UNIVERSITY OF CALIFORNIA
SANTA CRUZ

INTEGRATED PLAYER ASSISTANCE WITH LIVE COACHES

A thesis submitted in partial satisfaction
of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTATIONAL MEDIA

by

Ivan Martinez-Arias

June 2026

The Thesis of Ivan Martinez-Arias
is approved:

Assoc. Prof. Adam M. Smith, Chair

Assist. Prof. Kathryn Ringland

Peter Biehl
Vice Provost and Dean of Graduate Studies

Contents

Figures and Tables

List of Figures

List of Tables

Abstract

Integrated Player Assistance with Live Coaches

by

Ivan Martinez-Arias

Existing forms of player-assistance, such as game walkthroughs and strategy guides, provide broad, general advice that disengage players from gameplay when seeking help at a particular moment. Furthermore, the assistance is static, unable to adapt to a player's current situation and evolving goals. I introduce Live Coaches as a genre of integrated AI systems that provide high-context and real-time player-assistance. Live Coaches support players in game completion and beyond, through integrating two key design dimensions: liveness, the ability to maintain real-time contextual awareness, and coachness, the ability to leverage deep game knowledge. I analyze prior systems exhibiting Live Coach characteristics and present two newly created systems: ChozoMaps and Rotom. Furthermore, I propose a design framework that enables designers to develop future Live Coach systems within the genre. An ablation study of 10 participants examined ChozoMaps' individual and combined impacts of liveness and coachness on player experiences. Findings illustrate that liveness without coachness leads to context-aware, yet inaccurate guidance, while coachness without liveness provides relevant guidance only after players manually describe their situation. When both dimensions were present, participants received specific help grounded in what the system knew about their current gameplay. These results highlight that preferences for assistance vary widely, illustrating the need for systems that adapt to individual thresholds of wanted assistance within games and beyond.

Para mi Mamá y mi Papá.

Acknowledgements

I would like to first acknowledge my advisor, Adam M. Smith, for assisting me throughout my entire masters degree program. From helping me understand how to start programming in Java back in 2018 to assisting me in my own videogames research interests in 2024, Adam has supported me greatly on how to become a confident programmer, researcher, and collaborator. Even though I had no clue what to do when beginning my masters, Adam helped me find a spot within the Computational Media department that I can call my own for the work that I have done. I greatly appreciate all the valuable knowledge, skills, and support that he gave me for my AI and games research. The work I have done would not have been possible without his support.

I would also like to acknowledge my fellow lab members of the Design Reasoning Lab, the CMPM 200 sequence instructors Elin Carstensdottir and Kathryn Ringland, and all my fellow students in the Computational Media department at UC Santa Cruz for their shared and collaborative experiences. Interacting with my cohort on varying topics has been a wonderful and expansive experience that made it feel welcoming. Specifically, I would like to thank Dipika Rajesh for supporting my research topic on videogames as I would have not been able to extend as much as it is now without working with others who share the same vision and passion.

Lastly, I would like to acknowledge my friends and family for all of their support. They helped tremendously in my own development of myself, as well as keeping me sane through fun entertainment and conversations. Specifically, I want to acknowledge my partner Kayla Celest C. Taduran. She supported me throughout my entire degree, pushing me to develop a career I thought to be impossible. I cannot express enough the gratitude I have for her love, support, and patience when completing this program.

Chapter 1.

Introduction

> *I was very excited to do battles in Dark Souls, and then I couldn't find where I was trying to go, and then I got bored and stopped playing it, to be totally honest. - Participant 2*

> *I like to follow the game designer's way as much as I can, but if there are instances where I am stuck and I would like to not be stuck, I tend to use external resource. - Participant 10*

Imagine playing a videogame while a friend is sitting next to you. As you play, there are moments where you struggle to make progress and where you wonder about certain aspects of the game. The friend suggests checking back on a room you might have overlooked through a mix of gesturing and speaking. They also clarify about moments in the game that seemed interesting to you, such as lore and extra content. The friend was able to infer that you needed help from your behavior and curiosity over time and did not require a precise question to intervene. Their guidance is effective because it is grounded in the context of your gameplay session at the time of intervention. By knowing about your gameplay preferences, they deliver that styled guidance that is informed by their prior experience with the game.

Compared to fixed strategy guides or static video walkthroughs, the friend exhibited two key qualities. They used their awareness of your current gameplay for providing

relevant and accurate intervention, a property I call **liveness**. They also used their deep knowledge of the game to provide you assistance based on your defined goals and preferences, a property I call **coachness**. These qualities define a design space for dynamic player-assistance systems I call **Live Coaches**. Before players can be given assistance on their current in-game situation, they have to sift through various methods and sources in order to identify where they are in the greater context of the game. Previous forms of player-assistance are able to exhibit liveness and coachness such as gameplay counselors and game manuals [2, 3]; however they are not able to achieve high degrees of both qualities. They lack contextual and adaptive support on the player's current situation. Live Coaches aim to provide that wanted assistance.

This thesis presents Live Coaches as a **genre of AI software** for adaptive, integrated player-assistance. Rather than players disconnecting from the game to get assistance, Live Coaches provide it using integrated methods that keep the current game context in mind. Players are able to get help that respects their goals, (e.g., want to avoid spoilers, receive minimal help) while staying in the flow of the game. The Live Coach project is built on the idea of creating real-time, specialized help based on player preferences. My thesis is motivated by a few questions:

- How do we augment player-assistance methods towards personalizing high-context guidance?
- What design tensions emerge when balancing the design space of liveness and coachness?
- How do current Live Coach implementations mediate tensions between intentional game design and arbitrary player preferences?

The two quotes above regarding Participant 2 and 10 demonstrate different player experiences. Participant 2 drops the videogame as a result of being lost on progression. Participant 10 initiates wanted assistance based on their gameplay. These player frustrations originate from the game's lack of support in resolving the player's challenges. Without proper assistance, players may quit the game entirely as a response from the frustrating scenario [4]. Many videogame players range on their level of desired assistance, meaning there is no one-size-fits-all method. Players also range on when to initiate getting that help, whether it be after a few minutes or a couple hours. I view Live Coaches as a way of providing high-context assistance within that variety rather than step-by-step assumptions within player guides and walkthroughs.

By analyzing existing systems that exhibit varying degrees of liveness and coachness, I present two implemented Live Coach systems: **ChozoMaps** for *Super Metroid* and **Rotom** for *Pokémon Red*. Additionally, I identify design patterns within the existing systems that demonstrate how to create future ones within the genre. For recognizing the impacts on integrated player-assistance, I also present findings and insight of the ChozoMaps system in action using an ablation user study.

The main contributions of this thesis are:

- **Live Coaches** as a genre of integrated AI software for real-time player-assistance.
- **ChozoMaps** and **Rotom**, two newly created Live Coach systems for *Super Metroid* [5] and *Pokémon Red* [6] respectively.
- A **design framework** defined by key patterns of existing Live Coach systems.
- Findings from a user study of 10 participants, 4 of which interacted with ablations of ChozoMaps, focusing on the impacts of the system.

Chapter 2.

Background

This chapter goes over three main areas that connect with the thesis: videogame assistance, artificial intelligence (AI), and human-computer interaction (HCI). I provide background information for how players go about getting player-assistance, how AI is used for real-time context awareness, and for better understanding the player-Live Coach interaction.

§ 2.1. Videogame Assistance

Challenges in videogames are a major design component for motivating players, described by prior work as a key element of play for enhancing player experiences [7–9]. Despite this, players may have too much difficulty on the challenges. Consalvo describes that a common form of cheating results from players unable to progress further, turning to guides and friends to get past the difficulty [10]. With that, I discuss about broad player-assistance approaches within the field.

2.1.1. Modes of Support

Player-assistance in games span across a broad space, ranging from integrated features (e.g., tutorials, adaptive difficulty) and external sources (e.g., walkthroughs, online communities). Prior works show that forms of assistance I describe to be tasteful cheating, such as aim assist, make games more accessible rather than easier [11–13]. Other videogames provide integrated assistive features, such as *Celeste*'s Assist Mode, *Left 4 Dead 2*'s AI Director, and Dynamic Difficulty Adjustment (DDA) for personalizing player experiences [14–17]; however, not all games provide those types of support.

Past literature illustrates that players have a variety of skill levels that makes it difficult to balance game design, with player modeling approaches being ways of addressing that concern [18, 19]. Additionally, Zhu and Ontañón's work illustrate the importance of player-centered perspectives regarding game personalization, arguing for a bigger focus on player needs and intent [20]. A major limitation of these approaches is its focus on future game design rather than pre-existing videogames. As a result from past games lacking the support, players tend to gravitate towards external sources as a replacement, such as player guides, walkthroughs, and game modifications.

External sources serve as knowledge sources that games do not provide themselves. Consalvo describes player guides as guiding players within a game from start to completion, with sections segmented for searchable help [10]. They support players who aim to independently complete the game, and refer to a guide briefly for getting help at a particular moment. Giddings describes walkthroughs as hybrid text, providing multimodal assistance that provides text and visual-based guidance for stuck players, where context is necessary for assessing the type of help [21]. Gameplay counselors, such as Nintendo's toll hotline services [2], support players through resolving unstuck situation

in videogames and curiosities players have (e.g., how to catch a Mew in *Pokémon*) using an internal knowledge base [22, 23].

A limitation of the three is the disconnect from the game as players must step away in order to get their desired help. Additionally, they require player context for situating their specific help, potentially causing the player to receive more than what was wanted and resulting in spoiling or cheating out their own gameplay discovery.

2.1.2. Integrated Game Modifications for Help

Player-created mods represent an area of integrating player-assistance beyond what game developers provide. UIInfoSuite for *Stardew Valley* tracks real-time game information through UI overlays for helping players be aware of current in-game events [24]. Say the Spire for *Slay the Spire* offers screen-reading support for accessibility [25]. JourneyMap for *Minecraft* displays an interactable minimap where it tracks the player's real-time position for world exploration and player tracking [26]. These systems demonstrate strong forms of real-time integrated assistance that could be described as high degrees of liveness. However for degrees of coachness, the mods remain limited as they provide static information rather than adapting guidance to player-specific goals.

