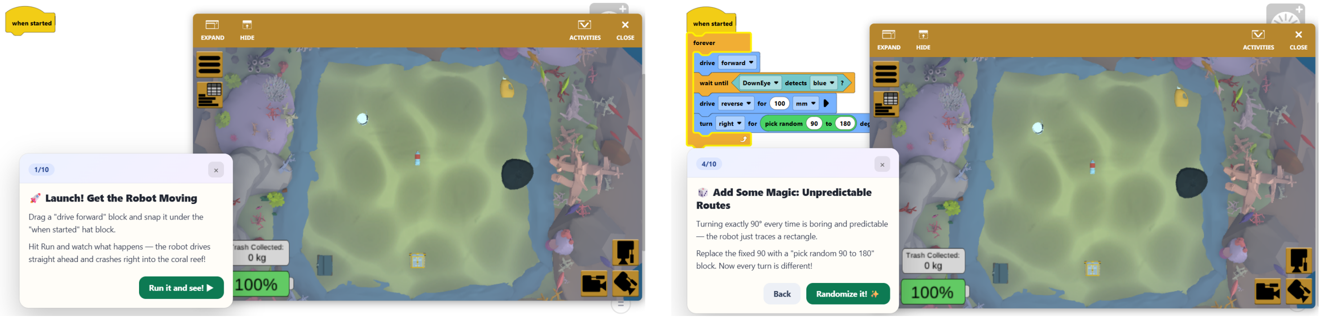
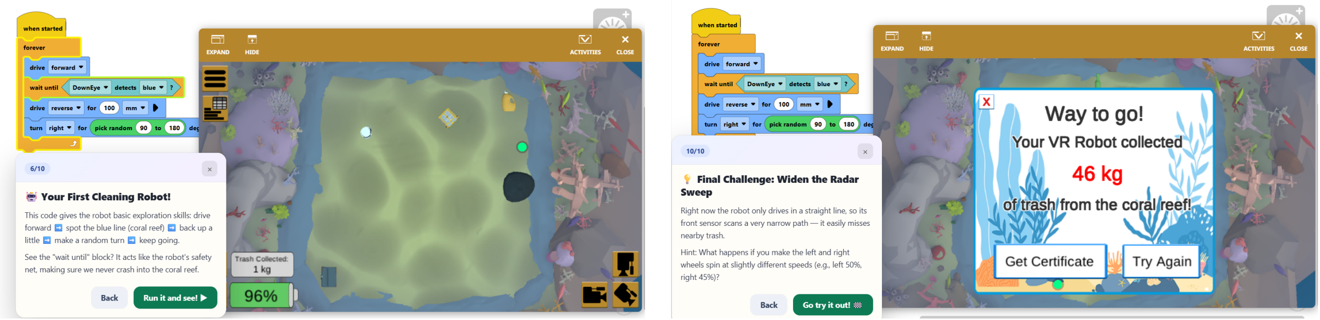


Figure 5. Program built by GUI agent without assistance.



(a) (b)

Figure 6. Basic learning scenario. The learner starts from scratch and gradually builds the solution with system guidance.



(a) (b)

Figure 7. Advanced learning scenario. The learner starts from a working example and focuses on improving the existing solution.

language models to simulate learners interacting with the environment, allowing us to compare different intervention strategies under controlled conditions.

4.2. Experimental Setup

GUI Agent Learners. We evaluate a GUI agent that interacts with the programming interface by searching, dragging, and composing functional blocks. The agent receives textual tips generated by our learning assistant and attempts to incorporate them into the program. For the baseline setting without assistance, the agent explores the VEX interface independently and constructs programs based solely on its internal reasoning.

Human Learners (Preliminary Observation). To understand how novice learners approach the task, we analyze video recordings of classroom activities involving middle-school students using the VEX VR platform. The dataset contains 14 students, each with approximately one hour of interaction time. Since these recordings were collected prior to the development of our system, the students did not receive assistance from our learning assistant. Our goal is to observe common challenges and behaviors when learners attempt the task through unguided exploration.

