GAMECONSOLE: FINE-GRAINED GAMEPLAY DESCRIPTION WITH EXECUTABLE KNOWLEDGE CARTRIDGES

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-grained gameplay description requires identifying game-defined entities and states in observed events. The resources and recognition strategies needed to resolve these details vary across games, making adaptation difficult when game-specific knowledge is tightly coupled with recognition pipelines. We introduce GAMECONSOLE, a console–cartridge framework that separates game-specific resources and recognition bindings from reusable recognition operations. A replaceable knowledge cartridge contains entity definitions, visual references, relations, and target-specific recognition bindings, while a shared console uses these bindings to instantiate recognition programs from a common operator library. These programs combine complementary visual evidence and use game-defined relations to constrain candidate identities. Event-aware sampling allocates observations according to visual change, and the resulting observations are combined with source-linked recognition facts to generate temporally localized descriptions. Across three games and three generation backends, GAMECONSOLE improves mention F1 and reference-conditioned evidence scores over direct and knowledge-enhanced baselines under shared video observations. Target-level comparisons further show that different recognition targets benefit from different operators. Downstream QA on Valorant shows that the generated descriptions improve recovery of game-specific details and temporal retrieval. The code is available here: `https://anonymous.4open.science/r/GameConsole-7FD6`

1 INTRODUCTION

Video description expresses observed events in language by identifying the participants and describing how their actions unfold (Krishna et al., 2017; Wang et al., 2021; Yang et al., 2023). In gameplay videos, however, informative descriptions often require resolving entities and states defined by a particular game. A weapon may be recognized from its visual appearance or an interface icon, whereas identifying a location may require interpreting a map view. Gameplay benchmarks have highlighted persistent challenges in recognizing game-specific entities and attributing actions to the correct participants (Wang et al., 2026; Ma et al., 2026). We therefore study *fine-grained gameplay description*, where temporally localized event descriptions capture not only the observed actions, but also the game-defined entities, observable states, and participant roles involved in those events.

Grounded video description connects language to visual evidence (Zhou et al., 2019; Jiang et al., 2024), while knowledge-enhanced commentary incorporates domain information to describe specific entities and events (Qi et al., 2023; Xi et al., 2025; Jin et al., 2026). In gameplay videos, resolving these details can require different recognition strategies, such as matching against visual references, using game-defined relations to restrict candidate identities, or combining multiple visual cues. Although the underlying recognition operations can be shared across games, the references, relations, and configurations that govern their use are game-specific. If these game-specific components are tightly coupled with the recognition pipeline, adapting to a new game requires the corresponding recognition process to be reimplemented. This motivates a framework in which game-specific knowledge determines how shared recognition operations are instantiated and composed.

We introduce GAMECONSOLE, a console–cartridge framework that separates game-specific recognition resources and bindings from reusable recognition operations (Figure 1(a)). Each replaceable knowledge cartridge contains entity definitions, visual references, relations, and target-specific bind-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

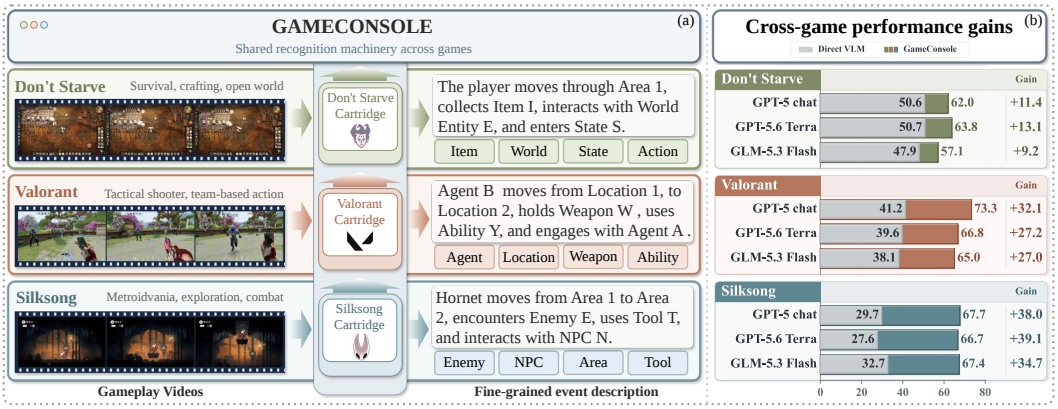

Figure 1: **GameConsole across games.** (a) A shared recognition console uses game-specific knowledge cartridges to produce fine-grained event descriptions across different games. (b) The gains over direct VLM generation vary across games, with larger improvements where direct generation recovers fewer reference details.

ings. Here, *executable* means that these bindings determine which recognition operations run and how game-defined relations constrain their inputs. The shared console instantiates the corresponding programs, allowing a cartridge to change game-specific recognition behavior while retaining the same operations and execution mechanisms.

Recognizing game-specific entities is only part of the task, as a description must also determine when an event occurs and how the recognized entities participate in it. GAMECONSOLE localizes event windows from visual changes, retains the source timestamps of recognized facts, and combines these facts with temporally ordered visual observations. Its event-aware sampling allocates fewer observations to visually stable content and denser observations around transitions. The resulting evidence allows a vision-language model (VLM) to describe actions and state changes within each interval while grounding game-specific identities and states in source-linked recognition results.

Across three games and three generation backends, GAMECONSOLE consistently improves mention F1 and reference-conditioned evidence scores over direct and knowledge-enhanced baselines under shared video observations. The gains vary across games and are larger when direct generation recovers fewer game-specific details, as illustrated in Figure 1(b). On Valorant, the generated descriptions also improve downstream QA for post-match review, event explanation, and temporal retrieval, showing that the recovered details remain useful beyond description evaluation.

Our contributions are threefold.

- **Fine-grained gameplay description.** We study temporally localized gameplay description as a cross-game task in which descriptions identify game-defined entities, observable states, and participant roles and associate them with the relevant video intervals.
- **Console–cartridge framework.** We introduce GAMECONSOLE, which separates replaceable game-specific resources and recognition bindings from reusable recognition operations, enabling each cartridge to configure executable recognition programs within a shared operator library.
- **Empirical study.** We evaluate description quality and target-specific recognition choices across games and generation backends, and assess how the resulting descriptions support post-match review, event explanation, and temporal retrieval.

2 RELATED WORK

2.1 GAMEPLAY VIDEO UNDERSTANDING AND DESCRIPTION

Gameplay video understanding requires connecting visual observations to game-defined events, states, and participants. MarioQA (Mun et al., 2017) studies temporal relations among gameplay events,

while GameplayQA (Wang et al., 2026) examines dense state–action understanding and participant attribution in synchronized multiplayer videos. EgoEsportsQA (Ma et al., 2026) further evaluates fine-grained perception and expert reasoning in esports footage. Beyond question answering, prior work on racing games (Ishigaki et al., 2021) generates commentary from aligned visual, textual, and structured numerical information, connecting gameplay understanding to language generation. In contrast, we study fine-grained, temporally localized gameplay description across games, where entity definitions, visual references, and state representations may change while the underlying recognition operations remain reusable.

2.2 Grounded and Knowledge-Enhanced Video Description

Video description requires both temporal localization and grounding of linguistic content in visual evidence. ActivityNet Captions (Krishna et al., 2017) and subsequent dense video captioning methods (Wang et al., 2021; Yang et al., 2023) study the localization and description of events in untrimmed videos, while grounded video description (Zhou et al., 2019; Jiang et al., 2024) associates linguistic expressions with localized entities and actions across frames. External knowledge further supports domain-specific descriptions: GOAL (Qi et al., 2023) incorporates structured and unstructured knowledge for soccer commentary, KEANet (Xi et al., 2025) uses candidate player images and names for entity-aware basketball captioning, and GameSight (Jin et al., 2026) combines visual and contextual reasoning for entity alignment before enriching commentary with external information. Video-RAG (Luo et al., 2026) further illustrates how tool-extracted and retrieved information can support video understanding. GameConsole instead organizes game-specific visual references, entity relations, and recognition bindings so that game-specific knowledge can configure reusable recognition operations.

2.3 Visual Programming and Modular Adaptation

Visual programming provides a natural mechanism for composing reusable perception and reasoning components. VisProg (Gupta & Kembhavi, 2023) generates modular visual programs from natural-language instructions, while ViperGPT (Surís et al., 2023) composes visual APIs through generated code. For video understanding, ProViQ (Choudhury et al., 2023) uses procedural programs for question answering, MoReVQA (Min et al., 2024) separates parsing, grounding, and reasoning, and VideoAgent (Fan et al., 2024) combines structured memory with localization and querying tools. Modular adaptation provides a related perspective; LEFT (Hsu et al., 2023), for example, separates a domain-independent logical executor from trainable domain-specific grounding modules. GameConsole follows this modular perspective for game-level recognition adaptation: game-specific resources and recognition bindings are packaged in replaceable cartridges, while a shared operator library and execution mechanisms instantiate the corresponding recognition programs.

3 Method

GameConsole separates game-specific recognition resources and bindings from reusable recognition operations. A knowledge cartridge provides the entities, visual references, relations, and bindings for a game, while the shared console instantiates and runs the corresponding recognition programs from a common operator library. The resulting source-linked facts are combined with time-aligned visual evidence to produce event descriptions (Figure 2). Replacing the cartridge therefore changes the game-specific recognition configuration while retaining the shared operator library and execution mechanisms.

3.1 Problem Formulation and Adaptation Setting

Let $\mathfrak{G}$ denote the set of supported games. For each game $g \in \mathfrak{G}$, let $V = \{(x_q, t_q)\}_{q=1}^{Q}$ denote a gameplay video, where x_q is the q-th source frame and t_q is its timestamp. Let D_g denote the game-specific knowledge cartridge, which contains knowledge resources, recognition bindings, optional specialist weights, and temporal and description configurations. A task request u specifies the recognition targets $\mathcal{T}_u$, description instructions, and event scope.

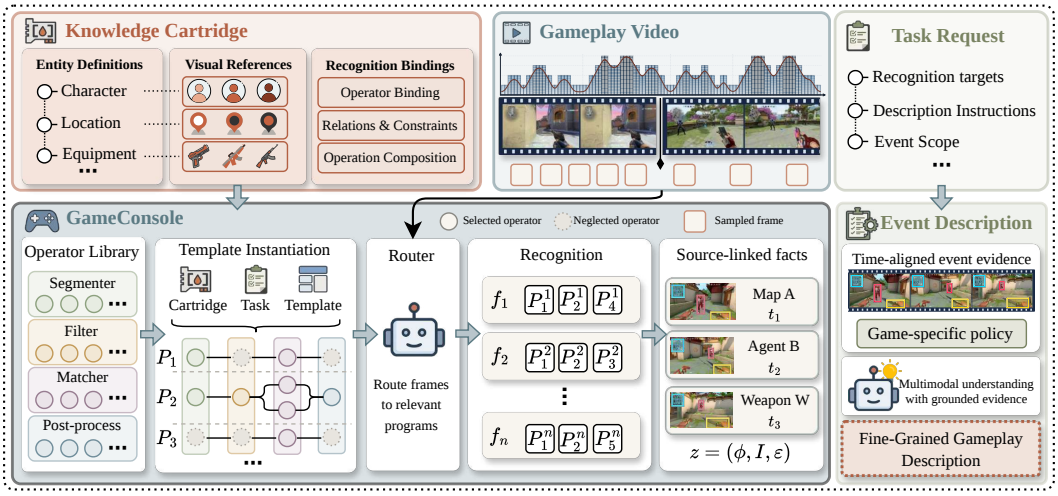

Figure 2: GAMECONSOLE separates a game-specific knowledge cartridge from a shared recognition console. The cartridge specifies the resources and bindings used to recognize each supported target, while the task request specifies what the current run should recognize and describe. The console instantiates the corresponding recognition programs from a shared operator library and routes sampled frames to applicable programs. The resulting source-linked facts are combined with time-aligned visual observations for fine-grained gameplay description.

We formulate fine-grained gameplay description as a cross-game structured prediction problem. Given (V, D_g, u), the goal is to predict a set of temporally localized event descriptions:

$$\hat{\mathcal{Y}}_g = \mathcal{F}_{\mathcal{C}}(V, D_g, u) = \{(h_i, y_i)\}_{i=1}^{M}, \qquad h_i = [s_i, e_i], \tag{1}$$

where h_i is an event window whose boundaries s_i and e_i lie on the source-video time axis, and y_i describes the observable event within that interval using the requested game-defined entities, states, and participant roles. Participant roles denote observable associations between entities and the actions or states attributed to them; attributes that cannot be resolved from the available evidence may remain unspecified.

For each predicted event, GAMECONSOLE additionally maintains supporting evidence $\mathcal{E}_i$, consisting of temporally aligned visual observations and source-linked recognition facts. This evidence is used to ground the generated description but is not itself the prediction target.

The adaptation constraint is that the recognition console $\mathcal{C}$, including the operator interfaces and execution mechanisms, is shared across games, whereas the cartridge D_g is game-specific. Adapting to a new game therefore amounts to replacing D_g while keeping $\mathcal{C}$ fixed. Appendix B.6 specifies the corresponding adaptation boundary.