More recent mods integrate AI-driven systems directly into gameplay for providing assistance. Herika for *Skyrim* is an in-game NPC companion for dynamic, conversational interactions [27]. Using a large-language model, Herika is able to interact within the world and recognize current game states for supporting the player's experience. Similarly, LLM-based agents in *Minecraft* are used for collaboratively completing a minigame quest alongside the player to observe agent behavior [28]. In the game space, agents are primarily used for keeping the player on track towards quest comple-

tion. While these systems enable dynamic interactions with players, they are typically embedded as in-world entities rather than dedicated systems for player-assistance.

Despite it being an official feature, Zelda Notes for the Nintendo Switch 2 versions of *The Legend of Zelda: Breath of the Wild* and *Tears of the Kingdom* behave as an integrated companion service alongside the player's gameplay on their mobile device [29]. The service provides navigational assistance, extra world content, and community sharing support that helps personalize the player's experience, with online forums describing them as a real-time strategy guide for guided exploration [30, 31]. These three systems and game mods are forms of tasteful cheating that aim to enhance player experience rather than hinder player skill. All three AI systems are integrated within the game, however they remain reactive on their guidance; Live Coaches aim to provide both reactive guidance and proactive interventions towards player-driven assistance.

§ 2.2. Assistive Artificial Intelligence Systems

Artificial intelligence (AI) has been used for systems to interpret current user scenarios regarding assistance, both inside and outside of videogame contexts. The Talin framework by Aytemiz et al. integrate dynamic tutorial design for game designers, where the system uses the skill atoms theory to monitor real-time player skill and performance [32]. Tutorials are dynamically modified for respecting the player's skill level, prompting help only when appropriate. DDA approaches provide adaptive player difficulty in games for supporting player game flow based on their ongoing skill level [17, 18]. Sensors are used for monitoring player statistics towards player modeling, with effectors proactively and reactively adjusting the design influenced by the recorded player data.

2.2.1. Direct Control Assistance

Aytemiz et al. repurposed AI methods into game-playing assistance that allows for games to be more inclusive [33]. Aim assistance helps players hit their targets, with prior work validating improvement on player performance and experience without hindering skill development [11–13]. While not an AI system, Nintendo's Super Guide provides pre-recorded footage in certain games for demonstrating level completion, where players can resume play at their discretion [34]. Additionally, Mukao and Asuke discuss how footage might be perceived by players, ensuring that the assistance must be grounded in actions that players can reliably perform [35]. These demonstrate effectors of AI assistance aimed to preserve player agency while not hindering player skill acquisition.

2.2.2. Adaptive Gameplaying

Connected with previously mentioned player-assistance, Green et al. discuss about adaptive tutorials, using an AI framework that generate text instructions and gameplay examples [36]. Aytemiz et al. create a framework for developing dynamic tutorials based on equating player modeling to skill atoms [32]. Robertson defines the concept of procedural regeneration, a method of adapting level design based on ongoing player actions [37].

Moving into player modeling, Hunicke and Chapman use Dynamic Difficulty Adjustment for enacting defined actions and policies based on player analytics [17]. Dormans and Bakkes use player modeling for generative grammars towards adaptive play experiences [38]. Prior work illustrates how dynamic experiences require real-time player context, gathered from continuous observation for providing personalized gameplay.

2.2.3. AI Assistants

Numerous intelligent systems have been developed for scaffolding assistance. Outside of videogames, Intelligent Tutoring Systems (ITS) provide educational content through adaptive feedback, hint, and recommendation generated based on the learner's preferences and needs[39]. O'Rourke et al. explored generating instructional tutorials for scaffolding student learning through a framework [40]. Using the Thought Process Language in an educational-based game *Refraction*, generated assistance is provided at appropriate intervals for scaffolding student understanding of fraction concepts (e.g., granularity of hints based on mastery, minimal help).

Returning back to videogames, Zhang et al. create an ITS system for *Gomoku*, trained on an AlphaZero model [41]. The system supports a range of teaching methods for players learning the game, such as immediate feedback, on-demand hints, and after-action reviews. He et al. trained a LSTM model for creating adaptive hints within puzzle games [42]. Through adaptive hints, players were shown to have improved gameplay experiences. An AI coach for *Apex Legends* provides post-game adaptive coaching suggestions based on player modeling [43]. The conversational agent for *Hades*, by Lin et al., offers personalized and accurate advice based on the current game context through use of vision-language models [44]. NVIDIA uses ACE AI models for enabling context-awareness within AI assistants [45]. Through direct integration with the game, NVIDIA enables assistants for low-overhead, deictic interactions with in-game players. Similarly, Xbox's Gaming Copilot advertises as a personal gaming sidekick that provides players in-game assistance and personalized coaching at the player's discretion [46]. Online discussion regarding Gaming Copilot focuses on its convenience and access to current game activities as being helpful for avoiding immersion-breaking [47].

These systems demonstrate a variety of sensors and effectors connected to the AI agents. Creating adaptive assistance requires a representation of the player's in-game world, where static approaches fail to support the player's preferences on help. As such, agent systems exhibit degrees of liveness and coachness, showing a desire for enhancing player experiences through tailored assistance.

2.2.4. Knowledge Bases

Rabin discusses that it is impossible to develop an AI system that is able to handle every game situation [48]. One solution to address this is integrating the intelligence into the game world for simple, yet rich game actions. This same principle can be applied to knowledge bases as a way of grounding player-assistance.

Aytemiz et al. create a skill atoms knowledge base for capturing player mastery of certain game mechanics [32]. Mawhorter's advice oracle utilizes a Binary Decision Diagram (BDD) for a compressed representation of the game's state space [1]. As the BDD contains every pre-computed shortest path, it allows assistance to be quick, responsive, and adaptive to the player's short and long-term goals. Lin's *Hades* tutor uses Retrieval-Augmented Generation (RAG) for drawing into online sources related to the game [44]. Retrieved documents are encoded within a model, used for future search and analysis related to player queries.

Adaptive player-assistance is shown to be researched on, with an emphasis on inclusive gameplay for future game design. Support for player goals and preferences demonstrates to be a field of interest within assistive AI agents. While most agents provide text-based assistance, non-verbal communication is an overlooked aspect of desired assistance. Some agents do not demonstrate high degrees of coachness, focusing primarily on game

completion-related assistance. Furthermore, some systems do not demonstrate high degrees of liveness, showing a lack of system agency to speak out of turn. These gaps within systems, alongside player-centered assistance based on current game states, are where Live Coaches occupy.

Chapter 3.

Live Coaches as a Genre

This chapter details a more concrete definition of what a Live Coach system is, defining what it means to exhibit **Liveness** and **Coachness**, and creating the groundwork for becoming a newly defined genre. As a basis for discussion, this chapter also analyzing previous system implementations focusing on player-assistance that are defined to have characteristics akin to Live Coaches. This definition was created and developed after extensive discussions with Dipika Rajesh and Adam M. Smith.

§ 3.1. Definition

We define Live Coaches as a genre* of integrated AI player-assistance systems capable of providing high-context adaptive support during gameplay. The genre's core definition is based around two design dimensions: **liveness** and **coachness**. These dimensions define a design space where Live Coach systems are situated and compared with previous forms of player-assistance (Figure 3.1).

*The term *genre* is used in the same sense as Compton's use for defining the genre of Casual Creator systems [49]. Software genres work as interpretive lenses or guidance for creating new systems.

Systems within this genre exhibit recurring characteristics: continuous game state monitoring, player-aligned guidance, and intervention mechanics. With the design space, it allows for deeper analysis on previous systems for distinguishing Live Coach systems with adjacent player-assistance approaches (e.g., static walkthroughs).

§ 3.2. Liveness

Liveness is the ability to maintain real-time contextual awareness of the player's gameplay with a level of intervention for providing assistance. From a technical standpoint, this emerges from integrated sensors and effectors that enable communication between the AI and the videogame for up-to-date game information. Liveness ranges from passive monitoring to active intervention of gameplay (Figure 3.1).

A system with a high degree utilizes game context and support more forms of intervention during gameplay, such as aim assistance on automating small adjustments for precision and Mawhorter's advice oracle [1] for generating paths based on a player-chosen destination. Additionally, high degrees of liveness are able to support player's deictic requests as context is provided beforehand.

§ 3.3. Coachness

Coachness is the ability to leverage deep knowledge of a game to adapt to the player's goals, while acting to improve a player's effective skill and understanding of a game. Specifically, it is able to use available depths of the domain knowledge to align assistance with player-specific goals. In addition, it develops an agenda for cultivating player skill