Evaluation Metrics. We measure task performance using the score provided by the VEX VR playground, which corresponds to the amount of garbage collected (kg). This metric directly reflects the effectiveness of the generated program and the efficiency of the robot's navigation strategy. We also analyze the time required to reach different performance levels in order to understand learning progress over time. For the agent learner, we also examine whether the score continues to improve after successive guidance tips, which helps us assess whether the learner can sustain attention and effectively use iterative assistance.

4.3. Results on Agent Learners (RQ1 & RQ2)

We first evaluate whether the environment assistant can improve the performance of GUI agent learners and whether the agent can consistently benefit from iterative guidance.

Without assistance, the GUI agent learner is unable to meaningfully solve the task. Although it can open the webpage, search for programming blocks, and construct a program, the resulting implementation is naive and achieves a score of only about 1 kg even after prolonged execution. As shown in Figure 5, the generated program simply drives the robot along a straight path to collect garbage until it reaches the boundary of the playground. This behavior demonstrates that the agent lacks the reasoning capability required to discover an effective program structure through unguided exploration. When assisted by our system, the agent initially benefits from the generated guidance. Around 150 seconds (after Tip 5), the agent achieves a score of 55.2 kg, indicating that the assistant successfully guides the agent toward a substantially better program.

CVPR #*****

However, additional guidance does not lead to further improvement. After the fifth tip, the agent's performance stagnates and then slightly degrades. Eventually, after receiving all ten tips, the agent fails to produce a runnable program. This pattern suggests that the GUI agent cannot reliably maintain effective reasoning across a long sequence of iterative instructions. In other words, the agent appears to gradually lose concentration when processing multiple rounds of guidance, leading to unstable improvements and even regression in later stages.

This observation is important because it reveals that improving an VLM-driven learner is not only a matter of providing better environment support, but also of ensuring that the learner can remain focused and consistently integrate those tips over time. Our results therefore motivate future work on how to prevent learners from losing concentration during long-horizon interactive problem solving.

4.4. Observations from Human Learners (RQ3)

Next, we analyze how novice human learners approach the same task without assistance. From the recorded interaction videos, we observe that most students struggle during the initial exploration phase. During the first 3-4 minutes, learners typically browse the available blocks in the toolbox and attempt to construct programs through trial-and-error. Many students are unable to successfully run their programs, while those who do often implement only simple logic and collect about 1 kg of garbage.

These observations suggest that novice learners have difficulty identifying effective program structures in block-based programming environments without guidance. The main challenges include understanding how different blocks interact, designing a reasonable navigation strategy for the robot, and debugging incomplete or incorrect programs. Overall, the human learners' behavior indicates that this task imposes substantial cognitive burden when no real-time assistance is available.

Since our current human analysis is based only on pre-existing videos rather than a controlled assisted study, we do not yet claim quantitative gains for human learners with our system. Instead, these observations serve as preliminary evidence that the task is challenging and that there is substantial opportunity for AI-assisted support. In future work, we plan to conduct controlled user studies with the learning assistant enabled for human learners once IRB approval is obtained, as discussed in Sec. 6.

4.5. Comparing Human and Agent Learners (RQ4)

Comparing the two settings reveals several differences in how humans and automated agents approach the programming task. Without assistance, human learners are eventually able to discover partial solutions through exploration, while the GUI agent remains stuck at trivial strategies. Humans appear to rely on intuitive reasoning and incremental experimentation when constructing programs. Although their progress is slow, they can still adapt their behavior based on partial observations from the environment.

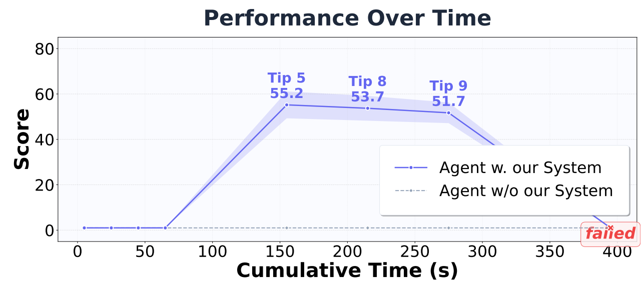


Figure 8. Learning curve of the GUI agent. Each point represents the solution of the agent after receiving each tip The time interval between points includes the inference time of the learning assistant, the time the agent spent on dragging and placing the blocks, and the time for evaluating the program.