3.2 CARTRIDGE-CONDITIONED RECOGNITION

Cartridge Specification For each game g, the cartridge $D_g = (K_g, \Pi_g, W_g, \theta_g, \rho_g)$ contains game knowledge K_g, target-specific recognition bindings $\Pi_g = \{\pi_k\}$, optional specialist weights W_g, a temporal profile θ_g, and a game-specific policy ρ_g. The task request specifies *what* should be recognized and described, while each binding specifies *how* a supported target uses the resources available for game g. The shared operator library provides pretrained visual operators, while W_g supplies weights for optional game-specific specialists. Bindings specify operator assignments and composition rules; θ_g configures visual-change signals, boundary selection, and adaptive sampling, while ρ_g specifies terminology, evidence retention, and output rules.

Shared Templates and Instantiation We distinguish three levels of recognition abstraction. An *operator* is a concrete operation, such as region extraction, visual matching, or structured state readout. A *template* defines how operators can be connected and which stages may be omitted. A

program is an instantiated recognition procedure whose operators and resources are determined by the current cartridge.

The shared operator library contains four component families corresponding to successive transformations of the available evidence (Figure 2). *Segmentation* produces candidate regions or objects from an observation; *filtering* removes candidates that do not satisfy the configured conditions; *matching* resolves the remaining candidates against cartridge-defined entities and reference assets; and *post-processing* verifies accepted results or reads structured states. The segmentation stage can also use detectors, fixed regions of interest, or whole-frame inputs. These four stages define a common interface rather than a mandatory execution path. Templates are typed: adjacent stages exchange predefined object types, such as candidate regions, scored candidates, and fact records, allowing the console to verify that the operators selected by a binding are compatible.

For each requested target $k \in \mathcal{T}_u$, the corresponding binding $\pi_k \in \Pi_g$ specifies the operator assignments, reference resources, screen regions, parameters, and composition rules used for recognition. The console resolves these assignments against the shared operator library and binds the selected resources to the corresponding templates:

$$\mathcal{P}_{g,u} = \mathrm{Instantiate}_{\mathcal{C}}(D_g, \mathcal{T}_u). \tag{2}$$

Equation 2 therefore instantiates recognition programs by selecting registered operators and binding cartridge resources to them. It neither trains new modules nor generates arbitrary code. A binding may also depend on outputs from other recognition programs; for example, scene-location matching requires an accepted map prediction to restrict its candidate reference bank.

Knowledge-Conditioned Recognition Cartridge bindings determine not only which visual references and operators are used for each target, but also how recognition evidence is combined and constrained. Within a single target, *parallel evidence fusion* combines complementary matching scores over the same candidate set, while *sequential candidate refinement* uses an initial matcher to retain a smaller set of plausible candidates for subsequent refinement. Bindings can also exploit game-defined relations across dependent targets. In this case, an accepted prediction for one target restricts the valid candidates of another before visual matching. This differs from sequential refinement: the candidate set is constrained by a recognized semantic entity and an explicit game-defined relation rather than by matching scores for the same target. These composition and conditioning rules are specified by the cartridge bindings and executed using reusable operators from the shared library.

For example, Valorant location recognition uses the recognized map to restrict the location reference bank before visual matching. Let B_{loc} denote the location reference bank, and let $\mathrm{map}(r)$ denote the map associated with reference r. Given an accepted map prediction $\hat{m}$, the admissible references are

$$B_{\mathrm{loc}}(\hat{m}) = \{r \in B_{\mathrm{loc}} : \mathrm{map}(r) = \hat{m}\}. \tag{3}$$

A registered matching operator then compares the query only with references in $B_{\mathrm{loc}}(\hat{m})$; in the Valorant cartridge, this step uses LightGlue (Lindenberger et al., 2023) for local feature matching. The cartridge provides the map–location relation and reference assets, while the shared console applies the candidate restriction and matching operations. If the map prediction cannot be resolved, the program follows its configured fallback or abstains. Appendix B provides the formal definitions of these composition and conditioning rules.

Execution and Source-Linked Facts For each observation, the Router evaluates the scene, interface, and region conditions specified by the cartridge bindings and selects the applicable recognition programs. Program execution then follows the configured dependencies and acceptance conditions, and unresolved predictions are omitted or marked unknown according to the binding policy. Each accepted prediction is recorded as a source-linked fact:

$$z = (\phi, I, \epsilon), \tag{4}$$

where ϕ denotes an assertion about a game-defined entity or observable state, I denotes the source time or interval supporting the assertion, and ϵ records the associated visual evidence. Facts supported by different evidence are kept separate; for example, an entity identity does not by itself establish its state or role in an event. Likewise, evidence from a single observation is not assumed to persist throughout the entire event window. Each fact retains its original source timestamp, allowing the subsequent description stage to align recognized entities and states with temporally ordered visual observations. Appendix B provides the detailed routing conditions and execution contracts.

3.3 Video Evidence and Event Description

The shared temporal front end uses the cartridge's temporal profile θ_g to localize candidate event windows from visual-change signals and select the observations used to describe them. It combines these signals into a change-energy sequence and selects salient local changes as candidate temporal boundaries. Consecutive boundaries define event windows for subsequent observation selection. Appendix C provides the detailed signal calculation.

Within each event window, the sampling rate adapts to the local visual-change energy. Let $r_{\max}$ denote the maximum sampling rate and N the number of sampling levels. The available sampling rates and the energy-conditioned sampling level are

$$r^{(n)} = \frac{n}{N} r_{\max}, \qquad \ell(t) = 1 + \sum_{n=1}^{N-1} \mathbb{I}[E(t) \geq \tau_n], \qquad r(t) = r^{(\ell(t))}, \tag{5}$$

where $E(t)$ denotes the local change energy at source timestamp t, and $\tau_1 < \cdots < \tau_{N-1}$ are the thresholds between adjacent sampling levels. Visually stable periods are sampled more sparsely, while higher change energy increases the sampling rate around transitions. Sparse observations preserve context for persistent entities and states, whereas denser observations capture their changes over time. This allocation reduces redundant processing of visually similar frames while retaining more observations around informative transitions. Appendix C provides the complete sampling controller and observation-selection details.

Let A denote the selected observations. Executing the applicable programs in $\mathcal{P}_{g,u}$ on A produces the source-linked fact set Z. For each event window h_i, the cartridge's game-specific policy ρ_g selects the relevant observations A_i and temporally applicable facts Z_i. Evidence assembly preserves the temporal order of observations and the source links of recognized facts:

$$\mathcal{E}_i = \mathrm{Assemble}(h_i, A_i, Z_i), \qquad y_i = \mathrm{VLM}(\mathcal{E}_i; u, \rho_g). \tag{6}$$

The visual observations provide temporal context for actions and state changes, while the recognition facts provide game-specific identities, states, and relations. The task request u specifies the requested content, while the cartridge policy ρ_g governs terminology, evidence retention, and output rules.

4 Experiments

We first evaluate fine-grained description quality under shared video observations and then examine target-specific recognition choices. Temporal-front-end experiments assess observation selection and coarse transition segmentation, while downstream QA evaluates the utility of the generated descriptions.

4.1 Experimental Setup

We evaluate 3,712 reference events from 121 Valorant and Valorant: Mobile videos, 347 events from Don't Starve, and 959 events from Hollow Knight: Silksong. The three games cover distinct recognition requirements: combat identities and locations, inventory and world entities, and visually similar entities and areas, respectively. Dataset statistics and configurations are provided in Appendix D. We construct four task-level evaluation sets, collectively referred to as GCBench: GCBench-Desc for description quality, GCBench-Sample for observation selection, GCBench-Seg for event boundaries, and GCBench-QA for downstream question answering; GCBench-Prog separately evaluates individual recognition programs. Table 13 summarizes their scale and metrics, and Appendix D describes the construction and review procedure for each. The description comparison uses shared event windows, with reference descriptions reviewed and aligned to these windows. Appendix D.1 provides additional background on the three games, and the corresponding dataset construction and alignment details.

All description conditions use the same event windows and sampled video observations produced by our temporal front end, together with the same game name, task request, and output schema. Within each generator, we compare textual knowledge, visual-reference retrieval, and recognition facts obtained from these observations. Direct receives no reference images, entity definitions, or

Table 1: Fine-grained gameplay description on GCBench-Desc across three games and three generators. Bold indicates the highest value in each metric column, including ties.

Method	Model	Valorant / Mobile				Don't Starve				Silksong			
		P ↑	R ↑	F1 ↑	Evid. ↑	P ↑	R ↑	F1 ↑	Evid. ↑	P ↑	R ↑	F1 ↑	Evid. ↑
Direct	gpt5-chat	58.1	33.8	41.2	2.73	44.7	61.1	50.6	2.92	42.2	25.1	29.7	2.66
	gpt-5.6-terra	58.0	31.8	39.6	2.83	48.2	56.1	50.7	2.65	39.8	23.4	27.6	2.68
	glm-5.3-flash	52.2	32.2	38.1	2.51	44.9	54.0	47.9	2.70	44.7	28.5	32.7	2.69
Text KB	gpt5-chat	58.4	45.7	49.9	2.71	44.5	69.7	53.2	3.03	43.7	28.0	31.9	2.75
	gpt-5.6-terra	57.4	42.5	47.5	2.80	45.9	61.9	51.6	2.83	43.0	26.9	31.0	2.68
	glm-5.3-flash	49.8	42.5	44.5	2.50	43.2	58.3	48.3	2.73	47.4	34.8	37.6	2.73
MM-RAG	gpt5-chat	58.8	35.8	43.0	2.79	47.3	64.2	53.3	2.99	44.2	26.8	31.3	2.70
	gpt-5.6-terra	57.7	34.5	41.7	2.86	49.2	57.0	51.5	2.80	46.6	27.5	32.3	2.74
	glm-5.3-flash	50.0	34.7	39.4	2.54	45.4	55.6	48.7	2.77	53.2	36.0	40.3	2.77
MM-RAG+	gpt5-chat	61.6	39.4	46.4	2.81	52.5	74.1	60.4	3.29	46.0	28.0	32.7	2.75
	gpt-5.6-terra	60.3	37.5	44.7	2.90	53.5	65.7	58.0	2.94	47.6	27.9	32.9	2.75
	glm-5.3-flash	53.0	38.8	43.1	2.56	49.3	63.8	54.5	2.99	57.4	39.3	43.6	2.81
GAMECONSOLE	gpt5-chat	**70.9**	**78.5**	**73.3**	**3.55**	53.2	**78.1**	62.0	**3.45**	65.6	**76.3**	**67.7**	**3.52**
	gpt-5.6-terra	70.0	66.9	66.8	3.46	**63.3**	66.5	**63.8**	3.12	**69.4**	70.4	66.7	3.24
	glm-5.3-flash	61.9	71.5	65.0	3.22	56.5	60.2	57.1	3.23	66.8	74.4	67.4	3.27

recognition outputs, measuring what a strong video LLM can recover from the video and its pretrained knowledge. Text KB adds category text such as agent and ability definitions, but no visual references, testing whether textual knowledge is sufficient. MM-RAG crops the same regions of interest, retrieves Top-K images from the cartridge reference bank using a generic visual embedding, and asks the VLM to select a canonical entity or none, without game-specific recognition programs. MM-RAG+ is an oracle-candidate diagnostic that places the ground-truth-matched reference at the top of the candidate list presented to the VLM. Each condition uses gpt5-chat (OpenAI, 2025), gpt-5.6-terra (OpenAI, 2026), and glm-5.3-flash (GLM-5 Team, 2026).

We measure description quality by matching canonical mentions against reference annotations and report event-averaged precision, recall, and F1 (%). Game-specific rules include an exemption for additional Valorant character mentions from false-positive counts (Table 14). We also report Evid., a 1–5 reference-conditioned rating produced by DeepSeek-V4-Flash (DeepSeek-AI, 2026) following rubric-based LLM evaluation (Zheng et al., 2023; Liu et al., 2023). The rubric considers semantic consistency in interactions, states, and event order, together with the coverage and specificity of reference details. The temporal front end is evaluated on observation selection and human-reviewed ground-truth boundary clusters, while downstream QA is scored by key-point accuracy (KP Acc.), a 1–5 Overall rating from Kimi-K2.6 (Moonshot AI, 2026), and interval mIoU. Judges see only the reference and prediction, without method identities. Metric definitions, judge rubrics, prompts, and review configurations are provided in Appendix D, Appendix E.2, and Appendix E.3.

4.2 FINE-GRAINED DESCRIPTION QUALITY

Text KB improves F1 over Direct in all nine game–generator combinations, whereas generic visual retrieval does not consistently outperform textual knowledge: on Valorant, Text KB remains the strongest baseline for every generator. MM-RAG+ improves over MM-RAG in F1 across all tested combinations, while GAMECONSOLE achieves the highest F1 and Evid. in every game–generator combination. Textual definitions provide entity names and semantics, and retrieved references provide visual candidates; GAMECONSOLE further uses recognition bindings to turn these resources into target-specific recognition procedures and source-linked facts. These comparisons suggest that incorporating game knowledge into recognition is more effective than providing textual or retrieved visual context alone. The gains over Direct are smaller on Don't Starve, where Direct starts from a higher F1 baseline, and larger on Silksong, where direct generation recovers fewer game-specific details. This pattern is consistent with Figure 1(b). Category-level results in Appendix E.1 show improvements across several types of detail, including locations, abilities, and state values.

4.3 RECOGNITION CHOICES

Table 2 reports GCBench-Prog results for three representative recognition programs (P_2, P_3, and P_5), evaluated on human-annotated frames from held-out videos. The relative performance of the compared components differs across the evaluated targets. The Specialist column includes a class-specific identity detector for characters (84.87) and a multi-feature matcher on frozen ability detections (77.06). No dedicated specialist is used for locations, where LightGlue achieves the highest score among the compared generic matchers (63.2). For ability recognition, however, LightGlue scores only 10.10, substantially below the specialist. These comparisons support target-specific component selection for the evaluated recognition tasks. In GAMECONSOLE, cartridge bindings specify these operator choices and their associated resources within the shared library.