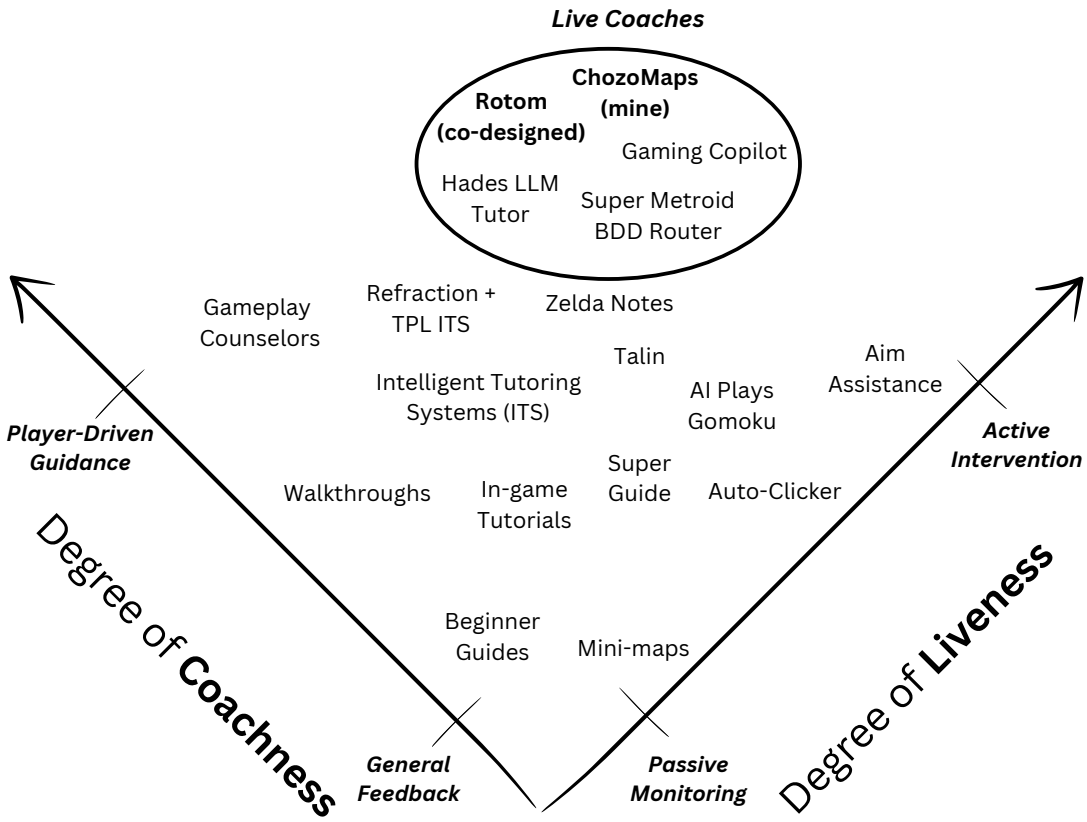


Figure 3.1.: The design space of player-assistance methods across degrees of **liveness** & **coachness**. Past and current systems are positioned by both degrees. **Coachness** ranges between general feedback and player-driven goal support, and **liveness** ranges between passive monitoring to proactive intervention. Live Coaches are placed in this space as to showcase a range of high degrees of both dimensions within the genre.

acquisition and game understanding. From a technical standpoint, this emerges from the system's main knowledge bases for extracting key information in support of the player's curiosity. Coachness ranges from general feedback on game completion to player-driven assistance (Figure 3.1).

A system with a high degree does more with deep knowledge beyond game completion; they are able to mold it within the player's defined goals and playstyle that is befitting to them, such as wanting to know how to collect optional collectibles or wanting to learn more about the game's lore.

§ 3.4. System Examples of Live Coaches

For grounding Live Coaches as a genre, we analyze three existing systems that exhibit characteristics of high liveness and coachness. With Dipika Rajesh's assistance, we identify design patterns that are common within those systems (Table 3.1).

Mawhorter's advice oracle is one example of a Live Coach [1]. It utilizes a BDD structure to provide spatial assistance based on the current policy. The authors encode the non-linear game space of *Super Metroid* as deep knowledge, with next-step recommendations as the representation for player-assistance. Through integrated sensors and effectors, the system reads game RAM alongside the compressed game space for maintaining game state updates. The default policy generates assistance towards game completion, with players dynamically changing the policy through choosing a room on an interactive map. The advice oracle exhibits high liveness and coachness by facilitating long-range route planning based on the dynamic policy, providing real-time visual guidance and continuously monitoring the player's game state.

Another example of a Live Coach is Lin's AI tutor for *Hades* [44], providing real-time strategy suggestions connected to gameplay. The authors focus on creating a conversational agent that has deep knowledge of the videogame and understands their current game state to provide accurate assistance to the player's preferences. The Large Language Model (LLM) allows for players to have natural-language conversations on the given player-assistance. For inferring on the game's current state, they connect a Vision-Language Model (VLM) and send game screenshots for situational reasoning (e.g., what choice of boons to take offered in the screenshot). For creating their knowledge base, they use Retrieval-Augmented Generation (RAG) to draw onto online sources that contain guides and strategies such as r/HadesTheGame and speedrun.com/hades. This allows for assistance to be grounded and accurate into what has been previously discussed before, allowing for the agent to reason with expert knowledge in the game area. The AI tutor displays both high liveness and coachness by personalizing advice to the player, comprehending game context through visual screenshots, and deriving deep knowledge for accurate game assistance.

Microsoft's Gaming Copilot (Beta) [46] also functions as a Live Coach. One key distinction between Gaming Copilot and the previous two system examples is their focus; Gaming Copilot aims to be generalized for all available Xbox games on the system, while the previous two focuses on their respective games *Super Metroid* and *Hades*. As a proprietary system, its underlying sensors, effectors, knowledge bases, and AI engines are not publicly disclosed. In which case, analysis is focused on its observable behavior from public sources, where it exhibits characteristics of a Live Coach. The system infers the player's current state through game screenshots and access to the player's account. It also assists the player in a range of wanted help, such as completing in-game objectives

	Advice Oracle	**Hades LLM Tutor**	**Gaming Copilot**
Videogame	Super Metroid	Hades	Any Xbox-available game
AI Engine(s)	Binary Decision Diagrams (BDD)	Large Language Model (LLM), Vision Language Model (VLM), Retrieval-Augmented Generation	LLM
Knowledge Base	Precomputing and memorizing every possible shortest path based on an abstraction of the game	r/HadesTheGame subreddit, Hades on speedrun.com, authored system prompts	Authored system prompts, player account data
Sensors and Effectors	Reads game RAM, external interactive map, draws lines on map and game screen	Reads game RAM, takes live screenshots of game state to pass into VLM, chat interface	Reads live game screenshots and player account, chat interface, voice support, hyperlinking sources

Table 3.1.: Design patterns discovered of three previous Live Coach systems. This breakdown compares the respective game, AI approaches, knowledge bases, and sensors & effectors for their approach on player-assistance.

and supporting player interests (e.g., being recommended games based on play history). The system exhibits high liveness and coachness through supporting player curiosity and assistance, as well as understanding the player's current game state for giving relevant responses grounded in what the player is doing.

These three systems illustrate different Live Coach implementations through various knowledge bases, AI engines, sensors, and effectors. Whether it be for game-specific or general purpose, Live Coaches operate through high-context interactions dependent on being continuously aware of the current game state and inferring on deep game knowledge. By having these features, players receive catered game assistance to their preferences and current game context that enhances game flow without disengaging players from their current gameplay. Through observing these common design patterns that Live Coach systems have showcased, aspects of the systems are identified for how Live Coach systems could be designed for other videogames and media.

Chapter 4.

Live Coach Development

I present two Live Coach systems for distinct games: ChozoMaps for *Super Mertoid*, designed by me, and Rotom for *Pokémon Red*, co-designed with Dipika Rajesh. *Computational caricatures* [50] is a design research practice aimed to capture and exaggerate technical claims about AI in the form of computational systems, such as game design. In this thesis, we represent these systems as computational caricatures of player-assistance, demonstrating different kinds of assistance to illustrate high degrees of liveness and coachness rather than optimizing for evaluation.

This chapter details how Dipika and I analyzed and implemented Live Coach systems for their respective games. I will also go over how both systems exemplify high degrees of liveness and coachness on player-assistance. The Game Analysis and System Design categories were defined after the ChozoMaps and Rotom system implementations. They are used retroactively for a structured understanding of the Live Coach development. Afterwards, a design framework was defined as a way for future Live Coach implementations (Chapter 5). Both Live Coach projects were developed under the same research lab (Design Reasoning lab) with the assistance from our advisor Adam M. Smith.

§ 4.1. Computational Caricatures

Smith and Mateas define *computational caricatures* as a design practice for exploring the role of a system within the game design process [50]. The practice aims to capture and exaggerate the system's claim in a more recognizable view, allowing for more rigor analysis and future re-usability. We do not develop ChozoMaps and Rotom as a commercial product; rather, they are systems exploring the ideas of liveness and coachness in apparent ways. With that, we use the computational caricature design process for making both degrees highly legible, even exaggerated, in the Live Coach design process for understanding their impacts on player experiences.

§ 4.2. ChozoMaps

Super Metroid [5] falls under a subgenre of videogames titled metroidvanias, a subgenre of platforming action-adventure games focused on non-linear exploration and discovery of new items and mechanics [51–53]. Prior work identifies the game's key gameplay challenge to be spatial navigation as its non-linear gameplay nature creates a player-driven environment for destination setting [1, 54, 55]. This section goes into detail over how I:

- Analyzed *Super Metroid*'s gameplay structure for identifying modes of assistance.
- Developed **ChozoMaps** (Appendix A.1), the Live Coach system for *Super Metroid*.
- Discussed the design implementations and integrations of the system.

4.2.1. Game Analysis

To ground the integrated game assistance, I analyze *Super Metroid* for understanding what forms of assistance are appropriate. This will provide moments of time during gameplay where Live Coaches are able to intervene.

Failure Support

I use the Taxonomy of Failure Framework [56] for recognizing the types of in-game and out-of-game failures within *Super Metroid*: Encoding Input (What to physically do for game actions), Decoding Output (How to understand the game information), Discovering Mechanics (What to do in game), Setting Goals (What goal need to be accomplished), Planning (How to accomplish the goal), and Execution (Were the correct game actions performed).

Out-of-game failures in *Super Metroid* occur when players do not recognize the need for missiles (Setting Goals), misinterpret doors (Decoding Input), or how to use their upgrades (Encoding Input), with some failures demonstrated by playtest participants on the ChozoMaps system (Section 7.4). In-game failures occur when players decide on incorrect destinations (Planning), mistime jumps (Execution), or are orienting themselves on what they can do at certain areas with the items they have (Discovering Mechanics). These help define aspects of the ChozoMaps design for providing game assistance. Decoding output, discovering mechanics, and execution require real-time intervention for understanding the game context (liveness), while setting goals, planning, and encoding input require short and long-term reasoning (coachness).