In contrast, the GUI agent relies on procedural interaction with the interface, which introduces additional overhead in searching for blocks and manipulating the programming environment. More importantly, while the agent can benefit from early-stage guidance, it struggles to consistently translate a sequence of high-level tips into correct program modifications. This concentration loss is much less evident in humans, who are generally better able to preserve the semantic intent of prior guidance while incorporating new instructions.

These findings highlight that human learners and automated GUI agents exhibit fundamentally different learning behaviors, and that AI-generated guidance must be designed to accommodate these differences.

4.6. Case Study of Agent Learner

To further illustrate how the system assists the learning process, we conduct two case studies.

The **basic learning scenario** (Figure 6) examines the situation where a learner starts from scratch and gradually constructs a solution with guidance from the system. The assistant provides step-by-step prompts that help the learner identify relevant blocks, place them appropriately, and progressively improve the program.

The **advanced learning scenario** (Figure 7) begins with a partially completed program, where the learner aims to refine and improve an existing solution. Here, the generated prompts focus more on correcting suboptimal logic and guiding the learner toward higher-quality implementations.

In both scenarios, the system generates prompts that adapt to the learner's current progress and interaction context, enabling personalized assistance during the programming process. These case studies provide qualitative evidence that the assistant can tailor its feedback to different stages of learning and support both initial construction and subsequent refinement.

5. Related Work

Intelligent Tutoring and Environment Shaping. In the evolution of Intelligent Tutoring Systems (ITS), resolving the "assistance dilemma"—balancing the provision of instructional support with the necessity of productive student struggle—has remained a central pedagogical challenge [12, 16, 25]. Historically, systems navigated this dilemma via reactive, help-on-demand mechanisms or proactive explicit interventions, both of which risk interrupting the learner's cognitive flow or inducing help-avoidance behaviors [15]. To optimize intervention timing, prior works have formulated pedagogical planning as a Markov Decision Process (MDP), most notably through policy-centric models like the Hint Factory [24]. Our work conceptually diverges by treating the environment itself as the reinforcement learning agent. Grounded in ecological niche construction theories [8], we propose implicit visual scaffolding that alters visual affordances rather than explicitly overriding the learner's policy. To enable this low-interruption paradigm, our system integrates Multimodal Learning Analytics (MMLA), triangulating eye-tracking, mouse kinematics, and coding logs to continuously assess learner engagement and derive learning progress gradients [21, 23]. This continuous state assessment allows the RL agent to dynamically shape the environment to maintain the learner within an optimal zone of productive struggle.

Visual Language Models and Agent-Based Evaluation in GUI Environments. Implementing environment shaping within block-based visual programming spaces introduces significant perceptual challenges, as program semantics are deeply nested in graphical topologies rather than plain text [22]. To interpret these complex visual workspaces, we leverage recent advancements in Vision-Language Models (VLMs) tailored for Graphical User Interface (GUI) comprehension, such as Spotlight [13], ScreenAI [4], and ShowUI [14]. These models excel at zero-shot visual grounding and widget parsing without requiring underlying DOM access, bridging the semantic gap between a learner's visual attention and the evolving state of the programming playground [30]. Furthermore, empirically validating dynamic tutoring interventions typically requires costly and high-variance longitudinal human studies [17]. To overcome this evaluation bottleneck, our research adopts an emerging machine-to-machine evaluation paradigm [18, 29]. By utilizing advanced generative agent orchestration platforms–such as Google Antigravity [9]–we deploy browser-controlling simulated students that possess autonomous visual reasoning and interaction capabilities. This closed-loop setup enables the continuous refinement of the environment-shaping RL policy $Q(s, a)$ through extensive simulated rollouts prior to human deployment, establishing a robust, scalable framework for testing next-generation AI-assisted learning interfaces.