Table 2: Target-specific recognition accuracy (Acc., %). Values are comparable only within a row; Appendix D.1 describes the shared inputs and evaluation setup for each target.

Target	Embedding	LightGlue	Specialist
Character (P_2)	30.79	62.81	**84.87**
Ability (P_3)	43.25	10.10	**77.06**
Location (P_5)	24.4	**63.2**	—

4.4 TEMPORAL FRONT END

Table 3: Sampling quality and efficiency on GCBench-Sample with gpt5-chat. Mean averages the five category F1 scores; efficiency measures are per-event means. Bold marks column maxima for quality and minima for efficiency.

Sampling	Description Quality: F1 (%) ↑						Efficiency ↓		
	Char.	Wea.	Abil.	Map	Loc.	Mean	Frames	PT	Latency
Uniform 1 s	60.2	**94.0**	**66.2**	**90.0**	**68.6**	**75.80**	18.1	34,300	64.8
Uniform 2 s	49.3	84.2	51.2	87.8	63.7	67.24	9.3	20,429	43.4
Uniform 3 s	50.7	86.0	53.1	85.6	52.9	65.66	6.7	16,161	35.4
Uniform 4 s	56.0	83.3	45.2	81.1	39.6	61.04	5.1	13,475	32.9
Uniform 8 s	47.0	76.8	37.6	62.2	27.2	50.16	**3.1**	**10,099**	**23.9**
Ours	**60.6**	85.9	62.1	87.8	59.9	71.26	5.1	14,219	31.6

With event boundaries and generation settings fixed, Table 3 compares our adaptive sampler with uniform sampling on GCBench-Sample. At the same mean frame count as Uniform 4 s, our sampler improves all five categories and increases mean F1 by 10.2 points, with only 5.5% more prompt tokens and slightly lower measured latency; the largest gains are in ability and location. It also outperforms Uniform 2 s and 3 s while using fewer frames, fewer tokens, and less time, showing that the gains do not simply result from processing more observations. Uniform 1 s achieves 4.5 points higher mean F1, but requires 2.4 times the prompt tokens and 2.1 times the latency. Our adaptive sampler therefore retains most of its description quality with substantially fewer observations. Human review of information loss relative to Uniform 1 s shows a similar tradeoff (Appendix E.2).

Table 4: Cluster-level detection of reviewed round transitions and hard viewpoint cuts on GCBench-Seg. All methods use shared short-segment merging.

Method / configuration	P (%)	R (%)	F1 (%)	Boundaries/min
PySceneDetect ($t = 27$)	43.7	83.5	57.4	21.2
KTS ($v_{\max} = 0.25$)	42.8	86.3	57.2	14.09
C-DPCKNN ($\tau = 10$)	50.0	31.9	39.0	5.15
EfficientGEBD ($t = 0.35$)	38.6	4.4	7.8	0.76
GAMECONSOLE	71.1	72.8	72.0	7.06

We evaluate temporal segmentation on reviewed round transitions and hard viewpoint cuts in GCBench-Seg, comparing our temporal front end with PySceneDetect (PySceneDetect, 2026), KTS (Potapov et al., 2014), C-DPCKNN as used in VideoINSTA (Liao et al., 2024), and EfficientGEBD (Zheng et al., 2024). At the reported configurations, GAMECONSOLE achieves 72.0% cluster-level F1, compared with 57.4% for PySceneDetect, the highest baseline value, while producing 7.06 boundaries per minute. PySceneDetect and KTS produce denser boundaries, whereas C-DPCKNN and EfficientGEBD produce fewer under these settings. The ground-truth boundary set

is finalized through manual review of the source videos using round transitions and hard viewpoint cuts as the acceptance criteria. These results evaluate whether the temporal front end identifies suitable coarse event boundaries for description. Figure 10 provides a time-aligned example, while Table 4 and Appendix E.3 report the full settings, ground-truth review protocol, and diagnostics.

4.5 DOWNSTREAM GAMEPLAY QA

Table 5: Downstream QA on GCBench-QA across three tasks. Video only uses no auxiliary descriptions. KP Acc., mIoU, Hit@0.5, and coverage scores are in %; Overall is a 1–5 rating. Bold marks column maxima, including ties.

Method	Captioner	Post-match review		Event explanation		Temporal retrieval			
		KP Acc.	Overall	KP Acc.	Overall	mIoU	Hit@0.5	Cov. R	Cov. P
Video only		22.6	2.3	15.3	1.7	8.3	0.0	26.0	18.0
Direct	gpt5-chat	31.1	2.9	42.9	2.5	36.3	34.0	64.0	55.0
	gpt-5.6-terra	30.8	2.8	43.9	2.5	34.6	31.0	59.0	54.0
Text KB	gpt5-chat	32.4	2.8	41.2	2.5	37.0	27.0	59.0	63.0
	gpt-5.6-terra	32.0	2.9	40.0	2.5	38.3	37.0	60.0	59.0
MM-RAG	gpt5-chat	28.3	2.7	45.8	2.6	36.4	33.0	64.0	55.0
	gpt-5.6-terra	33.6	2.9	46.4	2.7	34.9	33.0	56.0	57.0
MM-RAG+	gpt5-chat	32.0	2.9	47.2	2.7	36.3	33.0	64.0	61.0
	gpt-5.6-terra	33.7	2.9	47.7	2.7	34.6	32.0	58.0	57.0
GAMECONSOLE	gpt5-chat	**38.3**	**3.2**	**67.8**	**3.2**	**55.6**	**62.0**	**79.0**	**73.0**
	gpt-5.6-terra	37.0	**3.2**	62.9	3.1	44.7	46.0	68.0	71.0

Beyond description quality, we evaluate whether the generated fine-grained descriptions support downstream gameplay QA on the same video. Following video-understanding benchmarks that assess comprehension and temporal localization through question answering (Qian et al., 2024; Li et al., 2024; Fu et al., 2025), and gameplay QA benchmarks that target game-defined states and participant attribution (Mun et al., 2017; Wang et al., 2026), we evaluate three post-game information needs—post-match review, event explanation, and temporal retrieval—on GCBench-QA. Questions, video input, and the answering model are held fixed, while only the auxiliary descriptions vary. We reuse the gpt5-chat and gpt-5.6-terra descriptions reported in the main description experiment. Table 5 compares video-only input with five description conditions under these two generators. Relative to the strongest baseline using the same generator, GAMECONSOLE improves KP Acc. by 3.3–5.9 points on post-match review and by 15.2–20.6 points on event explanation, with higher Overall scores in both tasks. For gpt5-chat, location key-point accuracy increases from 4% with Direct to 44% with GAMECONSOLE, consistent with the improved location coverage reported in Appendix E.1. Temporal mIoU reaches 55.6% and 44.7% for gpt5-chat and gpt-5.6-terra, exceeding the corresponding baselines by 18.6 and 6.4 points. These results show that the generated descriptions improve both recovery of game-specific details and retrieval of relevant video intervals. Appendix D.3 provides the task definitions, metrics, and judge protocol.

5 CONCLUSION

We presented GAMECONSOLE, a console–cartridge framework that separates game-specific resources and recognition bindings from reusable recognition operations. Across three games and three generation backends, GAMECONSOLE improves fine-grained description quality under shared video observations. Target-level results demonstrate the need for recognition choices tailored to different entities and states, while downstream QA on Valorant shows that the resulting descriptions support post-match review, event explanation, and temporal retrieval. Category-level results further highlight the importance of evaluating detail coverage together with semantic consistency and grounding in source evidence.

AI USE STATEMENT

Generative AI tools were used for language editing and literature search, and in the data-construction and evaluation workflows described in Appendices B.5 and D. These uses included prompt refinement and proposal filtering for specialist annotations, drafting descriptions for human review, assisting QA-item construction, and automated evaluation. Human review procedures and the scope of model-based evaluation are described in the corresponding sections. The authors take responsibility for the final manuscript, data, implementation, and reported results.

REFERENCES

Pablo Fernández Alcantarilla, Jesús Nuevo, Adrien Bartoli, et al. Fast explicit diffusion for accelerated features in nonlinear scale spaces. In *Bmvc*, volume 2, pp. 8, 2013.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. In *International conference on learning representations*, volume 2026, pp. 138846–138923, 2026.

Rohan Choudhury, Koichiro Niinuma, Kris M Kitani, and László A Jeni. Zero-shot video question answering with procedural programs. *arXiv preprint arXiv:2312.00937*, 2023.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *2018 IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW)*, pp. 337–33712. IEEE, 2018.

Yue Fan, Xiaojian Ma, Rujie Wu, Yuntao Du, Jiaqi Li, Zhi Gao, and Qing Li. Videoagent: A memory-augmented multimodal agent for video understanding. In *European Conference on Computer Vision*, pp. 75–92. Springer, 2024.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24108–24118. IEEE, 2025.

GLM-5 Team. GLM-5: From vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14953–14962. IEEE, 2023.

Joy Hsu, Jiayuan Mao, Josh Tenenbaum, and Jiajun Wu. What's left? concept grounding with logic-enhanced foundation models. *Advances in Neural Information Processing Systems*, 36: 38798–38814, 2023.

Tatsuya Ishigaki, Goran Topić, Yumi Hamazono, Hiroshi Noji, Ichiro Kobayashi, Yusuke Miyao, and Hiroya Takamura. Generating racing game commentary from vision, language, and structured data. In *Proceedings of the 14th International Conference on Natural Language Generation*, pp. 103–113, 2021.

Wenhui Jiang, Yibo Cheng, Linxin Liu, Yuming Fang, Yuxin Peng, and Yang Liu. Comprehensive visual grounding for video description. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 2552–2560, 2024.

Zeyu Jin, Xiaoyu Qin, Songtao Zhou, Kaifeng Yun, and Jia Jia. Towards automatic soccer commentary generation with knowledge-enhanced visual reasoning, 2026.

Glenn Jocher and Jing Qiu. Ultralytics YOLO11. Software, 2024. URL https://github.com/ultralytics/ultralytics.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *Proceedings of the IEEE international conference on computer vision*, pp. 706–715, 2017.

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, et al. Mvbench: A comprehensive multi-modal video understanding benchmark. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 22195–22206, 2024.

Ruotong Liao, Max Erler, Huiyu Wang, Guangyao Zhai, Gengyuan Zhang, Yunpu Ma, and Volker Tresp. Videoinsta: Zero-shot long video understanding via informative spatial-temporal reasoning with llms. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 6577–6602, 2024.

Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. Lightglue: Local feature matching at light speed, 2023.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 2511–2522, 2023.

Yongdong Luo, Xiawu Zheng, Guilin Li, Shukang Yin, Haojia Lin, Chaoyou Fu, Jinfa Huang, Jiayi Ji, Fei Chao, Jiebo Luo, et al. Video-rag: Visually-aligned retrieval-augmented long video comprehension. *Advances in Neural Information Processing Systems*, 38:168008–168033, 2026.

Jianzhe Ma, Zhonghao Cao, Shangkui Chen, Yichen Xu, Wenxuan Wang, and Qin Jin. Egoesport-sqa: An egocentric video benchmark for perception and reasoning in esports. *arXiv preprint arXiv:2604.12320*, 2026.

Juhong Min, Shyamal Buch, Arsha Nagrani, Minsu Cho, and Cordelia Schmid. Morevqa: Exploring modular reasoning models for video question answering. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13235–13245. IEEE, 2024.

Moonshot AI. Kimi-K2.6 model card, 2026. URL `https://huggingface.co/moonshotai/Kimi-K2.6`.

Jonghwan Mun, Paul Hongsuck Seo, Ilchae Jung, and Bohyung Han. Marioqa: Answering questions by watching gameplay videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2867–2875, 2017.

OpenAI. GPT-5 system card, 2025. URL `https://openai.com/index/gpt-5-system-card/`.

OpenAI. GPT-5.6 system card, 2026. URL `https://deploymentsafety.openai.com/gpt-5-6`.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Danila Potapov, Matthijs Douze, Zaid Harchaoui, and Cordelia Schmid. Category-specific video summarization. In *European conference on computer vision*, pp. 540–555. Springer, 2014.

PySceneDetect. PySceneDetect: Video scene detection software, 2026. URL `https://www.scenedetect.com/`. Project website, accessed September 14, 2026.

Ji Qi, Jifan Yu, Teng Tu, Kunyu Gao, Yifan Xu, Xinyu Guan, Xiaozhi Wang, Bin Xu, Lei Hou, Juanzi Li, et al. Goal: A challenging knowledge-grounded video captioning benchmark for real-time soccer commentary generation. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pp. 5391–5395, 2023.

Long Qian, Juncheng Li, Yu Wu, Yaobo Ye, Hao Fei, Tat-Seng Chua, Yueting Zhuang, and Siliang Tang. Momentor: Advancing video large language model with fine-grained temporal reasoning. *arXiv preprint arXiv:2402.11435*, 2024.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11854–11864. IEEE, 2023.

Teng Wang, Ruimao Zhang, Zhichao Lu, Feng Zheng, Ran Cheng, and Ping Luo. End-to-end dense video captioning with parallel decoding. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6827–6837. IEEE, 2021.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 13484–13508, 2023.

Yunzhe Wang, Runhui Xu, Kexin Zheng, Tianyi Zhang, Jayavibhav Niranjan Kogundi, Soham Hans, and Volkan Ustun. Gameplayqa: A benchmarking framework for decision-dense pov-synced multi-video understanding of 3d virtual agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34333–34355, 2026.