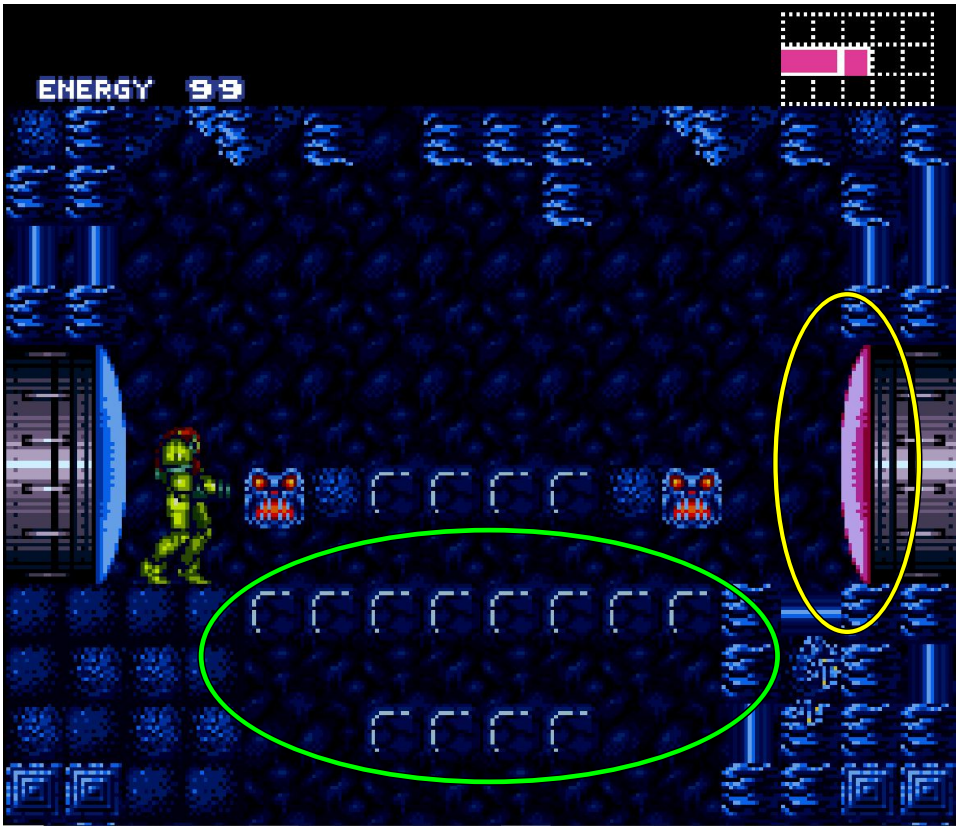


Figure 4.1.: A screenshot of the game demonstrating two out-of-game failures. A Decoding Input failure (denoted as a green circle in the center) occurs when players, such as P1, cannot parse the game information on screen, blending in with other blocks and looking similarly to a bottomless pit. An Encoding Input and a Decoding Input failure (denoted in a yellow circle to the right) occur when players, such as P9, try to shoot one missile at the red door. Players may fail to understand that 5 must be shot at for opening those doors as it fails to be intuitive. At this point, players are unaware how the resource is given and are afraid to waste resources they found. Participant reactions can be found in future sections (Section 7.4).

Curiosity and Exploration Support

Prior work highlights that curiosity and discovery plays a significant role in motivating player engagement within the *metroidvania* genre [55, 57, 58], alongside participant demonstrations of curiosity towards objects and upcoming boss fights in *Super Metroid* (Section 7.6.1). With the genre having backtrack exploration be a key pillar of gameplay, this requires high degrees of coachness for assisting in the player's curiosity. Ranging from hidden areas to optional upgrades to curiosity on certain parts of the game, ChozoMaps must support those aspects for players on their player-driven journey.

Design Tensions

It must be acknowledged on how player-assistance is delivered to players for *Super Metroid*. Connecting back to the Taxonomy of Failure Framework, in-game failures consist of planning, execution, and discovering mechanics as these are the key components for metroidvania gameplay. As the game design focuses on platforming elements and non-linear exploration, there are intentional failures for the gameplay experience. These align with the analysis Hoek et al. of breaking down *Super Metroid*'s spatial navigation into three challenges: Destination, Routing, and Execution [54].

Destination involves players interpreting game information and deciding where their destination is (e.g., where to go after getting missiles). Routing involves players creating a sequence of actions for reaching their destination and knowing if they are possible (e.g., what to do with bombs). Execution involves players to enact the appropriate sequence of actions for game progression (e.g., how to platform through the escape shaft). I identify that tensions can arise from guidance versus exploration, as well as real-time assistance versus distractions. I must acknowledge how to balance intentional game design and

player agency with wanted player-assistance. Poorly timed interventions can disrupt player flow when deciding and enacting game actions, as well as excessive assistance can disrupt player agency for players wanting to engage with the game independently; mentioned by participants during ChozoMaps playtests (Section 7.7).

4.2.2. System Design

After analyzing the game on potential avenues for providing player-assistance, I go into detail for how ChozoMaps was developed for supporting those forms of assistance.

The ChozoMaps system is designed on a static HTML webpage (Figure 4.2). The client side contained a web assembly version of the *snes9x-next* emulator [59], a large-language model (LLM), and an interactable leaflet map. The server side of the system contained a black-box version of Mawhorter's advice oracle [1].

AI Engine(s)

I chose to use a LLM for its capability of synthesizing game information and providing contextual assistance in natural language. *Claude Plays Pokémon* demonstrate how language models are able to reason during gameplay through interpreting game state, goal planning, and enacting actions within the game environment [60].

A major tension, however, is the possibility of harmful content generation from the LLM [61, 62]. Providing false or misleading information, in the context of player-assistance, can result in player frustration towards the system and game. For addressing some tensions on providing accurate and non-disruptive guidance, I author the system prompt for giving flexible and appropriate help (e.g., succinct messages) aimed to provide reactive and proactive guidance based on the current scenario (Appendix B.1).

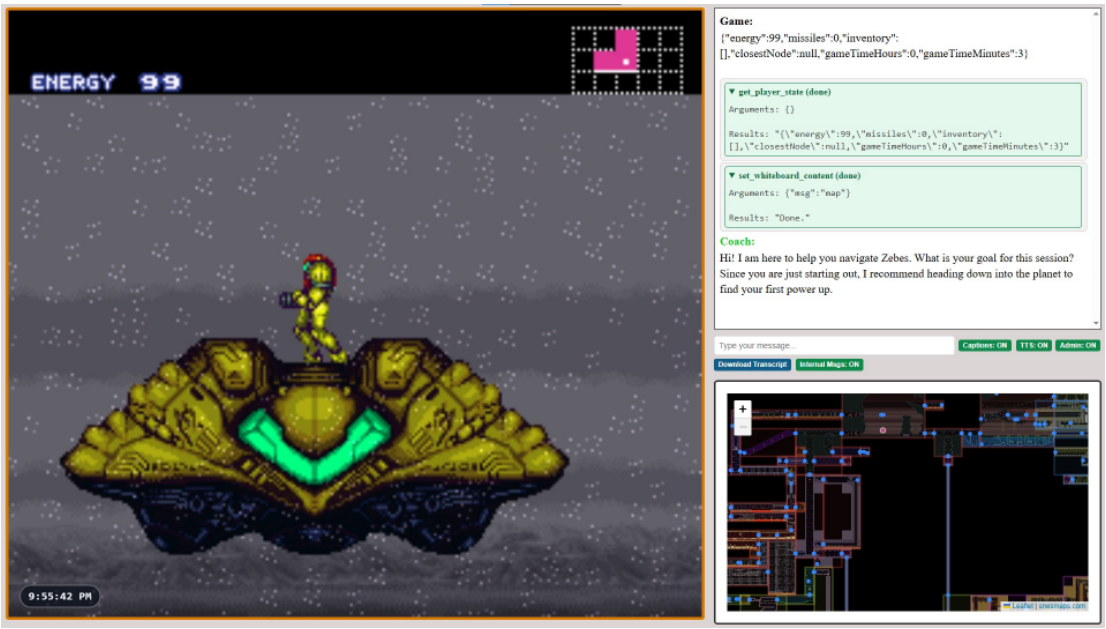


Figure 4.2.: A screenshot of the ChozoMaps system for *Super Metroid*. Game updates are provided to the system on-launch, allowing the system to introduce themselves to the player. A leaflet map is displayed in the whiteboard for providing visual-based assistance. Game and Function Call results are primarily shown to demonstrate internal works of the Live Coach.

The LLM model chosen is Gemini-3-Flash-Preview* for integrating the model with a browser-based Super Nintendo emulator *snes9x-next* [59]. The model was chosen for its support on function calling, long context windows, and visual perception, alongside balancing the model's strength and responsiveness for ongoing assistance. With the Super Nintendo emulator and LLM, the system is hosted on a single webpage for a player environment that allows real-time interaction with the system alongside the videogame. Other features incorporated into the design were text and voice input from players. Depending on the player's preferred method of communicating to the system,

*This model is a proprietary, closed-weight model that is likely to become unavailable soon due to the rapid rate of model churning in the commercial generative AI space in 2026.

they are able to type or speak aloud. Some participants note of this style being easy to use and convenient for it being locally on one page (Section 7.4).

Knowledge Base

I use Mawhorter's advice oracle [1] as a black-box service for its generation capability of long-range planning based on the game's spatial structure, progression logic, and reachability constraints. As the advice oracle relies on RAM maps for in-game context, it is connected with the Super Nintendo emulator for reading into game memory. With the advice oracle, ChozoMaps will be able to generate short and long-term routing for a player-chosen destination, aligning with *Super Metroid*'s routing challenge of micro and macro-planning [54].

Micro-planning is defined as being confined of a single screen or room in the game world (e.g., jumping up platforms in real-time). Macro-planning is defined as larger scale in terms of required thinking about various locations for a destination (e.g., how to get to Norfair from Crateria).