6. Limitations & Future Work

Large-scale Human Studies. In this paper, we conduct a preliminary analysis with IRB-approved video of human learners. While the results demonstrate the potential of our approach, a larger-scale study is required to fully understand the effectiveness of the system across diverse populations. In future work, we plan to recruit learners from different age groups, educational backgrounds, and programming experience levels to evaluate how the assistant affects learning outcomes in broader settings. We plan to divide students into two groups (with and without our system), let the student complete the VEX challenges, and measure their completion score, completion time, and the completion status over time. We also plan to extend the evaluation beyond the Coral Reef Cleanup playground to additional VEX VR environments and other programming platforms, enabling us to study how the system supports learning across different tasks and difficulty levels.

Multimodal Signal Integration. Our current prototype collects multimodal signals such as eye gaze, webpage screenshots, and browsing history. However, the interaction between eye tracking and area highlighting is not explored in the current implementation, which will be evaluated in a large-scale human study in our future work. Future work will incorporate tighter multimodal fusion to jointly analyze gaze patterns, cursor movement, and interaction sequences. Such integration can help the system detect learner distraction, confusion, or hesitation more accurately.

Generalization to Other Learning Domains. Our current system focuses on block-based programming tasks in the VEX VR environment. Although the framework is general, additional work is needed to validate its effectiveness in other learning domains such as mathematics problem solving [3, 11], robotics programming [5, 6], or interactive simulations [1, 27]. Future work will explore how the multimodal assistant can be adapted to support a wider range of educational environments and digital learning platforms.

7. Conclusion

Motivated by the lack of timely guidance in online learning and the limitations of help-on-demand systems, we introduce *environment shaping* as a proactive approach that supports learners without interrupting their cognitive flow. We formulate it as a Markov Decision Process and instantiate it in VEX VR using multimodal signals and vision-language models to provide adaptive visual guidance. Our results show that such interventions can significantly improve learning performance, highlighting the promise of proactive environment support and motivate future work on robust long-horizon learning support.

References

[1] Clark Aldrich. *Learning by doing: A comprehensive guide to simulations, computer games, and pedagogy in e-learning and other educational experiences*. John Wiley & Sons, 2005. 8

[2] David Alpizar, Olusola O Adesope, and Rachel M Wong. A meta-analysis of signaling principle in multimedia learning environments. *Educational Technology Research and Development*, 68(5):2095–2119, 2020. 1

[3] John R Anderson, Albert T Corbett, Kenneth R Koedinger, and Ray Pelletier. Cognitive tutors: Lessons learned. *The journal of the learning sciences*, 4(2):167–207, 1995. 8

[4] Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Cărbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. Screenai: A vision-language model for ui and infographics understanding. *arXiv preprint arXiv:2402.04615*, 2024. 8

[5] Tony Belpaeme, James Kennedy, Aditi Ramachandran, Brian Scassellati, and Fumihide Tanaka. Social robots for education: A review. *Science robotics*, 3(21):eaat5954, 2018. 8

[6] Fabiane Barreto Vavassori Benitti. Exploring the educational potential of robotics in schools: A systematic review. *Computers & education*, 58(3):978–988, 2012. 8

[7] Simon KS Cheung, Lam For Kwok, Kongkiti Phusavat, and Harrison Hao Yang. Shaping the future learning environments with smart elements: challenges and opportunities. *International Journal of Educational Technology in Higher Education*, 18(1):16, 2021. 1

[8] Andy Clark. Language, embodiment, and the cognitive niche. *Trends in cognitive sciences*, 10(8):370–374, 2006. 8

[9] Google. Google antigravity: An agent-first ai development platform. https://antigravity.google/, 2025. AI-powered IDE with autonomous agents, accessed: 2026-03-16. 5, 8

[10] Google DeepMind. Gemini pro. https://deepmind.google/models/gemini/pro/, 2026. Accessed: 2026-03-16. 5

[11] Arthur C Graesser, Patrick Chipman, Brian C Haynes, and Andrew Olney. Autotutor: An intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions on Education*, 48(4):612–618, 2005. 8