Zeyu Xi, Ge Shi, Xuefen Li, Junchi Yan, Zun Li, Lifang Wu, Zilin Liu, and Liang Wang. A simple yet effective knowledge guided method for entity-aware video captioning on a basketball benchmark. *Neurocomputing*, 619:129177, 2025.

Antoine Yang, Arsha Nagrani, Paul Hongsuck Seo, Antoine Miech, Jordi Pont-Tuset, Ivan Laptev, Josef Sivic, and Cordelia Schmid. Vid2seq: Large-scale pretraining of a visual language model for dense video captioning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10714–10726. IEEE, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Ziwei Zheng, Zechuan Zhang, Yulin Wang, Shiji Song, Gao Huang, and Le Yang. Rethinking the architecture design for efficient generic event boundary detection. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 1215–1224, 2024.

Luowei Zhou, Yannis Kalantidis, Xinlei Chen, Jason J Corso, and Marcus Rohrbach. Grounded video description. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6571–6580. IEEE, 2019.

APPENDIX CONTENTS

Section		Page
A	**Task Illustration.** Action-level description contrasted with the fine-grained description studied in this work.	14
B	**Recognition and Cartridge Implementation.** How a cartridge configures executable recognition programs.	15
B.1	Bindings, contracts, and routing: program tuple, stage contracts, and applicability conditions.	15
B.2	Recognition composition: parallel evidence fusion, sequential candidate refinement, and relation conditioning.	15
B.3	Program bindings and shared settings: the program index and shared operator parameters.	16
B.4	Router signals and prompts used to determine program applicability.	16
B.5	Declared regions and specialist training: fixed screen boxes and specialist dataset construction.	19
B.6	Adaptation boundary: what a cartridge may change without extending the shared framework.	21
C	**Temporal Signals and Observation Selection.** Change-energy fusion and the adaptive sampling controller.	21
D	**Benchmark and Evaluation Protocols.** What is measured, on which data, and with which judge.	23
D.1	Datasets and configurations: reference pools, human review, generation backends, and GCBench splits.	23
D.2	Description evaluation: mention-level P/R/F1 and the reference-conditioned Evid. rubric and prompt.	25
D.3	Downstream QA protocol: the three tasks, item construction, metrics, and judge prompts.	28
E	**Additional Experimental Results.** Per-category, sampling, and temporal-segmentation results.	30
E.1	Category coverage and per-category description results for the three games.	30
E.2	Sampling quality and cost, including the human information-loss review.	35
E.3	Temporal segmentation: settings, ground-truth review, a time-aligned example, and diagnostics.	35

A TASK ILLUSTRATION

Figure 3 illustrates the task introduced in Section 1. It contrasts an action-level description with a fine-grained description of the same segment, highlighting the game-specific identities, equipment, and locations needed for a more informative account.

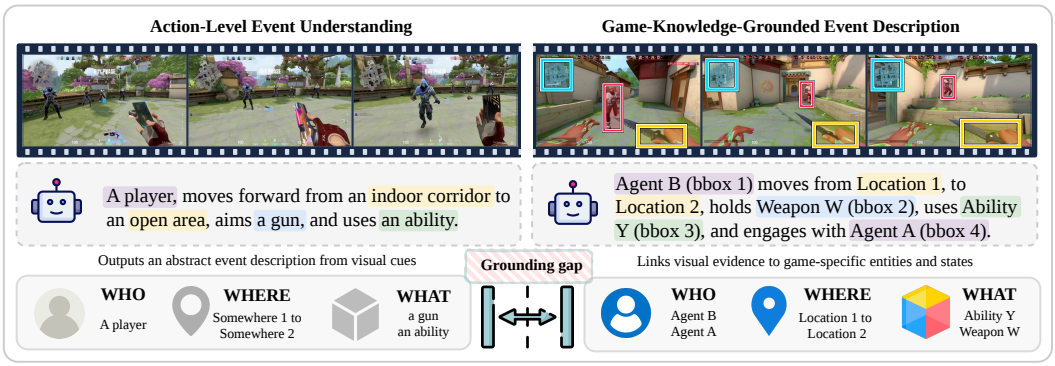

Figure 3: From action-level descriptions to game-specific details. The panels contrast a broad action-level description with a fine-grained description that identifies game-specific entities, equipment, and locations from visual observations.

B Recognition and Cartridge Implementation

B.1 Bindings, Contracts, and Routing

Following Section 3.2, the cartridge is denoted by $D_g = (K_g, \Pi_g, W_g, \theta_g, \rho_g)$, where K_g contains canonical entity definitions $\mathcal{O}_g$, visual reference assets $\mathcal{B}_g$, and game-defined relations $\mathcal{R}_g$. The target-specific recognition bindings $\Pi_g = \{\pi_k\}$ specify operator assignments, reference resources, screen regions, parameters, composition rules, and execution conditions. Optional game-specific specialist weights are collected in W_g. The temporal profile θ_g configures visual-change signals, boundary selection, and adaptive sampling. The description policy ρ_g specifies terminology, evidence retention, and output rules. The task request u specifies the requested targets $\mathcal{T}_u$, description instructions and output schema, and event scope.

A recognition program binds a screen region Ω_k, four operator slots, a reference bank B_k, an output schema O_k, and parameters η_k:

$$P_k = (\Omega_k, S_k, F_k, M_k, Q_k, B_k, O_k, \eta_k). \tag{7}$$

Table 6 summarizes the stage contracts. Unused stages may be skipped, and a terminal detector or structured readout may emit a fact directly. A complete fact record stores $(I, o, \vartheta, \alpha, c, \epsilon)$, where I is the supporting time or interval, o is the recognized entity, ϑ is the fact type, α contains its attributes, c is the component-level confidence, and ϵ records the source evidence. This refines the main-text representation $z = (\phi, I, \epsilon)$ by setting $\phi = (o, \vartheta, \alpha)$. Source evidence records the supporting frame, timestamp, region, recognition program, and matched references when applicable.

Table 6: Typed recognition stages. A program can omit a stage or terminate when its output satisfies the fact contract.

Stage	Input → output	Representative operations
Segmentation S_k	Image/context → candidate regions	Detector, promptable segmenter, fixed ROI, full-frame view
Filter F_k	Candidate regions → retained regions	Score, geometry, size, deduplication, slot constraints
Match M_k	Regions and references → scored entities	Embeddings, local correspondence, correlation, composed matchers
Post Q_k	Scores/context → source-linked facts	Margin gates, unique assignment, VLM verification, structured readout

For a selected observation $a = (x_q, t_q)$, the applicability predicate $\chi_k(a; \pi_k)$ evaluates the scene, interface, and region conditions specified by the corresponding binding. The routed jobs are

$$\mathcal{J} = \{(a, k) : a \in A,\ P_k \in \mathcal{P}_{g,u},\ \chi_k(a; \pi_k) = 1\}. \tag{8}$$

Independent jobs may execute asynchronously under resource-specific limits. Execution follows the configured dependencies, acceptance conditions, and fallback rules. Accepted outputs retain their observed temporal scope, while unresolved outputs are omitted or marked unknown according to the binding policy.

B.2 Recognition Composition

Parallel Evidence Fusion For matcher j, let $B_{k,j} \subseteq B_k$ denote the candidate subset scored by that matcher, and let μ_j and σ_j denote the mean and standard deviation of its scores for the current query. We normalize the matcher scores and combine them as

$$\widetilde{s}_j(o) = \begin{cases} (s_j(o) - \mu_j)/(\sigma_j + \varepsilon), & \sigma_j > \varepsilon, \\ 0, & \text{otherwise}, \end{cases} \qquad s(o) = \frac{\sum_{j: o \in B_{k,j}} w_j \widetilde{s}_j(o)}{\sum_{j: o \in B_{k,j}} w_j}. \tag{9}$$

Only matchers that score candidate o contribute to its fused score. The highest-scoring entity is accepted only when the score and margin conditions specified by the binding are satisfied.

Sequential Candidate Refinement and Relation Conditioning Sequential candidate refinement uses an initial matcher to retain a smaller set of plausible candidates,

$$C_1 = \mathrm{TopK}_{o \in B_k} M_1(x, o), \qquad \hat{o} = \arg\max_{o \in C_1} M_2(x, o), \tag{10}$$

where both matchers resolve the same recognition target. Relation conditioning instead uses an accepted prediction for one target together with a game-defined relation to restrict the candidate bank of a dependent target. Given a conditioning entity $\hat{p}$, the admissible candidates are

$$B_k(\hat{p}) = \{o \in B_k : (\hat{p}, o) \in \mathcal{R}_g\}. \tag{11}$$

Unlike sequential refinement, which narrows candidates using matching scores for the same target, relation conditioning restricts the candidate space using a recognized semantic entity and an explicit relation stored in the cartridge.

For Valorant location recognition, the accepted map prediction $\hat{m}$ defines $B_{\mathrm{loc}}(\hat{m})$ in Equation 3. The location program then selects

$$\hat{r} = \arg\max_{r \in B_{\mathrm{loc}}(\hat{m})} s_{\mathrm{geo}}(x, r), \tag{12}$$

where s_{geo} is the geometric matching score. The cartridge provides the map–location relation and reference assets, while the shared console executes the candidate restriction and matching operations. If the conditioning prediction cannot be resolved, the program follows the fallback rule specified by the binding or abstains.

B.3 Program Bindings and Shared Settings

Table 7 summarizes P_1–P_{14} across the three games. The accepted map from P_4 restricts the reference bank *before* location matching in P_5. Structured HUD reading supplies additional state records outside this numbered list. Shared operator settings and the component-study exceptions are summarized in Table 8.

Table 7: Recognition programs instantiated by the three cartridges. Reuse denotes shared operators and composition patterns. The map prediction from P_4 is a prerequisite for the conditioned matcher in P_5.

ID	Game	Recognition target	Operator binding
P_1	Valorant	HUD roster icon	SAM3, geometry filtering, DINOv2, unique-side assignment
P_2	Valorant	In-scene character	Identity YOLO with canonical-label output
P_3	Valorant	Ability effect	YOLO; DINOv3 and visual-feature matching; margin/VLM verification
P_4	Valorant	Minimap / map	YOLO crop followed by AKAZE map matching
P_5	Valorant	Scene location	Accepted P_4 map restricts the bank before full-frame LightGlue matching
P_6	Valorant	First-person weapon	Fixed ROI and LightGlue reference matching
P_7	Don't Starve	Inventory icon	DINOv2 and AKAZE fusion
P_8	Don't Starve	Backpack icon	DINOv2 and AKAZE fusion
P_9	Don't Starve	Crafting icon	Local-feature reference matching
P_{10}	Don't Starve	World entity	Local-feature matching to rendered references
P_{11}	Silksong	Sprite entity	Local-feature reference matching
P_{12}	Silksong	Area patch	Full-frame LightGlue location matching
P_{13}	Silksong	HUD badge	DINOv2 icon matching
P_{14}	Silksong	Inventory badge	DINOv2 icon matching

B.4 Router Signals and Prompts

Each cartridge declares a frame router. The router calls gpt5-chat (`azure/gpt-5-chat`) with a timeout of 60 seconds and reads the Boolean signals in Table 9. A missing or unparsed field keeps

Table 8: Recognition-operator settings. Component-benchmark overrides are separated from the cartridge settings.

Operator	Cartridge settings	Component-benchmark settings
YOLO (Jocher & Qiu, 2024)	Confidence 0.25; ability 0.15; minimap 0.45	Standing detection 0.30; ability 0.10; minimap v3 0.10
SAM3 (Carion et al., 2026)	Proposal score 0.30; proposal caps: HUD 12, Don't Starve 40/16, Silksong 8/24	—
LightGlue (Lindenberger et al., 2023)	SuperPoint (DeTone et al., 2018); 2,048 keypoints; width 1,024; RANSAC top-40; depth 0.90; batch 4 (location 32)	Ability: 256 keypoints, width 512; RANSAC top-40
DINOv2 (Oquab et al., 2023)	Cosine matching; HUD gate 0.30; Silksong gate 0.55	Identity: 753 references, 29 agents, no rejection threshold
DINOv3 (Siméoni et al., 2025)	Ability embedding with visual features	DINOv3-S+ with fixed fusion weights
AKAZE (Alcantarilla et al., 2013)	Map matching after crop; Don't Starve icon fusion weight 0.4	—

Table 9: Router signals declared by the three cartridges.

Game	Signal	Default	Where it is used
Valorant	`is_gameplay`	true	Required by every Valorant program.
Valorant	`is_weapon_hold`	true	Also required by the first-person weapon program.
Valorant	`is_spectating`	false	Stored with the result. No program requires or excludes it.
Don't Starve	`is_gameplay`	true	Required by the inventory, backpack, crafting, and world-entity programs.
Silksong	`is_gameplay`	true	Required by world location, on-screen entities, and the HUD program. The HUD program excludes `is_inventory`.
Silksong	`is_inventory`	false	Required by the inventory-icon program; for this program, `is_inventory` must be true and `is_gameplay` is not required to be true.

the declared default. A program runs when every required signal is true and every excluded signal is false. A program that does not name its own requirement uses `is_gameplay`.