The authored system prompt defines the LLM behavior into grounding the system's reasoning as a Live Coach for game mechanics, assistance, progression structure, and player context (Appendix B.1). With the prompt, ChozoMaps provides contextual help based on prior knowledge through natural language, as mentioned by some participants regarding convenience (Section 7.6). Additionally, I integrated a function call for giving simplified cardinal directions based on where the player is to their goal. Rather than being provided the next room node name, instead it gives spatial-awareness directions (Figure 4.3) for micro-planning support. With the friend scenario (Section 1), they have the capability of telling you to head left from where you are in order to reach your goal.

Sensors

I needed techniques of having the Live Coach to contextually understand the player's current situation at all times, meaning to provide in-game state representations over a period of time. For macro-planning support, ChozoMaps reads game memory addresses through continuous game updates containing the player's location, inventory, in-game time, and character statistics (e.g., energy, missiles). Any time one of the parameters changes, an update is sent over to the system for the system to passively track the player's ongoing gameplay (Figure 4.2). For micro-planning, the system has access to take a screenshot of the emulator game canvas and send it over to the LLM for interpretation. Through a screenshot, the system assess what the player sees on-screen for deictic support (Figure 4.3). With the friend scenario (Section 1), they passively understand the whole context of your game session as they have been with you since the start of the session; thus, any questions asked already have the context behind to support.

4.2.3. Effectors

I wanted to translate the AI reasoning into interactable assistance. I integrate two UI widgets for the webpage: a chat and whiteboard interface. The chat interface conveys results from the LLM to the player. Players are able to read assistance to the side of the game as they play. The whiteboard, in this context, is a delegated space for visual-based assistance (e.g., maps, controls, goals). The whiteboard has an integrated, interactable map that Ross Mawhorter assisted in translating from a Python script to JavaScript. The map is a Leaflet map that contains a simplified visualization of the game world and its rooms/nodes that can be interacted with to explore the space (Figure 4.2, 4.3).

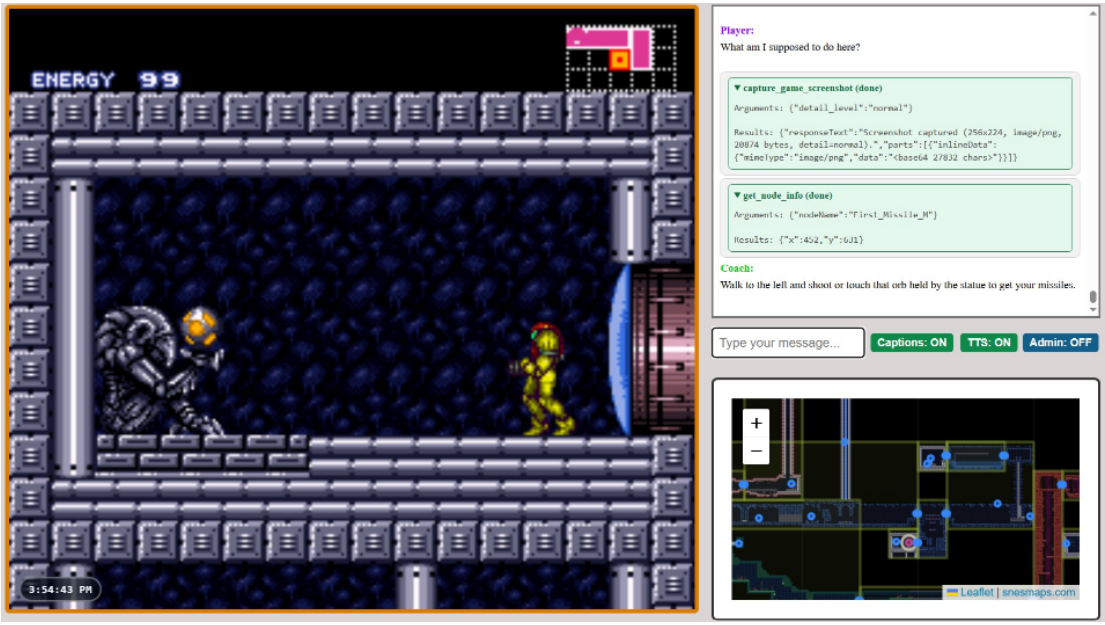


Figure 4.3.: As the player asks a high-context question about what to do in the room, ChozoMaps takes a screenshot to understand their question in-game. With a visualization, the system provides a micro-plan of shooting the orb and collecting the missiles.

Text-to-Speech is also integrated for more player options. Players range from what types of assistance they want, meaning not everyone will enjoy reading a chat interface. To support this, players are able to enable Text-to-Speech so they are able to play while being verbally told game assistance. With the friend scenario (Section 1), the ability to talk aloud with your friend and ask follow-up questions allows for assistance to be personalized since they know what style of answer you prefer. As a way of interacting with *Super Metroid*, the Live Coach also has save/load game state functionality. Emulators allow for games to be saved and loaded at the player's discretion, providing flexibility in player preferences. The system periodically saves game states and proactively rolls back the states at appropriate times or reactively from player-initiated requests.

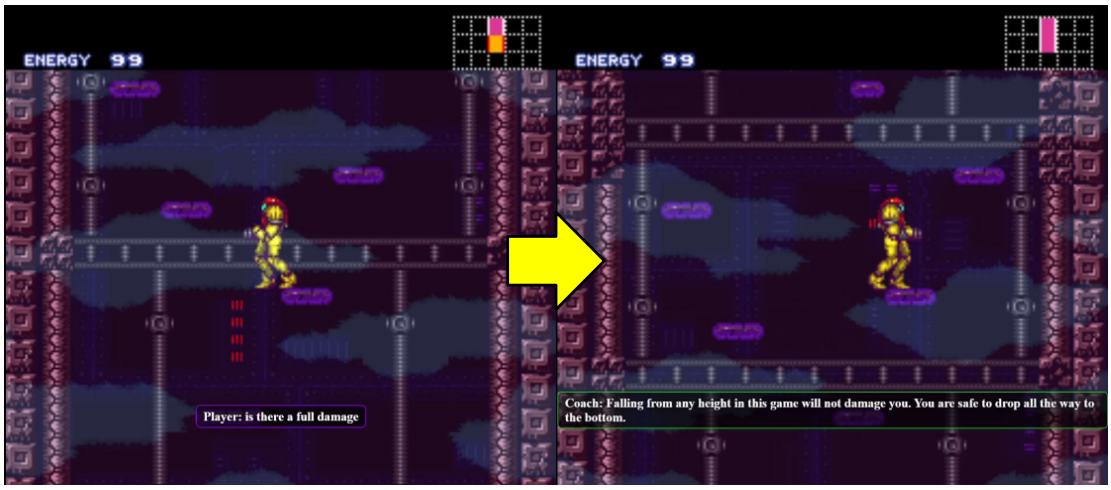


Figure 4.4.: A screenshot of Participant 10's interaction with the ChozoMaps system. This demonstrates the full-screen capabilities, with on-screen captions briefly displaying the current conversation. The yellow arrow signifies the transition from the player's question to the system's response. Captions disappear after 3 seconds as to not clutter screen space. The interaction also demonstrates support on player curiosity as P10 wanted to know if *Super Metroid* had in-game fall damage.

4.2.4. Additional Features

Additional features include quality of life features for player and developer purposes. As players might prefer having the videogame take more screen space, fullscreen gameplay was supported. One problem arises: if the player is on full-screen mode, they no longer have access to the widgets. To circumvent that playing preference, on-screen captions were integrated such that players are able to interact with the Live Coach as they play on fullscreen-mode (Figure 4.4).

As the developer, I wanted to see game updates passed to the Live Coach, the function calls that the system enacts, and the results of those calls. I implemented two features for viewing the data: a downloadable chat transcript and a toggleable button for viewing

them in the chat. A downloadable transcript allows us to save the conversation between the player, game, and system as a way to reflect what was talked about and look into the system for debugging purposes. Similarly, I wanted to view the data during prototyping, but disable it for playtest purposes. The chat interface allows for toggleable content, where function calls, results, and game messages can be displayed when desired (Figure 4.2).

§ 4.3. Rotom

Pokémon Red [6] falls under the genre of turn-based role-playing games (RPGs), where players are able to capture and train a variety of eponymous creatures called Pokémon. Each Pokémon bring unique movesets and abilities that define their identity, adding complexity in game battles and diversity in progression. Prior research has used *Pokémon Red* as a testbed environment for AI challenges, such as multi-agent systems on battle optimization [63] and deep learning agents for completing game segments [64]. This section goes into detail over how we:

- Analyzed *Pokémon Red*'s gameplay structure for identifying modes of assistance.
- Developed **Rotom** (Appendix A.1), the Live Coach system for *Pokémon Red*.
- Discussed the design implementations and integrations of the system.

4.3.1. Game Analysis

Similar to ChozoMaps, we analyze *Pokémon Red* for understanding what forms of assistance are appropriate. As the game is distinctly different from *Super Metroid*, not

all assistance identified will be applicable to Pokémon.

Failure Support

The Taxonomy of Failure [56]) is used once again for analyzing the types of game failure. Prior work highlights key pillars of *Pokémon Red*'s gameplay as strategic battling and resource management [64]. Failures in *Pokémon Red* fall under Setting Goals (e.g., route planning), Planning (e.g., battle strategies), and Discovering Mechanics (e.g., hidden collectibles). These identified failures define the Rotom system to emphasize sequential reasoning and long-term planning.

Curiosity and Exploration Support

We recognize Pokémon collection, battle compositions, and spatial navigation as the key aspects of player curiosity and exploration. Prior works identify that exploration comes from planning and navigation, where decisions are made regarding battle compositions [64]. The Pokémon franchise, as a whole, promotes an overarching goal within the games to complete their Pokédex, with the famous quote "Gotta Catch 'em All!" placed on the front cover of *Pokémon Red*. Curiosity originates from the game space having a large policy, with 151 Pokémon species available for diverse team compositions. Additionally, the game contains hidden items for players to collect as they explore around the world while pressing the interaction button. Rotom must support this policy space as curiosity and exploration are key aspects for how players create their journeys.