[12] Kenneth R Koedinger and Vincent Aleven. Exploring the assistance dilemma in experiments with cognitive tutors. *Educational psychology review*, 19(3):239–264, 2007. 8

[13] Gang Li and Yang Li. Spotlight: Mobile ui understanding using vision-language models with a focus. *arXiv preprint arXiv:2209.14927*, 2022. 8

[14] Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui visual agent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19498–19508, 2025. 8

[15] Sruti Mallik and Ahana Gangopadhyay. Proactive and reactive engagement of artificial intelligence methods for education: a review. *Frontiers in artificial intelligence*, 6:1151391, 2023. 8

[16] Amogh Mannekote, Adam Davies, Juan D. Pinto, Shan Zhang, Daniel Olds, Noah L. Schroeder, Blair Lehman, Diego Zapata-Rivera, and ChengXiang Zhai. Large language models for whole-learner support: opportunities and challenges. *Frontiers in Artificial Intelligence*, Volume 7 - 2024, 2024. 8

[17] Amogh Mannekote, Adam Davies, Jina Kang, and Kristy Elizabeth Boyer. Can llms reliably simulate human learner actions? a simulation authoring framework for open-ended learning environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 29044–29052, 2025. 8

[18] Luis Marquez-Carpintero, Alberto Lopez-Sellers, and Miguel Cazorla. Simulating students with large language models: A review of architecture, mechanisms, and role modelling in education with generative ai. *arXiv preprint arXiv:2511.06078*, 2025. 8

[19] Michael Noetel, Shantell Griffith, Oscar Delaney, Nicola Rose Harris, Taren Sanders, Philip Parker, Borja del Pozo Cruz, and Chris Lonsdale. Multimedia design for learning: An overview of reviews with meta-meta-analysis. *Review of Educational Research*, 92(3):413–454, 2022. 1

[20] Erol Ozcelik, Ismahan Arslan-Ari, and Kursat Cagiltay. Why does signaling enhance multimedia learning? evidence from eye movements. *Computers in human behavior*, 26(1):110–117, 2010. 1

[21] Alexandra Papoutsaki. Scalable webcam eye tracking by learning from user interactions. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, page 219–222, New York, NY, USA, 2015. Association for Computing Machinery. 4, 8

[22] Anna Riedmann, Philipp Schaper, and Birgit Lugrin. Reinforcement learning in education: A systematic. 2025. 8

[23] Kshitij Sharma and Michail Giannakos. Carry-forward effect: providing proactive scaffolding to learning processes. *Behaviour & Information Technology*, 44(11):2760–2799, 2025. 8

[24] John Stamper, Tiffany Barnes, Lorrie Lehmann, and Marvin Croy. The hint factory: Automatic generation of contextualized help for existing computer aided instruction. In *Proceedings of the 9th international conference on intelligent tutoring systems young researchers track*, pages 71–78. Montreal, Canada, 2008. 8

[25] John Sweller. The role of evolutionary psychology in our understanding of human cognition: Consequences for cognitive load theory and instructional procedures. *Educational Psychology Review*, 34(4):2229–2241, 2022. 1, 8

[26] John Sweller. Cognitive load theory and individual differences. *Learning and Individual Differences*, 110:102423, 2024. 1

[27] Kurt VanLehn. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational psychologist*, 46(4):197–221, 2011. 8

[28] VEX Robotics Education Foundation. A virtual robotic solution: Insights from implementation and implications for the future. https://kb.vex.com/hc/en-us/articles/4406167739028-A-Virtual-Robotic-Solution-Insights-from-Implementation-and-Implications-for-the-Future, 2024. VEX Knowledge Base, accessed March 2026. 2, 4

[29] Songlin Xu, Hao-Ning Wen, Hongyi Pan, Dallas Dominguez, Dongyin Hu, and Xinyu Zhang. Classroom simulacra: Building contextual student generative agents in online education for learning behavioral simulation. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2025. Association for Computing Machinery. 8

[30] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023. 8