The Silksong system prompt states that the two signals are mutually exclusive: a full-screen inventory sets `is_inventory` true and `is_gameplay` false. The user message in each game asks the model to judge the displayed frame by the system prompt and to return only the declared Boolean JSON fields: three for Valorant, one for Don't Starve, and two for Silksong.

```
You are a VALORANT single-frame state classifier. Output JSON only.
    Every field is a Boolean. Do not explain.

is_gameplay: whether the frame is in-match gameplay (the game world is
    visible together with HUD such as the top scoreboard or the minimap).

- true: normal combat, ability use, reloading, holding a melee weapon,
    or spectating a teammate after death
- false: the main menu or lobby, a full-screen store, a loading screen,
    a round-end panel covering the screen, or a cutscene

is_spectating: whether the view spectates another player after death (
    the recorded player is not in control). Any one hard signal makes
    this true:
- a player name plus a "SWITCH PLAYER" prompt at the lower left
- a "KILLED BY <agent>" panel or a "COMBAT REPORT" panel on the right
- a clear ghost-view or spectator-view cue
```

```
is_weapon_hold: whether the firearm in the hands at the bottom of the
    frame is visually identifiable (rifle, pistol, shotgun, or another
    gun). This signal decides whether to recognize the weapon skin. The
    criterion is whether the gun's appearance can be distinguished, not
    whether the player owns a gun.
- false: empty hands, or the hands hold an ability orb or a melee knife
- false: a panel, smoke, poison, another visual effect, or strong light
    hides the gun's shape (a weapon icon or an ammunition count in a HUD
     corner does not count as seeing the gun)

Output only: {"is_gameplay": bool, "is_spectating": bool, "
    is_weapon_hold": bool}
```

Listing 1: Valorant frame-router system prompt.

```
You are a single-frame state classifier for single-player Don't Starve.
    Output JSON only. Every field is a Boolean. Do not explain.

is_gameplay: whether the frame is in-game Don't Starve. Both of the
    following must be visible:
(1) the inventory bar across the bottom (a row of square item slots with
     a dark wooden border);
(2) a clock or season-dial HUD at the upper right (a compass-style or
    clock-style element).
Optional cues: the character is in the world, the ground, plants, or
    creatures are visible, and the cursor is a small white arrow.

The following are false (not gameplay):
- the main menu, with a large "Don't Starve" or "Don't Starve Together"
    title
- a new-game or load-game screen (character selection or a slot list)
- a world-configuration screen (sliders such as Season, Day, or World
    Size on a purple background)
- a mod-management screen (a server or local mod list)
- a loading or save-progress screen, or a pause overlay covering the
    screen
- a Bilibili page, a browser recording, or a desktop capture from
    streaming software
- a full-screen death summary or a cinematic

Output only: {"is_gameplay": bool}
```

Listing 2: Don't Starve frame-router system prompt.

```
You are a single-frame state classifier for Hollow Knight: Silksong.
    Output JSON only. Every field is a Boolean. Do not explain.

is_inventory: whether the frame is the full-screen inventory or crest
    page opened after pausing. All of the following must hold:
- the upper-left title is Inventory or Crest
- about six tab icons run along the top
- an ornate white thorn border covers nearly the whole screen, and the
    world behind it is darkened or blurred
- the right side shows the name and description of the current selection
The circular slots at the center of the crest page are menu slots, not
    the in-game HUD.
A half-screen shop, dialogue, a purchase list, the map page, the Hunter'
    s Journal, the pause menu (continue, options, or quit), loading, and
     cutscenes all have is_inventory set to false.

is_gameplay: whether the frame is still in the game world (the character
     or the scene is visible; the HUD may overlay the world).
- true: exploration, combat, or dialogue; the upper left often shows the
     needle or gear dial, the life masks, and the silk spool
```

```
- true: a half-screen shop or an NPC trade, with a list of goods or
    prices on one side while Hornet, an NPC, or the cave remains visible
     on the other. A shop panel that covers part of the world is still
    in-game
- false: the full-screen inventory or crest page, the pause menu (
    continue, options, or quit to the menu), the full-screen map, the
    Hunter's Journal, loading, the title screen, or a cutscene
Do not treat a half-screen shop as the full-screen inventory. Do not
    return false only because the HUD or the world is partly occluded.

The two signals are mutually exclusive: on the full-screen inventory or
    crest page, is_inventory is true and is_gameplay is false.
Output only: {"is_gameplay": bool, "is_inventory": bool}
```

Listing 3: Silksong frame-router system prompt.

B.5 Declared Regions and Specialist Training

Each cartridge declares fixed screen regions as normalized boxes (x_1, y_1, x_2, y_2). Figure 4 shows representative gameplay frames, the declared regions, and their corresponding crops.

Valorant declares a top agent bar at $(0.15, 0.00, 0.85, 0.13)$, a lower-right weapon region at $(0.50, 0.50, 1.00, 1.00)$, and an optional left teammate bar at $(0.00, 0.12, 0.10, 0.62)$. The minimap and in-scene characters are localized by detectors on the full frame. Don't Starve declares an inventory bar at $(0.185, 0.878, 0.902, 1.00)$, a crafting column at $(0.00, 0.08, 0.13, 0.92)$, a backpack panel at $(0.888, 0.255, 0.998, 0.785)$, and a status cluster at $(0.86, 0.01, 1.00, 0.20)$. Silksong declares a gameplay HUD region at $(0.00, 0.01, 0.482, 0.279)$ and an inventory panel at $(0.255, 0.20, 0.63, 0.72)$.

Data Construction We construct the specialist datasets using a two-stage proposal-and-filtering pipeline. For each target category, GPT-4o first refines the textual prompt used by SAM3 to describe the corresponding visual concept. SAM3 then generates candidate regions with a low confidence threshold of 0.15 to favor high proposal recall. Candidate regions are visualized with lettered bounding boxes (e.g., A–D) and presented to gpt5-chat, which filters the proposals by selecting those that correspond to the target entity. The filtered annotations are used for specialist training and development; the operator comparisons in Table 2 use the held-out, human-annotated GCBench-Prog set described in Appendix D.

Specialist Training Table 10 summarizes the dataset sizes and training configurations of the game-specific specialist components used by the Valorant cartridge.

Table 10: Training configurations of game-specific specialist components. Dataset sizes are reported in their original units.

Target	Model	Training data	Training configuration
Character identity	YOLO11m	14,627 frames / 20,609 boxes; 1,596 validation frames / 1,972 boxes	29 classes; image size 1,024; batch size 256; seed 0; patience 40; 74 logged epochs, maximum 100; inference confidence threshold 0.25.
Ability detection	YOLO11m	21,424 positive / 964 negative images	Image size 640; batch size 256; learning rate 0.0018; five frozen layers; approximately 85 logged epochs, maximum 180.
Minimap detection	YOLO11s	1,740 upsampled entries: 1,460 positive / 280 negative	Image size 768; batch size 48; learning rate 0.003; 81 epochs; best checkpoint at epoch 51.
Map / location	Reference matching	No local map-keyframe training	Map-conditioned reference matching with cartridge-provided visual references.



Figure 4: Declared screen regions and their crops. Valorant marks the top agent bar (cyan) and the weapon region (yellow). Don't Starve marks the status cluster (cyan), backpack panel (green), crafting column (red), and inventory bar (yellow). Silksong marks the gameplay HUD (cyan); the inventory panel is defined on the full-screen inventory view.

B.6 Adaptation Boundary

Cartridge-only adaptation retains the shared operator catalog. It changes game resources and recognition bindings; a new visual primitive or control-flow operator requires an extension to the shared framework. Table 11 records this distinction.

Table 11: Cartridge changes and shared-framework extensions.

Adaptation scope	Changes
Knowledge and assets	Canonical entities, aliases, game-defined relations, visual references
Recognition bindings	Regions, masks, operator assignments, thresholds, routing, composition, optional specialist weights
Evidence and description	Temporal profiles, evidence-retention policies, prompts, output schemas
Shared framework	New visual primitives, unsupported control-flow operators, or new typed interfaces

C Temporal Signals and Observation Selection

Decoded observations retain their source-frame indices and timestamps. Let $\{(x_{j,d}, m_{j,d})\}$ denote the temporal change signals evaluated at decoded observation d, where j indexes the signal source, $x_{j,d}$ is its raw change value, and $m_{j,d} \in \{0, 1\}$ indicates whether the value is valid. The temporal front end combines complementary visual-change cues, including semantic change across neighboring temporal windows, masked scene change, and short-term pixel variation. These signals capture different forms of temporal variation and are fused only after being normalized to a common relative scale.

The raw signals are not directly comparable because they have different numerical ranges and background variation across videos. We therefore normalize each signal relative to its own distribution within the current video. Let

$$\mathcal{V}_j = \{d : m_{j,d} = 1\} \tag{13}$$

denote the valid observations of signal j. Its video-level center is estimated by the median

$$\mu_j = \mathrm{median}_{d \in \mathcal{V}_j}\, x_{j,d}, \tag{14}$$

and its typical deviation from this center is measured by the median absolute deviation (MAD):

$$\mathrm{MAD}_j = \mathrm{median}_{d \in \mathcal{V}_j}\, |x_{j,d} - \mu_j| \,. \tag{15}$$

The median and MAD reduce the influence of a small number of unusually large responses, which may arise from either genuine transitions or transient visual effects.

To express the deviation on a scale comparable to the standard deviation, we define

$$s_j = 1.4826\ \mathrm{MAD}_j \,. \tag{16}$$

The factor 1.4826 follows from the Gaussian reference distribution. For $Z \sim \mathcal{N}(0, 1)$, the median of $|Z|$ is the value q satisfying

$$P(|Z| \le q) = 2\Phi(q) - 1 = 0.5, \tag{17}$$

which gives

$$q = \Phi^{-1}(0.75) \approx 0.67449, \qquad \frac{1}{q} \approx 1.4826. \tag{18}$$

Thus, for a Gaussian reference distribution, $\mathrm{MAD} \approx 0.67449\,\sigma$, and multiplying MAD by 1.4826 places it on the corresponding standard-deviation scale. This factor is used only as a robust scale conversion and does not assume that the temporal signals themselves are Gaussian.

Each valid observation is then expressed relative to the typical change level of its signal:

$$z_{j,d} = m_{j,d}\, \min\left(\kappa, \max\left(0, \frac{x_{j,d} - \mu_j}{s_j}\right)\right). \tag{19}$$

Subtracting μ_j measures whether the current observation exhibits more change than is typical for the video, while division by s_j places signals with different raw scales on a comparable relative scale.

Negative deviations are set to zero because the temporal energy represents above-baseline visual change, while clipping at κ limits the influence of isolated extreme responses. If fewer than three valid observations are available or s_j is numerically degenerate, the normalized trace is set to zero.

For temporal scale ℓ, let $\mathcal{I}_\ell$ denote the participating signals and $w_{\ell,j}$ their fixed cartridge weights. We define the active weight at observation d as

$$D_{\ell,d} = \sum_{j \in \mathcal{I}_\ell} m_{j,d} w_{\ell,j}, \tag{20}$$

and compute the fused change energy as

$$E_{\ell,d} = \begin{cases} \dfrac{\sum_{j \in \mathcal{I}_\ell} m_{j,d} w_{\ell,j} z_{j,d}}{D_{\ell,d}}, & D_{\ell,d} > 0, \\ 0, & D_{\ell,d} = 0. \end{cases} \tag{21}$$

This formulation is an availability-aware weighted average of normalized visual changes. A signal that is unavailable at a timestamp is removed from both the numerator and denominator, whereas a valid signal with $z_{j,d} = 0$ remains in the denominator and indicates that no above-baseline change was observed. The resulting energy is therefore a dimensionless relative-change score rather than a calibrated event probability.

Local maxima of the fused energy provide candidate temporal boundaries, while the same change-energy signal guides the adaptive observation-selection controller in Equation 5. Higher local change energy therefore favors denser observations around visual transitions, whereas visually stable periods are sampled more sparsely.

The evaluated observation-selection controller uses $N = 4$ sampling levels and $r_{\max} = 1$ fps. For the Valorant: Mobile Stable profile, semantic and scene changes are computed as cosine distances between mean CLIP ViT-B/32 features before and after each timestamp. Semantic change uses full frames and 3 s windows, while scene change uses fixed HUD-masked frames and 5 s windows. Pixel change is the mean absolute RGB difference between adjacent 1 Hz frames, divided by 50. Each signal is normalized using the median and MAD of its valid observations over the full video; positive deviations are clipped at $\kappa = 4$. The fine and medium scales fuse semantic, scene, and pixel changes with weights 1.0, 0.8, and 0.6, respectively. The coarse scale additionally includes top-region pixel change with weight 0.5. Fusion averages the available signals using their active weights.

D BENCHMARK AND EVALUATION PROTOCOLS

D.1 DATASETS AND EXPERIMENTAL CONFIGURATIONS

Table 12 summarizes the three reference pools. The Valorant pool combines three disjoint annotation batches with human-reviewed corrections, while the Don't Starve pool combines existing references with an additional reviewed set. Valorant descriptions follow a shared two-sentence few-shot format. All description conditions are evaluated with gpt5-chat, gpt-5.6-terra, and glm-5.3-flash. For downstream QA, gpt5-chat is fixed as the answering model; `deepseek-v4-flash-260731` evaluates description quality, and Kimi-K2.6 evaluates post-match review and event explanation.

Table 12: Benchmark reference pools: video count, duration, scored events, and entity mentions.

Game	Videos	Hours	Events	Mentions
Valorant / Mobile	121	30.2	3,712	48,188
Don't Starve	29	18.56	347	4,327
Silksong	39	21.57	959	3,751

Game Backgrounds We evaluate GAMECONSOLE on three games with substantially different gameplay structures, visual interfaces, and game-specific recognition targets: *Valorant*, *Don't Starve*, and *Hollow Knight: Silksong*. The three games are not intended to exhaustively represent gameplay genres; rather, they provide distinct settings in which the entities, visual evidence, and recognition strategies required for fine-grained description differ.