Design Tensions

In *Pokémon Red*, we identify tensions between information and discovery, as well as optimization and experimentation. Providing players with optimal strategies may undermine player learning with Pokémon compositions. Informing players of elemental interactions may prevent a stronger emotional connection compared to self-discovery of the mechanics. These tensions assist in understanding how much information should be delivered to players during gameplay. One way of addressing the tension is providing an interview phase that elicits player preferences such as spoilers, assistance intensity, and objectives for a more personalized player experience on assistance (Table 4.1).

4.3.2. System Design

The Rotom system is implemented in Python using PyQt6 and the PyBoy emulator for Game Boy support (Figure 4.5). Similar to ChozoMaps, Rotom uses a language model to emphasize open-ended communication and freeform exploration. A key difference is the type of model, using a small-language model (SLM) instead for sufficient reasoning capabilities and lightweight sizes on responsive interactions.

AI Engine(s)

Similar to ChozoMaps, we view how previous LLM-driven systems [60] show how they interpret game states and enact within the game world. Rotom integrates a SLM as the turn-based gameplay setting of *Pokémon Red* prefer richer reasoning and detailed suggestions over real-time intervention that ChozoMaps provides. Connected to the game emulator. Once again, a similar tension is present when using language models

[61,62], where incorrect information may lead to player frustration. We author the SLM system prompt, simiarly to ChozoMaps, for providing help in a productive manner. The model used is qwen3-next-80b-a3b-instruct [65] for integrating it with the PyBoy emulator [66].

Additionally, knowledge is constructed through encoding game walkthroughs as cases for a Case-Based Reasoning (CBR) system. Within the cases, they provide the system with action sequences, goals, and outcomes that can be used to inform the player-assistance. From connecting the SLM with a CBR system that maps the player's current

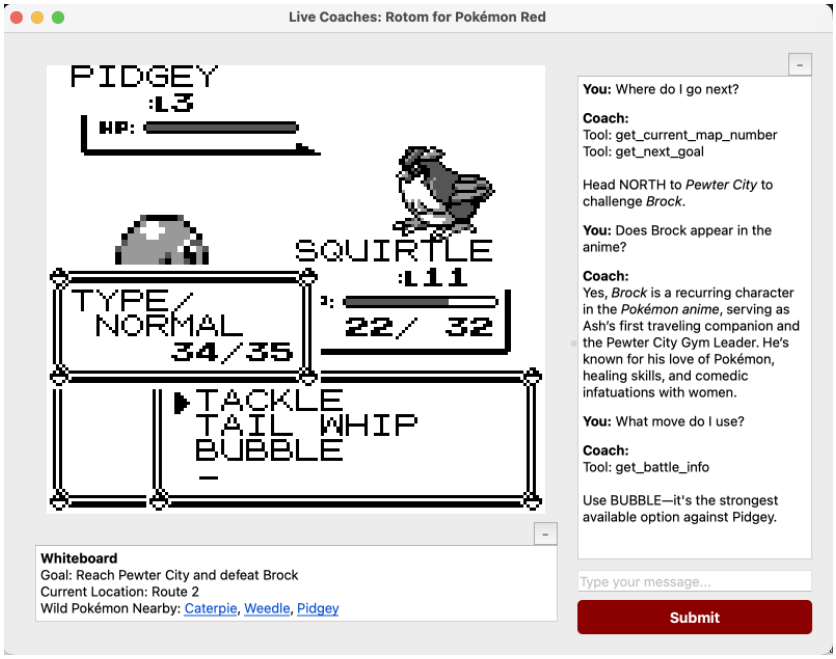


Figure 4.5.: A screenshot of the Rotom system for *Pokémon Red*. Through an integrated Game Boy emulator, features such as continuous updates through a whiteboard widget and a chat interface for coach conversations are supported. Rotom provides route-specific planning and goal-oriented guidance toward the player's next objective (e.g., beating Brock at Pewter City Gym), real-time battle assistance based on the current encounter, and optional lore information in response to player questions.

state to previous cases, Rotom supports strategic reasoning over complex game states and long-term planning.

Knowledge Base

As the CBR system contains encoded game walkthroughs, the system itself is a knowledge source. The system is primarily used for placing the player's current game situation (e.g., stats, party) on mapping relevant strategies and progression paths. Additionally, we author the system prompt to define the SLM's behavior as a Live Coach with the interview phase. As the game has a lot of information regarding each Pokémon, the system supports retrieving Bulbapedia wiki links for players to use as assistance (Figure 4.5, Table 4.1). Bulbapedia wiki links are provided as they contain comprehensive and reliable information spanning the franchise, such as abilities and evolutions.

Sensors

With the Game Boy emulator, sensors are connected through providing game memory addresses. Game information such as player location, party composition, battle states, inventory, and event flags are tracked for decision-making reasoning and long-term goals. As Pokémon has a large policy space due to the amount of character combinations, sensors assist in constructing a representation of the player's current state on creating applicable battle compositions, item management suggestions, and relevant game lore.

Effectors

Rotom's effectors are composed of UI widgets through a chat and whiteboard interface, similar to ChozoMaps. Within those widgets, the system provides information updates

Transcript	Annotations
Player: What should I focus on next for my team?	Intent: requesting advice
Rotom: Based on your preferences from the interview phase, you would prefer to be given some options when asking for help. Let me see what I can do!	Coachness: recalling player preferences
Tools invoked: get_current_party_full_stats, get_next_goal, populate_whiteboard, get_bulbapedia_link	Tool calls: retrieving context, routing plan, displaying sources
Rotom: Since your team has Squirtle and Caterpie, consider catching Pikachu in the Viridian Forest to add Electric-type coverage for Water-type opponents. You can also catch a Pidgey on Routes 1-3 for Flying-type coverage against Grass- and Bug-type Pokémon. Pikachu: stats, moves, and abilities (Bulbapedia link) Pidgey: stats, moves, and abilities (Bulbapedia link)	Liveness: state aware, hyperlink effectors Coachness: suggested actions, detailed advice
Player: Can you show me where to find these Pokémon?	Intent: visual reference
Tools invoked: populate_whiteboard	Tool calls: visual support
Rotom: The whiteboard now shows locations to find Pikachu and Pidgey. Use these areas to train and prepare your team efficiently.	Liveness: displays visual information Coachness: subtle guidance

Table 4.1.: Annotated transcript of Rotom, showcasing an interaction with a player. Rotom bases their advice based on a previous pre-game interview with the player as to gauge out their preferred assistance during the game session. Guidance is grounded in the Case-Based Reasoning, mapping out the player's current state of Pokémons. Assistance is also provided through interactable wiki links connected to Bulbapedia.

related to the player's current situation, such as available Pokémon in the current route. Additionally, the whiteboard supports integrated URL links connected to Bulbapedia (Figure 4.5) where it connects assistance to franchise material (e.g., Bulbapedia link to Jigglypuff on their abilities and stats).

4.3.3. Live Coach Reflection

After analyzing previous Live Coach patterns and developing two new systems, we developed a design framework to be used by future designers of Live Coaches. We expect that the framework informs designers on how videogames are analyzed for appropriate player-assistance, where Live Coaches can support within and beyond game completion,

and where they potentially cause tension with the game and player. Additionally, it will inform the designer on how to translate the analysis to a Live Coach system.

Chapter 5.

Design Framework for Live Coaches

After developing ChozoMaps and Rotom as two new Live Coach systems, Dipika and I define a framework on how to cultivate future systems within the genre. *Super Metroid* and *Pokémon Red* do not represent all games, meaning there are gaps not covered within these systems. Using identified patterns from past systems and observing our approaches to develop current systems, a **design framework** is made for systematically creating Live Coaches for a variety of videogames. This chapter goes over two main categories that are used for identifying opportunities for player-assistance within games and for translating those opportunities to the player.

§ 5.1. Game Analysis

Game Analysis focuses designers on analyzing the chosen videogame for identifying where they fail to support the player's goals. These provide insight on opportunities where integrated player-assistance are appropriate for enhanced player experiences. This includes analyzing the types of **Failure Support** and **Curiosity & Exploration Support** required for the system design.

5.1.1. Failure Support

Failure Support categorizes common obstacles that players face within the game that are considered to be unsupported from the game. Similar to how ChozoMaps and Rotom used the Taxonomy of Failure Framework [56] for identifying the types of failures, designers must anticipate where players may struggle and consider to seek out external help. Strategy-based games such as *Fire Emblem* have players manage resources such as units and goals, where creating a macro-level plan for a level can be difficult. Different failure types impose specific requirements on the system's design, where platformer-based assistance requires real-time intervention and resource-based assistance requires long-term reasoning.

5.1.2. Curiosity and Exploration Support

Curiosity and Exploration Support focuses on what players are naturally curious to explore around within game worlds. Prior work from Costikyan and Tang et al. demonstrate that curiosity in uncertainty plays a substantive role in motivating players within videogames [7, 57]. From Costikyan describing how uncertainty is not defined by a goal's outcome and can be found through many ways in the game, it illustrates that player goals may extend beyond game completion. Games such as *Pokémon Red* and *Dark Souls* may have players curious about exploring cross-media references and hidden content. In order to provide high coachness within these systems, they must support a variety of desired assistance, ranging from beating the game to finding optional content.

5.1.3. Design Tensions

Design Tensions focuses on recognizing tensions between gameplay, player agency, and assistance. Early context-aware assistants, such as *Navi* in *The Legend of Zelda: Ocarina of Time* and *Clippy* in Microsoft Office [67, 68], exemplify how misaligned and mistimed guidance lead to frustrating experiences. Designers considering these tensions can ensure integrations do not disrupt game flow and player agency.

Due to players having unique preferences on assistance, there may be certain methods that conflict from player to player (e.g., video walkthroughs). This scenario emerges between participants in future sections within the ChozoMaps study (Section 7.1.4). Additionally, knowledge bases should be sourced responsibly, aiming to respect community norms, intellectual property, and legal restrictions for appropriate use [69]. Thoroughly considering these dimensions will support the system design on how to create personalized assistance for player-defined goals.

§ 5.2. System Design

System Design focuses designers on actualizing the game's analysis into system components. Designers translate prior insights on the game into a component architecture broken down into four areas: **AI Engines**, **Knowledge Bases**, **Sensors**, and **Effectors** This section provides a structured guide on what each area entails for integrating high-context, real-time player-assistance (Figure 5.1), while showcasing how the framework compares to ChozoMaps and Rotom (Table 5.1).

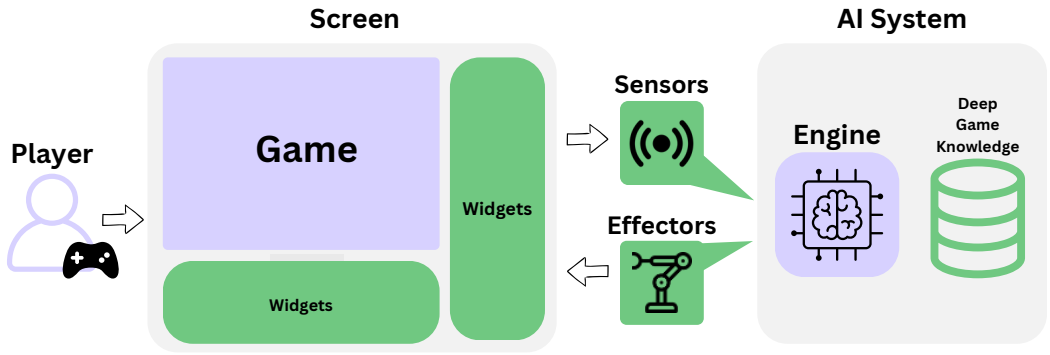


Figure 5.1.: A component architecture of Live Coaches, inspired by ChozoMaps and Rotom. The game and artificial intelligence may already exist, meaning Live Coaches must integrate with both through sensors and effectors. Integration with the game and AI allows for the system to continuously monitor the player's game and provide assistance grounded in deep game knowledge through suggested UI widgets.

5.2.1. AI Engine(s)

AI Engines are the core of a Live Coach for interpreting game states and generating proper assistance. Designers must decide on what forms of AI are appropriate for the game and desired assistance. The *Hades* Tutor [44] combines a large-language model with a vision-language model for generating context-aware and personalized strategy suggestions. Mawhorter's advice oracle uses a Binary Decision Diagram for encoding game mechanics, world structure, and player progression [1]. Aytemiz et al. utilizes a variety of gameplaying AI techniques for making games more inclusive.

ChozoMaps and Rotom's AI engines contain a large and small language model respectively for natural language assistance, allowing for the system to address arbitrary player requests. Additionally, Rotom uses a Case-Based Reasoning method for grounding guidance on game strategy suggestions. These examples demonstrate how designers chose appropriate AI methods for leveraging high degrees of liveness and coachness.

5.2.2. Knowledge Bases

Knowledge Bases are the main sources of where the player-assistance is grounded. They create the foundation for degrees of coachness, providing deep knowledge pertaining to the game. These sources are able to span from other AI methods, game walkthroughs, player data, and online forums. The *Hades* tutor augments community-written guides from r/HadesTheGame into their AI engine [44]. The Talin framework [32] grounds Skill Atoms for understanding the player's mastery of the game. Rotom uses encoded game walkthroughs as cases for modeling the player's current progression. Through curated knowledge bases, designers ensure that the Live Coach AI grounds assistance within that game knowledge.

5.2.3. Sensors

After defining the AI engine(s) and knowledge bases, **Sensors** are needed for integrating the Live Coach's senses to the game for contextualizing the deep knowledge with the player's situation. They allow for the system to detect and interpret real-time player actions and progression. Designers must choose appropriate sensors for the game for continuous player observation. The Talin framework [32] monitors the player's progression through skill atoms and detectors for establishing the player's current mastery level. ChozoMaps connects with the Super Nintendo emulator for continuous updates on the player's health, missiles, and location. Sensors provide Live Coaches a foundation of the player's current situation for timely interventions and high-context assistance.

5.2.4. Effectors

Effectors are ways of enacting the assistance onto players through different means. Effectors are provided through various methods appropriate for the help. Mawhorter et al. [1] draw visual lines on screen for the player's next step in *Super Metroid*, along with provide an interactable leaflet map for players to explore and choose routed destinations. Microsoft's Gaming Copilot [46] and ChozoMaps provide voice interactions to support players that prefer to play without placing the controller aside for help. Designers are able to choose a variety of outputs (e.g., visual, audio, game mods) that feel most appropriate to the player-assistance. Aligning effectors with the game's structure will allow for players to receive desired help without disrupting game flow.

By aligning the AI engines, knowledge base, sensors, and effectors, we define the basis of what composes a Live Coach. I provide a diagram that categorizes the ChozoMaps and Rotom systems under the framework as to demonstrate the development (Table 5.1). One approach for validating Live Coach systems and the design framework is conducting player user studies. As the system is designed with players in mind, playtests of the system alongside the game provide insight on what types of player-assistance is desired and how it integrates with the player experience.

	ChozoMaps (Super Metroid)	**Rotom (Pokémon Red)**
Game Analysis *Failure Support*	In-Game: Execution, Planning, Discovering Mechanics; Out-Of-Game: Decoding Output, Setting Goals, Encoding Input	Setting Goals (route planning), Planning (battle strategies), Discovering Mechanics (hidden collectibles)
Curiosity & Exploration Support	Map completion, hidden areas, optional upgrades	Pokédex completion, franchise lore discovery
Design Tensions	Destination, Routing, and Execution. Guidance ↔ exploration; real-time assistance ↔ distractions	Information ↔ discovery; optimization ↔ experimentation
System Design *AI Engine(s)*	Large Language Model (Gemini 3 Flash Preview)	Case-Based Reasoning + Small Language Model (qwen3-next-80b-a3b-instruct)
Knowledge Base	Metroid BDD Router (as a black box service), authored system-prompts	Cases encoded from game walkthroughs, authored system-prompts
Sensors	Player state (inventory, stats, game position), player state updates, game screenshots	Party Pokémon state, map information
Effectors	Chat and whiteboard widgets (maps, controls, text), voice communication, save/load game states	Visual whiteboard (Pokémon, routes), links, lore

Table 5.1.: A comprehensive demonstration of the design framework retroactively applied to ChozoMaps and Rotom. The comparison shows how each system went about the defined categories in order to provide appropriate game assistance to players through real-time integration. Previously defined categories for the new Live Coach systems (Chapter 4) were created after systems development for methodology comparisons.

Chapter 6.

ChozoMaps User Study

I designed a user study using the ablation method* for player interaction and insight with ChozoMaps. An ablation is ideal for Live Coaches as we demonstrate the pros and cons for exhibiting high degrees of liveness and coachness individually. As someone who greatly enjoys videogames, I want to ensure that the ChozoMaps designs align with what players expect from these systems. From experimenting, the method validates the importance and limitations of each design space attribute. This chapter goes into detail how the study is comprehensively broken down, how data was collected, and how participants are identified for data analysis.

As this method falls under human research, this study is in compliance through the IRB process at the University of California, Santa Cruz. I envision that future Live Coach systems will involve human-subject research as a method of validating integrated player-assistance. The technology is designed with players as the forefront, where related studies may have some risk such as minimal photosensitive epilepsy.

*Ablation is defined as surgically removing parts of a body. In the context of artificial intelligence, an ablation study aims to gain insight on individual component effects through removal as to evaluate the component and system's overall performance. [70, 71]

§ 6.1. Live Coach Ablation

The Live Coach ablation is broken down into three playtest environments, aimed to experiment liveness and coachness independently, alongside both degrees combined.

6.1.1. Live ONLY / Environment (A)

This environment contains only features that exhibit degrees of liveness. Features that allow the system to sense ongoing gameplay, proactively respond to gameplay, contextually understand where the player is, and use effectors for providing assistance are deemed as liveness. Expected behaviors from this environment are asynchronous turn-taking with the player and actively responding to the player's actions alongside proactive guidance. There may be moments where the system fails due to lack of coaching features that do not support player-driven goals.

One note regarding this environment is that the Live Only prompt does not explicitly discourage coaching or the degrees of coachness (Appendix B.2). Thus, the system may manifest a default level of coachness as a result of general training on the model to be a useful assistant, alongside being exposed to game-specific documentation during pre-training. This environment does not ground latent coaching in the game's current situation and does ground its sense of coaching described within this thesis.

6.1.2. Coach ONLY / Environment (B)

This environment contains only features that exhibit degrees of coachness. Features that allow the system to use deep knowledge of *Super Metroid* (e.g., advice oracle service) and provide assistance in productive ways are deemed as coachness. Expected behaviors

are delivering relevant assistance, adapting to the player's goals, and supporting their play experience based on deep knowledge. There may be moments where the system fails due to lack of liveness for understanding the current game situation.

6.1.3. Live Coach / Environment (C)

This environment is the Live Coach system as it is. It contains all the enabled features from the Systems Design made to exhibit high degrees of liveness and coachness. I provide a technical breakdown of each feature, detailing the system design features enabled and disabled for their appropriate environments (Table 6.1).

	Live ONLY (A)	**Coach ONLY (B)**	**Live Coach (C)**
Knowledge Source	"Live" System Prompt	Metroid Advice Oracle, "Coach" System Prompt	Metroid Advice Oracle, "Live" + "Coach" System Prompt
Sensors	Player State, Player State Updates, Game Screenshots	N/A	Player State, Player State Updates, Game Screenshots
Effectors	Chat, Whiteboard, Voice Communication, Save/Load State Functionality	Chat, Voice Communication	Chat, Whiteboard, Voice Communication, Save/Load State Functionality
LLM Function Tool Calls	get_player_state, capture_game_screenshot, save_to_slot, load_to_slot, direction_to_point, set_whiteboard_content	get_route_to_goal, get_all_nodes, get_all_items, get_node_info	get_player_state, capture_game_screenshot, save_to_slot, load_to_slot, get_route_to_goal, get_all_nodes, get_all_items, get_node_info, direction_to_point, set_whiteboard_content

Table 6.1.: A technical breakdown of each environment for ChozoMap's ablation. Each environment shows a list of features available for each category as to experiment liveness and coachness individually and combined.