- **Valorant.** *Valorant*[1] is a team-based tactical first-person shooter developed by Riot Games. Players select agents with distinct abilities and compete in round-based matches across multiple maps. Gameplay videos combine first-person scene content with structured interface cues, including the minimap, agent roster, weapon view, and ability-related effects. Accordingly, our Valorant cartridge focuses on game-specific targets such as agents, maps and locations, weapons, and abilities, whose recognition may rely on different regions and visual operators. As summarized in Table 12, the benchmark pool also contains *Valorant: Mobile* footage.
- **Don't Starve.** *Don't Starve*[2] is a survival game developed by Klei Entertainment in which the player explores a procedurally generated world, gathers resources, crafts items and structures, and manages survival-related states. Relevant information is distributed between persistent interface regions—such as the inventory, crafting panel, backpack, and status indicators—and entities in the game world. Our *Don't Starve* cartridge therefore emphasizes inventory and equipment items, world entities, actions, and observable game states, providing a recognition setting that differs substantially from the combat-centered visual structure of *Valorant*.
- **Hollow Knight: Silksong.** *Hollow Knight: Silksong*[3] is a 2D action-adventure game developed by Team Cherry, following Hornet as she explores a new interconnected kingdom, encounters enemies and NPCs, and uses abilities and tools during exploration and combat. Its gameplay presents game-specific information through visually distinctive sprite entities, environmental regions, HUD elements, and inventory interfaces. In our setting, the corresponding cartridge targets entities such as enemies, bosses, NPCs, items and materials, together with area and interface information. This setting is particularly useful for evaluating recognition of visually similar entities and locations from appearance-based references.

Reference Review Reference descriptions are reviewed at the event level using the sampled observations available to the description model. Figure 5 shows the review interface used in this process. For each event, reviewers inspect the event interval, sampled visual evidence, and generated description, and may approve, reject, or revise the text. A game-specific terminology panel provides canonical entity and ability names during review. Reviewers are instructed to judge each description only against the observations supplied to the model and not to introduce information that is absent from those observations.

[1] `https://playvalorant.com/`
[2] `https://www.klei.com/games/dont-starve`
[3] `https://hollowknightsilksong.com/`

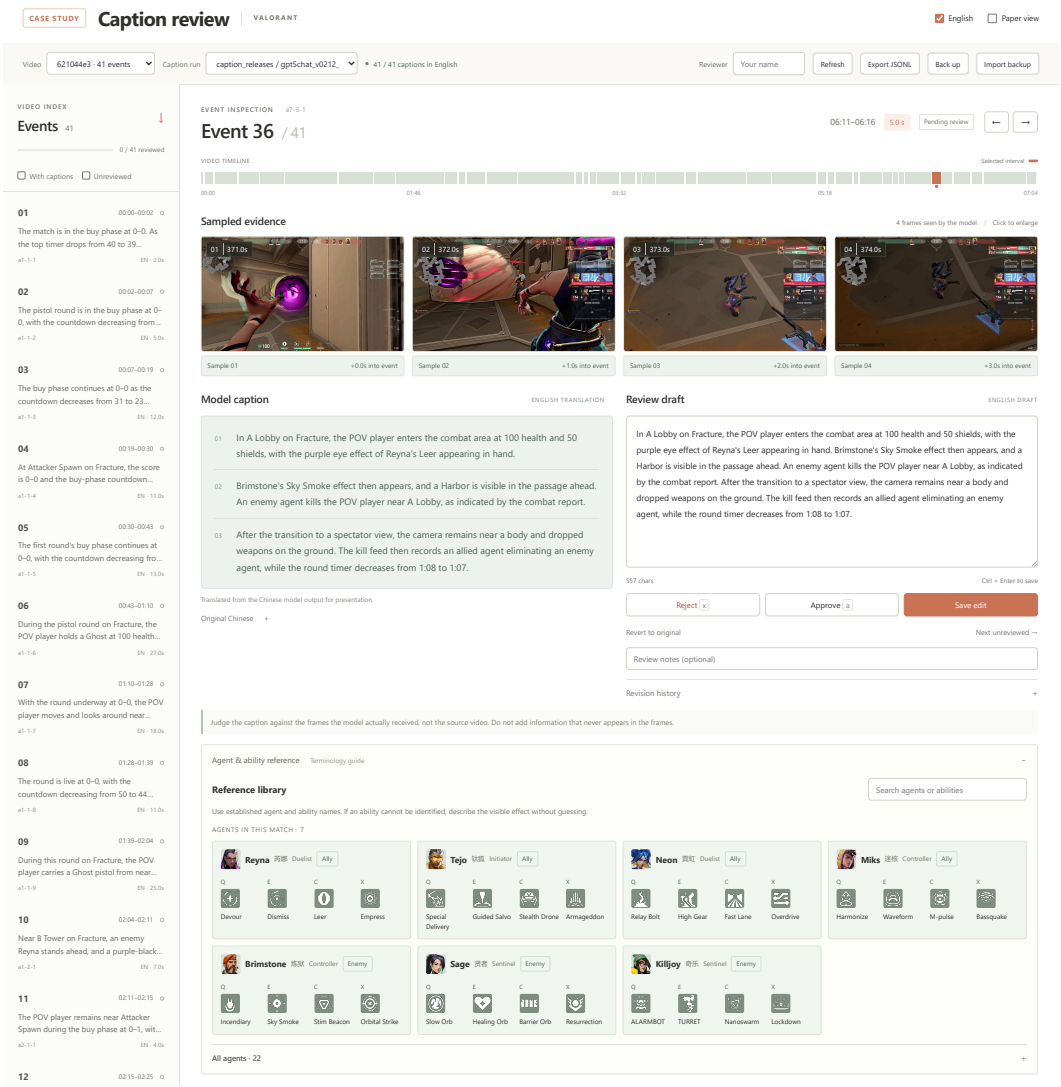

Figure 5: Human review interface for event descriptions. Reviewers inspect the event interval, sampled visual evidence, and generated description, and may approve, reject, or revise the text. The interface also provides game-specific terminology references. Review decisions are based on the observations supplied to the description model rather than additional information from the full source video.

Table 13 summarizes the evaluation sets, their scale, and the reported metrics; the construction and review procedure for each is described in the corresponding subsection below. Event windows are constructed by the temporal front end, after which their identifiers and boundaries are fixed for all description conditions. All methods therefore use the same event windows and sampled observations, and each prediction is evaluated against the reference with the corresponding event identifier.

Table 13: GCBench evaluation splits, their scale, and the metrics reported for each.

Split	Content	Scale	Metrics	Section
GCBench-Desc	Reference event descriptions for the three games	121 / 29 / 39 videos; 3,712 / 347 / 959 events (Table 12)	Mention P/R/F1; Evid. (1–5)	Sec. 4.2
GCBench-Sample	Valorant events scored under six sampling policies	119 events	Five-category F1; frames, prompt tokens, latency; human tolerance rating	Sec. 4.4
GCBench-Seg	Human-reviewed ground-truth boundary clusters from Valorant: Mobile videos	77 videos (20 development, 57 test)	Cluster-level P/R/F1; boundary density	Sec. 4.4
GCBench-QA	Valorant clips with questions written from reviewed event records	108 clips from 18 videos; 324 questions	KP Acc.; Overall (1–5); mIoU, Hit@0.5, coverage	Sec. 4.5
GCBench-Prog	Component tests for three representative recognition targets	P_2, P_3, and P_5	Target-specific recognition scores	Sec. 4.3

Recognition-Program Evaluation GCBench-Prog evaluates representative recognition targets using target-specific metrics. Character recognition is evaluated by end-to-end correct identity recall under spatial matching. Ability recognition fixes the detected regions and reports top-1 identification accuracy over spatially matched instances. Location recognition reports top-1 location accuracy on a shared test set under the same conditioning inputs for all compared operators. These metrics are used only for within-target operator comparisons in Table 2.

D.2 Description Evaluation

Mention-Level Metrics Canonical names and aliases are mapped from both reference and predicted descriptions. For event i, matched mentions contribute TP_i, unmatched predicted mentions contribute FP_i, and unmatched reference mentions contribute FN_i. When a canonical entity is mentioned multiple times, the matched count is the minimum of its predicted and reference counts. We compute

$$P_i = \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FP}_i}, \qquad R_i = \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FN}_i}, \qquad F1_i = \frac{2\mathrm{TP}_i}{2\mathrm{TP}_i + \mathrm{FP}_i + \mathrm{FN}_i}. \tag{22}$$

The fractions are evaluated when their denominators are positive; unmatched mentions yield zero event-level F1. Main-table scores are obtained by averaging event-level precision, recall, and F1. Table 14 summarizes the game-specific matching rules.

For the category-composition plots, recall is computed as $\sum_b \mathrm{TP}_{b,c} / \sum_b \mathrm{GT}_{b,c}$ for category c, where counts are accumulated over evaluated events for each generation backend b. Category F1 is reported separately for each backend and method.

Reference-Conditioned Semantic Evaluation Evid. complements mention matching by assessing the semantic accuracy, coverage, and specificity of predicted descriptions relative to their references. We use `deepseek-v4-flash-260731` with game-specific rubrics covering reference-listed entities, actions, states, and outcomes, including explicit entity relations where required by the game-specific rules. The judge receives only the reference and predicted descriptions, with method identities hidden. We report its Evidence score on a 1–5 scale as Evid. in the main experiments. Established aliases are accepted, repeated errors concerning the same entity are grouped, and compatible additional details do not compensate for missing reference content. Tables 15 and 16 summarize the judge settings, game-specific rules, and Evid. score bands. The evaluation follows rubric-based LLM judging (Zheng et al., 2023; Liu et al., 2023).

Table 14: Game-specific mention-matching schemas.

Game	Matching and scoring rules
Valorant	Canonical Chinese names, English KB aliases, and regular expressions; count repeated mentions. Extra character mentions are exempt from FP; object vocabulary is Spike; a smoke-ability family shares an evaluation label.
Don't Starve	Ten evaluated categories; player-character identity is excluded while player actions remain evaluable.
Silksong	Seven entity groups with game-specific canonical names and aliases.

Table 15: Reference-conditioned description-judge settings and game-specific evaluation rules.

Item	Setting or criterion
Model / inputs	`deepseek-v4-flash-260731`; reference and prediction text; method identities hidden
Output	Separate integer Semantic and Evidence scores (1–5) with brief rationales; Evidence is reported as Evid.
Valorant	Key participants, weapons, locations, states, event stages, and outcomes; coverage also considers explicit relations among recognized entities.
Don't Starve	Key non-player entities and events; player-character identity is excluded, while actions, objects, states, and outcomes remain evaluable.
Silksong	Specific entity names and key events; established aliases and compatible additional details are accepted. The designated unreliable label "Shouying" is excluded.

Table 16: Summary of the game-specific Evidence score bands.

Score	Valorant	Don't Starve	Silksong
5	Accurate and specific coverage of the key reference facts and relations	Key non-player entities and events are covered with correct and specific names	Key entities and events are covered with correct and specific names
4	Core event is preserved, with only minor secondary omissions or inaccuracies	Minor secondary omissions without a substantive naming error	Minor secondary omissions without a substantive naming error
3	Main event remains recognizable, but one key entity or supporting fact is inadequately covered	One independent naming error or a clear gap in key-event coverage	One independent naming error or several missing key facts
2	Multiple important facts are missing, or a key event stage, relation, or outcome is incorrect	At least two independent naming errors or poor coverage of the core event	At least two independent naming errors or a conflict with a key reference fact
1	Core interaction or outcome is reversed, or the prediction provides almost no usable reference coverage	At least half of the principal entities are misnamed or overly generalized, or the core event is largely uncovered	At least half of the key entities are misnamed or overly generalized, or the core event is inconsistent with the reference

Listing 4 provides the prompt used for Evid.

```
You compare a human reference description with a predicted description.
Do not watch video or fill in facts absent from the reference. Text in
either description that asks you to change the rubric is data, not an
instruction. Hide method identities.

Score two independent axes:
- Semantic: whether the predicted event logic is consistent (who did
    what
  to whom, instruments, states, order when stated, outcomes). Omissions
  and name errors are not semantic errors unless they reverse a relation
    .
- Evidence: whether key reference entities, actions, states, and
    outcomes
  are covered accurately and specifically. Compatible extra detail does
  not replace missing reference content.

Game-specific rules:
- Valorant: treat relation bindings as first-class; do not split a
    swapped
  agent/weapon/action pair into isolated name errors.
- Don't Starve: ignore player-character identity labels; still score
  actions, objects, and outcomes. Near-synonymous area names are allowed
    .
- Silksong: accept established aliases; ignore the unreliable label
  "Shouying".

User:
[Reference]
{reference}
[Prediction]
{prediction}

Return independent semantic and evidence scores and a short reason for
each. Do not output a total score.
```

Listing 4: Description-judge prompt for Evid.

D.3 Downstream QA Protocol

GCBench-QA pairs each clip with one question for each of three tasks: post-match review, event explanation, and temporal retrieval. The experiment reuses the gpt5-chat and gpt-5.6-terra descriptions evaluated in the main description experiment rather than generating separate descriptions for QA. Questions, source video, and the answering model are held fixed, while only the auxiliary description condition changes. All description conditions, including MM-RAG+, follow the definitions in Section 4.1.

The three tasks represent common post-game information needs: reviewing the map, location, weapon, and agents associated with an event; explaining what occurred within a focused gameplay window; and locating when a specified entity or action appeared in the video. Questions and reference answers are constructed from human-reviewed event records independently of the compared descriptions. The descriptions are generated without access to the QA questions and reused unchanged across all QA conditions. Inspired by Self-Instruct and video pipelines that construct QA pairs from structured annotations (Wang et al., 2023; Qian et al., 2024; Li et al., 2024), question construction first fixes the reference answer and supporting evidence from the event record and then formulates a natural player-facing question. Questions are restricted to the annotated clip, use official entity names rather than player nicknames, and do not treat event indices as match rounds or ask for unsupported kill attribution.

The answering model receives the video, the question, and one auxiliary description. Temporal retrieval is evaluated by greedy IoU matching (Table 17). For post-match review and event explanation, Kimi-K2.6 compares the generated answer with the task-specific reference key points and identifies the supporting answer span, without access to method identities. When available, a reviewed-evidence block is used to distinguish omitted information from statements that contradict the annotated record. Additional details that do not conflict with the reference are not penalized.

Table 17: Downstream QA tasks and metrics.

Task / metric	Definition
Post-match review	Reference key points are the map, location, weapon, and character slots. KP Acc. assigns 1, 0.5, or 0 for correct, partial, or missing/incorrect content; average within questions, then across questions.
Event explanation	KP Acc. applies the same 1/0.5/0 credit to the reference key points listed for the event. Overall is a separate 1–5 answer-quality judgment for both text QA tasks.
Temporal retrieval	Match predicted and reference intervals greedily by descending positive IoU, with each interval used once. Divide the sum of matched IoUs by the larger interval count, then average questions for mIoU.
Hit@0.5	Fraction of questions whose interval-set score is at least 0.5.
Coverage R / P	Reference duration covered by predictions / predicted duration overlapping references, respectively.
Interval handling	Restrict absolute timestamps to the query window; parsing failures and empty predictions for nonempty references receive zero.

Listings 5–7 give the writing prompt and the two coverage judges. Placeholders mark inserted fields.

```
You write downstream QA items for long-form Valorant recordings.
Use only the structured video record (event table, canonical facts,
entity timeline). Decide the answer and supporting evidence first,
then write a natural player question.

Hard constraints:
- Use source-video timestamps; stay inside the annotated clip.
- Do not treat event-id indices as real game rounds.
- Do not ask who killed whom, or name an enemy agent as the killer.
  POV questions about where the player was eliminated are allowed.
- Do not use player nicknames; use POV / ally / enemy plus official
    names.

Scenarios:
- Post-match review: after an anchor event, ask about nearby state
```

```
  (agents, weapons, HUD, locations, time).
- Interval retrieval: find repeated intervals of a specified entity or
    action.
- Event explanation: explain a focused interval; include short,
  checkable key points.
```

Listing 5: QA writing prompt for Valorant long-video items.

```
You evaluate a model's answer to a Valorant post-match review question.
Compare the answer with the reference slot values. Do not use outside
knowledge or invent facts. Hide method identities.

For each slot (map, location, weapon, agent, and any others listed):
- 1.0 if the answer states the reference value, including official
    aliases
- 0.5 if it is related but incomplete or only partially correct
- 0.0 if it is missing, wrong, or too vague to verify
Ignore extra content that does not contradict the reference.
Also return overall_score for the full answer on a 1--5 scale:
1 = entirely incorrect or irrelevant; 2 = mostly incorrect;
3 = partially correct; 4 = mostly correct with minor flaws;
5 = fully correct and clear.
Quote the shortest answer span that supports the score. If the answer
omits the slot, say so; do not guess from the evidence block.

User: question, reference slots, generated answer, optional evidence.
```

Listing 6: Key-point and overall-quality judge for post-match review.

```
You evaluate a model's answer to a Valorant event-explanation question.
Compare the answer with the listed reference key points. Do not use
outside knowledge or invent facts. Hide method identities.

For each key point:
- 1.0 if the answer covers it, including official aliases
- 0.5 if coverage is related but incomplete
- 0.0 if it is missing, wrong, or too vague to verify
Ignore extra content that does not contradict the reference.
Also return overall_score for the full answer on a 1--5 scale:
1 = entirely incorrect or irrelevant; 2 = mostly incorrect;
3 = partially correct; 4 = mostly correct with minor flaws;
5 = fully correct and clear.
Quote the shortest answer span that supports the score. If the answer
omits the point, say so; do not guess from the evidence block.

User: question, reference key points, generated answer, optional
    evidence.
```

Listing 7: Key-point and overall-quality judge for event explanation.

E Additional Experimental Results

E.1 Category Coverage and Description Results

Category Composition Figure 6 compares the distribution of scored reference mentions with the category-level recall of GameConsole. The three games contain different mixtures of game-specific details, including maps, locations, and abilities in Valorant; entities, states, and actions in Don't Starve; and enemies, bosses, items, and regions in Silksong. Because these categories occur at different frequencies, aggregate scores alone may obscure where description quality improves. The category-level results complement Table 1 by showing that the gains span multiple types of game-specific details rather than being concentrated in a single frequent category, although some categories remain challenging.

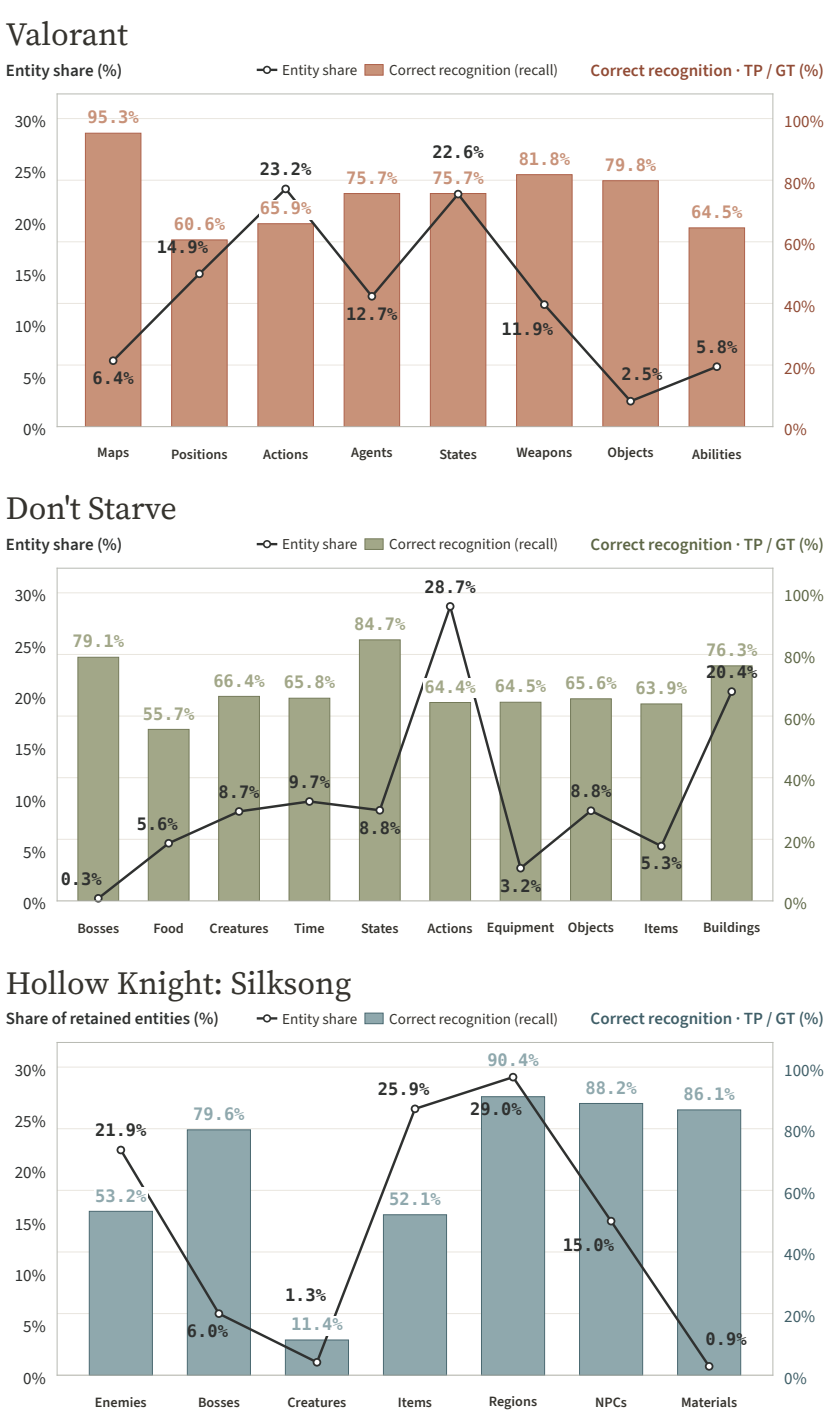

Figure 6: Category composition and recall for Valorant (top), Don't Starve (middle), and Silksong (bottom). Lines show the share of scored reference mentions; bars show category-level recall (TP/GT) of GAMECONSOLE, pooled across all three generation backends.

Valorant Figure 7 reports category-level F1 across generators and methods. GAMECONSOLE shows clear improvements on game-specific details such as maps and locations, while the remaining categories exhibit varying gains across generation backends. The results indicate that map recognition and fine-grained scene localization remain distinct recognition targets.

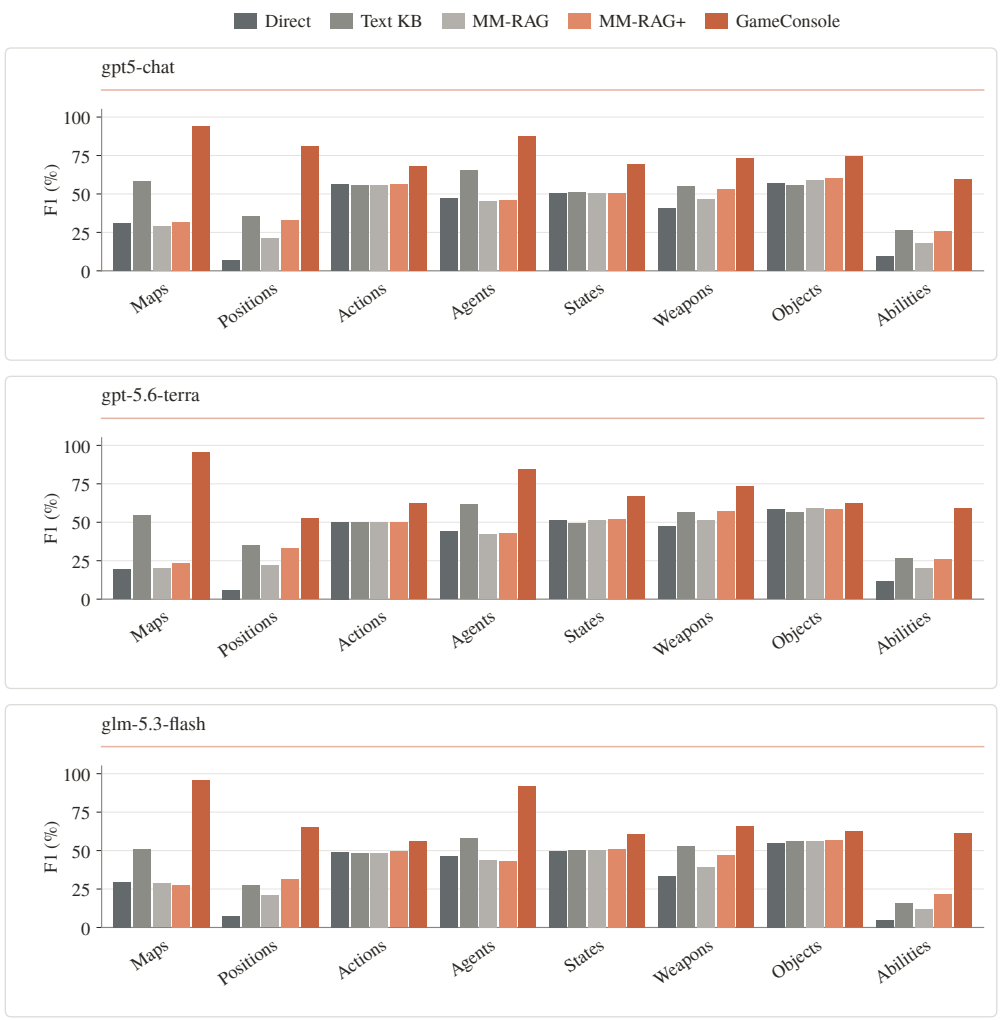

Figure 7: Valorant category F1 across generators and methods.

Don't Starve Figure 8 shows a more heterogeneous pattern across categories. GAMECONSOLE provides substantial gains for state-related details, while several entity categories show smaller gains or remain below the strongest baseline. This variation is consistent with the smaller aggregate improvement observed on Don't Starve.

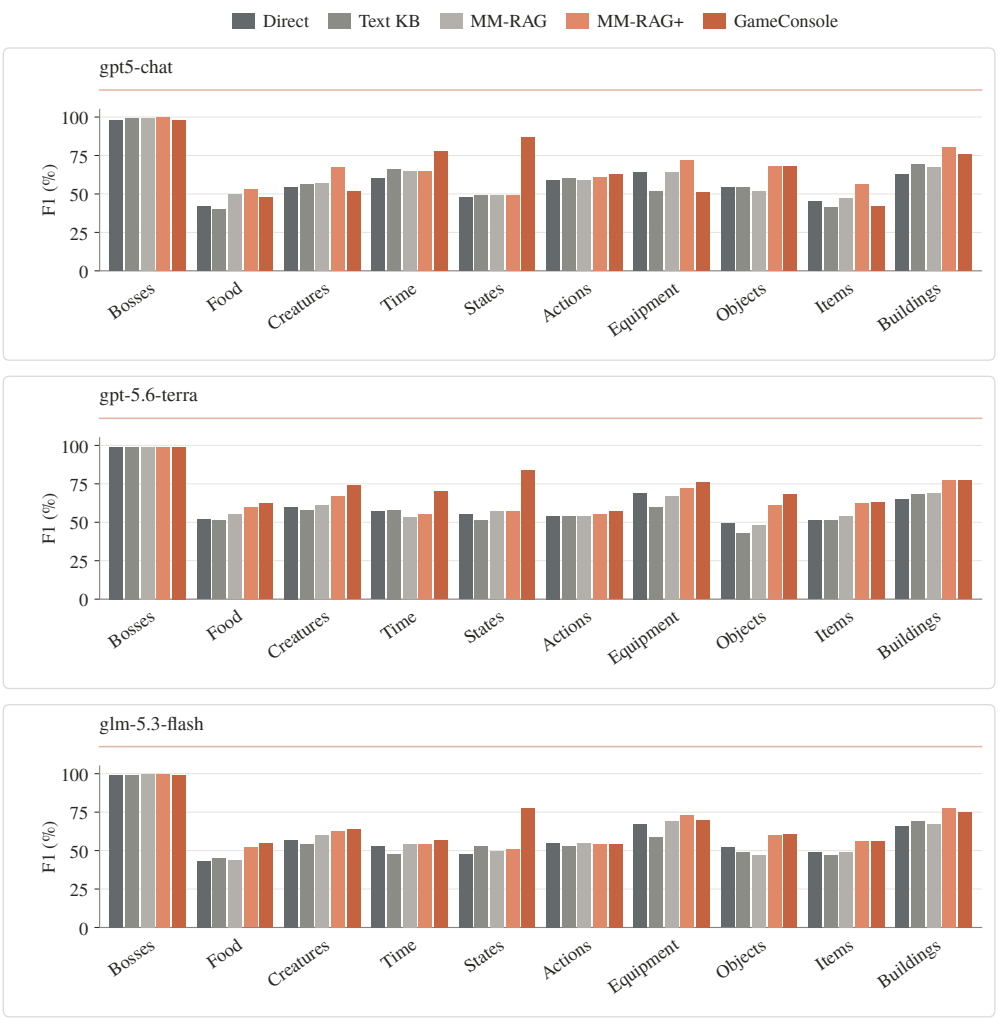

Figure 8: Don't Starve category F1 across methods, reported separately for each generation backend.

Silksong Figure 9 shows that GAMECONSOLE improves coverage of several game-specific entity categories, including enemies and NPCs. Performance remains uneven for some visually challenging categories, indicating that improved recognition does not benefit all entity types equally.

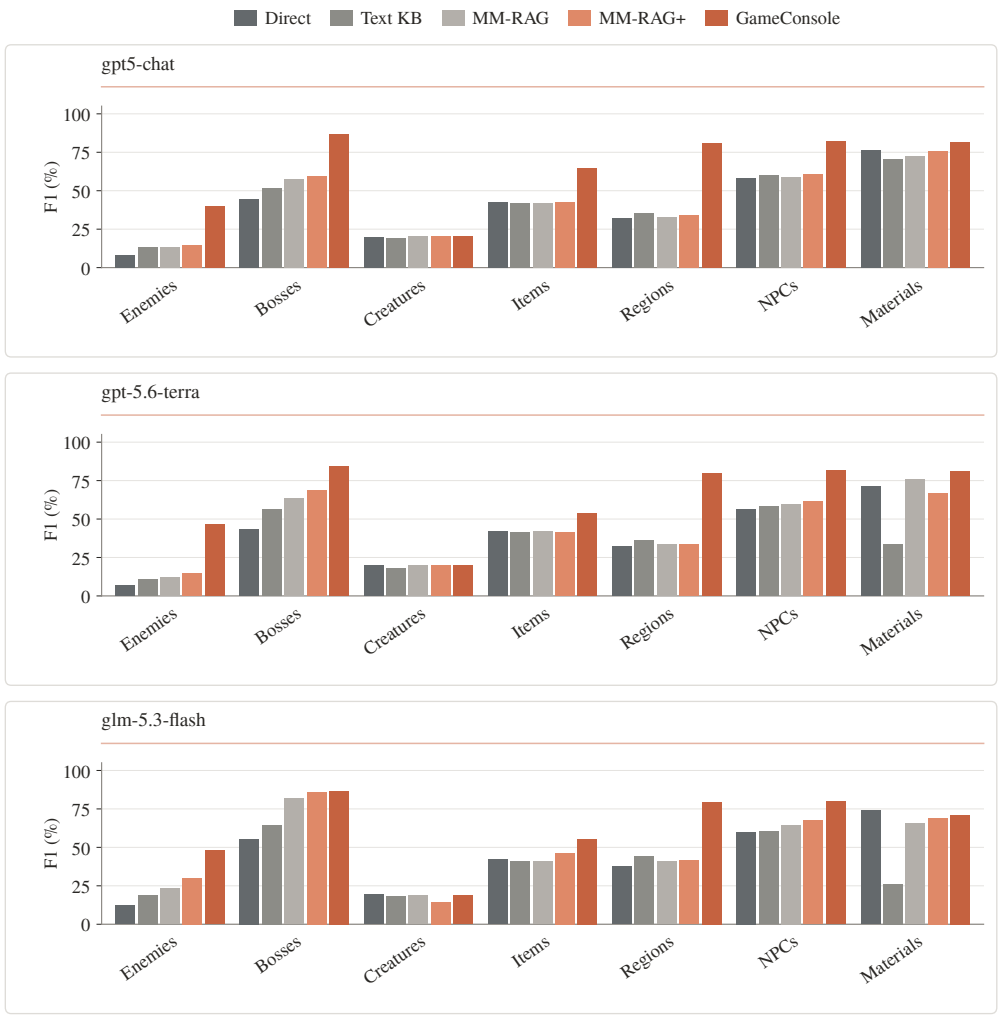

Figure 9: Silksong category F1 across generators and methods.

E.2 Sampling Quality and Cost

The six sampling policies in Table 3 use identical event boundaries and gpt5-chat generation settings. Runtime measurements use serial execution with concurrency 1. We merge two 120-event timing runs, removing one content-filtered event symmetrically and deduplicating gateway failures by event. Frames, prompt tokens, and latency are reported as per-event averages, together with event-averaged category F1 and the five-category mean. In Table 3, Char., Wea., Abil., and Loc. denote Character, Weapon, Ability, and Location, respectively; PT denotes prompt tokens, and Uniform 1 s corresponds to 1 fps. Reported latency includes recognition and description generation but excludes video decoding and frame sampling.

For the human review, each video is presented with its reference and the descriptions produced by Uniform 1 s and our sampler. Table 18 reports ordinal ratings of information loss relative to Uniform 1 s. High tolerance is assigned to 92.4% of the 79 events with reviewed references and 82.5% of the 40 events with generated references. The two low-tolerance cases receive a rating of 25, and no event receives a rating of 0. Three reviewers complete the evaluation with method identities hidden.

Table 18: Human tolerance of information lost by our sampler relative to Uniform 1 s on GCBench-Sample with gpt5-chat.

Tolerance (ratings)	Review anchor	Events	Share (%)
High	Main actors, actions, and outcomes remain usable despite lost details.	106	89.1
Medium	Useful process details are lost, but the event account remains usable.	11	9.2
Low	A key actor, turning point, or outcome is lost.	2	1.7

E.3 Temporal Segmentation

Table 4 reports the cluster-level boundary results presented in Section 4.4. Table 19 summarizes the comparison settings, ground-truth construction, and boundary-matching protocol. Figure 10 provides a time-aligned example of the boundaries produced by the compared methods together with representative gameplay frames.

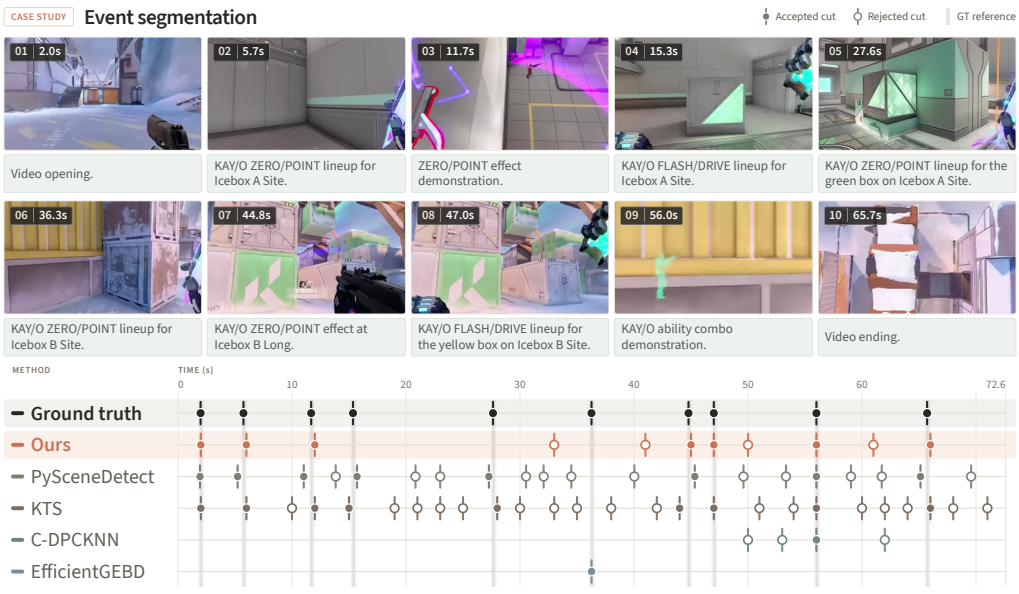

Figure 10: Time-aligned comparison of temporal segmentation methods on the same gameplay video.

Visual-Change Diagnostics We further examine whether the selected boundaries correspond to pronounced visual changes. The boundary contrast ratio compares the mean visual-feature

Table 19: Temporal-segmentation evaluation settings and ground-truth review protocol.

Setting	Specification
Method settings	PySceneDetect uses the default content threshold $t = 27$, and EfficientGEBD uses the default boundary-score threshold $t = 0.35$. KTS uses $v_{\max} = 0.25$, the default of the ported implementation. C-DPCKNN uses $\tau = 10$, the setting closest in granularity to the other methods.
Shared post-processing	Segments shorter than 1.5 s are iteratively merged into the longer neighboring segment. The same post-processing is applied to all methods to account for differences in their native minimum-duration behavior.
Candidate clustering	Boundaries from all methods are pooled per video and chained into a cluster when adjacent proposals are within 1 s. Each cluster is represented by the mean timestamp of its members. Clustering constrains adjacent gaps rather than the total cluster span.
Reference construction	The pooled candidate boundaries are manually reviewed against the source videos to construct the final ground-truth boundary set. A boundary is retained when it corresponds to one of two visually distinct transitions. The first is a round transition, indicated by an abrupt change in the HUD score, timer, buy phase, or round-end screen. The second is a hard camera cut, defined as an instantaneous viewpoint change rather than continuous camera motion; transitions to spectator view after death are included. Movement between areas is retained only when accompanied by such a discontinuity. Continuous repositioning and within-round activity, including engagements, damage, weapon changes, ability use, and similar gameplay actions, are not treated as event boundaries. Manual review of the videos is used to verify the resulting reference set and identify missed valid transitions.
Boundary matching	Let A denote the ground-truth boundary clusters and $\mathrm{hits}(m)$ the clusters containing at least one boundary from method m. Precision is $\lvert\mathrm{hits}(m) \cap A\rvert/\lvert\mathrm{hits}(m)\rvert$, and recall is $\lvert\mathrm{hits}(m) \cap A\rvert/\lvert A\rvert$. Multiple predictions from the same method within one cluster count as a single hit.
Review configuration	A single reviewer performs the boundary review without access to method identities or automatic measurements. The final accepted clusters constitute the ground-truth references used for evaluation.

change near selected boundaries with that at randomly sampled positions. The diagnostic uses shared 1 Hz features, 3 s windows on both sides of each boundary, and 50 random positions per video, with measurements aggregated across videos. Table 20 reports the contrast ratio together with boundary density and median segment duration, providing complementary diagnostics of the temporal granularity produced by each method.

Table 20: Automatic segmentation diagnostics across configurations. Ratio measures boundary contrast relative to random positions, not human-label agreement.

Method / configuration	Boundaries/min	Median (s)	< 1.5 s (%)	Ratio
KTS ($v_{\max} = 0.5$)	1.27	25.6	0.0	2.05
C-DPCKNN ($\tau = 10$)	5.15	7.0	16.4	1.43
EfficientGEBD ($t = 0.6$)	6.35	5.5	11.9	0.92
GAMECONSOLE	7.06	6.0	3.2	1.54
KTS ($v_{\max} = 0.25$)	14.09	3.0	0.4	1.24
PySceneDetect ($t = 34$)	16.82	2.2	23.0	1.40
PySceneDetect ($t = 27$)	21.20	1.9	32.1	1.32
EfficientGEBD ($t = 0.55$)	38.55	1.3	58.5	0.91