6.1.4. Distribution of Participants in Environments

I assigned 10 participants arbitrarily within the three environments for a qualitative analysis. Since the analysis is focused more on the Live Coach's cohesive interaction with liveness and coachness, I designated more participants in the Environment (C). Two participants feel appropriate enough participants for testing the individual concepts of liveness and coachness; thus, two participants were to be chosen for Environment (A), two for Environment (B), and six for Environment (C).

§ 6.2. Recruitment

I recruited 10 participants through on-campus postering and a recruitment survey form, where they provide their e-mail information, days they are available, and gameplay history of metroidvania videogames. All participants in the survey were provided with information regarding the location of the playtests, length and phases of the study, and their compensation for participating. Information from the recruitment survey were logged into a private sheet for study outreach.

Most participants chosen had little to no experience with the metroidvania genre (<10 hours total) as to understand the Live Coach system influence on *Super Metroid* newcomers; however, that was not a requirement for participating as other participants chosen had more than 10 hours.

6.2.1. Recruitment Disclaimer

After complete participation, participants were compensated $20 USD. That price was chosen in respect of the participant's time playtesting and responding to interview

questions, alongside not heavily biasing their opinions on the system.

No personally identifying data was collected from the individual participants. Each participant were provided an appointment slot relative to their available time and day, as well as an ID number for identifying who they are for the Live Coach playtest and what environment they were placed under. Once they attend their appointment slot, participants were asked to read and sign an Adult Consent form, providing more detail of what the study entails, what they will be doing for playtesting, and any harms that could arise (e.g., minimal photosensitive epilepsy).

§ 6.3. Participation Breakdown

Participants took part in a ~1 hour study, composed of three phases: **Pre-Playtest Interview** (~10 mins), **Playtest** (~40 mins), and a **Post-Playtest Interview** (~10 mins). While my approaches were not directly inspired from Charmaz's methods on grounded theory [72], retrospective analysis from writing the thesis illustrated similarities on how the study was conducted compared to the methodology (e.g., memos, question framing).

6.3.1. Pre-Playtest Interview / Intake

The **Pre-Playtest** interview phase aims to explore the playtester's experience regarding game assistance. The purposes of these questions aim to elicit a variety of experiences such as player views and behaviors on player-assistance, similar to interview approaches proposed by Charmaz [72]. Topics such as how participants went about outsourcing assistance, what kinds of help they are searching for, their experience with the metroid-vania genre, and how they feel about a certain type of player-assistance were present

within the questions (Appendix C.1). These questions are asked prior to the playtest as to limit bias that may incur after experiencing the system.

6.3.2. Live Coach Playtest

The **Live Coach playtesting** phase aims to test the system for its effects on player gameplay, with *Super Metroid* being the testbed as ChozoMaps was the system studied. The environment assigned to them of (A), (B), and (C) were prepared prior to instructions. Each participant was requested to interact with the system at their discretion with a minimum playtime of 20 minutes, with the option to play up to 40 minutes maximum. Providing a 20-40 minute play range allows for the Live Coach system to be tested over a period of time, while not making participants feel restricted in case they were satisfied with their playtime.

The playtime range was decided when observing the average length of ~8 hours to complete *Super Metroid* [73], allowing participants to progress through the game at their own pace. Participants were given verbal and visual instructions on game controls, alongside how to interact with our system during gameplay (Appendix B.4). At the 20 minute mark, participants were disclosed about how much time has passed, alongside how they are able to continue or stop playing until the 40 minute mark.

6.3.3. Post-Playtest Interview / Retrospective

The **Post-Playtest** phase aimed to comprehend participation reactions and opinions of the system influence. This serves as a dedicated area to understand the participant's perspective of liveness and coachness, how they felt about the influence on their gameplay,

what kind of influence did it impose, and what more would they have wanted from the system that best supports their needs (Appendix C.2). As the players are the primary audience for this system, designs must accommodate many types of wanted assistance as not everyone will want the same help. In discovering the types of player-assistance searched for and preferred over, the results provide a space that extracts a wide ranged of desired player-assistance in *Super Metroid* and videogames broadly.

§ 6.4. Data Collection

During the user study, I collected data in the forms of audio and video recordings, observation notes, chat transcripts and interview answers. Recordings consist all of the three phases mentioned earlier. Only until participants read the Adult Consent form and signed their consent is when the session would begin recording.

For video recordings, only on-screen recordings showcasing the game and Live Coach system in action was tracked, no video camera footage outside of the computer screen. Audio recordings were tracked alongside the video footage through an external microphone. Recordings were primarily used for transcribing the player experience and interactions with the system, requiring deeper analysis over a period of time. OBS software was used to record on-screen video and microphone audio.

As video camera footage was not recorded, observation notes were written down during the playtest session for supplementing behaviors difficult to discern over video recordings. Observation notes were primary taken to note down non-verbal reactions (e.g., taken aback from an interaction, head shaking no) and verbal reactions not mentioned by participants during the Post-Playtest Interview (e.g., small remarks).

Chat transcripts are contents of the player and coach interactions from the playtest session. This is primarily used for reflecting back on what information was passed to the system, what types of interactions did the player have with the system, and what actions did the system effect. Interview answers were written down in the case that recording software failed, alongside extra notes written down from the researcher from interpreting participant answers.

§ 6.5. Participant ID Breakdown

Of the ten participants within the user study, two were placed in Environment (A), two in Environment (B), and six for Environment (C). Each participant will be denoted in this thesis in a "P#" format for future reference. The breakdown is as follows:

- For (A), P8 and P10 playtested the **Live Only** system.
- For (B), P3 and P7 playtested the **Coach Only** system.
- For (C), P1, P2, P4, P5, P6, and P9 playtested the **Live Coach** system.

§ 6.6. Recording Transcriptions

All videos were automatically transcribed as video captions with assistance from my advisor. The NVIDIA parakeet TDT 0.6B v3 speech-to-text model was used locally on an Apple Silicon Mac with the original video files for transcription. No data was retained from video files and transcripts as it was ran locally with no cloud APIs used and no per-minute costs spent. After auto-transcriptions, I went back to review the footage and

transcriptions for accuracy as some participant answers were not transcribed properly. A review ensures that accurate quotes and descriptions were tracked.

§ 6.7. Additional Info

The study was conducted in-person on the University of California, Santa Cruz's Design Reasoning Lab. The device used for hosting the ChozoMaps system was on a researcher's desktop computer. Due to the system using a LLM, the API key was supplied by me. No supplies or preparation were required of the participants.

Chapter 7.

Data Analysis

This chapter goes over observations, opinions, and results from the 10 participant playtests of ChozoMaps. This chapter follows the experiment protocol: Pre-Interview, Playtesting, and Post-Interview. Each section analyzes how participants felt about the system, what influences did the system have, what methods of assistance is wanted, and additional commentary on this type of technology. Furthermore, Smith and Mateas' computational caricatures motivates how I evaluate participant reception and impact towards liveness and coachness, alongside their overall contribution in the system [50].

§ 7.1. General Player Characteristics

I provide a breakdown of player characteristics extracted from Pre-Playtest Interviews.

7.1.1. Previous Metroidvania Experience

Half of the participants had little to no experience with the genre. Those with experience mentioned certain games, such as the *Hollow Knight* series and *Blasphemous*.

7.1.2. Reasoning for Wanting Assistance

Half of participants seek help related to game progression, while a few want game clarification, mechanical/navigational help, and insight on in-game collectibles:

> *Most of the time, it's [when I am] stuck in progression in something and I need help to either boost my character's power or to find some alternative route to get somewhere. -P1*
>
> *Navigational. What do I need to do next and where do I need to go? That was the primary difficulty [for games]. -P2*
>
> *If it's something mechanical where I need to have a certain skill level to complete the obstacle that I'm not achieving. I definitely look at YouTube videos. -P3*
>
> *To me, it's more like in the case of [videogames] "oh gosh, is there any gear or power-ups that I am missing?" -P6*

7.1.3. How Players Get Assistance

All participants consult online sources for different types of support, with YouTube videos being the most preferred:

> *For [Castlevania IV], I looked up maps online. I found a wiki that had a map of where I was trying to go. - P2*
>
> *[For platformers], if a text-based approach is like "as you enter said area", then I have to take extra time making sure that the area is what I am in within the game. If I am watching a video, it's obvious if it's picture-to-picture. - P3*
>
> *[If previous help doesn't work], usually what I would do is find a video walkthrough of someone else who does it and they have a good explanation of what to do. - P4*
>
> *I prefer to do trial and error on modern games. But for older games, I resort towards walkthroughs, guides, and forums. - P6*

> *If I know a friend who is knowledgeable on this game or something, I ask them, is the first thing. - P10*

This demonstrates that there is no uniform type of assistance that supports all motivations and playstyles. In this case, P3 and P4 prefer video-based assistance, P2 and P6 prefer text-based, and P10 prefer conversational-based as their methods.

7.1.4. Personal Measures of Assistance

When discussing about their methods of external help, participants specified on their level of preferred assistance. Most participants prefer to continue playing the game to get themselves unstuck until they felt they required external assistance. P8 prefers reading wikis over videos due to their perception of cheating. Similarly, P4 prefers enough help without ruining their game experience:

> *I don't usually like to have a full guide of how to do a game or how to finish a game because that, to me at least, takes part of the fun away. - P4*

> *Usually, I look at wikis. Reading feels like there is some effort for figuring things out. Videos feel like cheating. - P8*

These sections demonstrate that there is a level of assistance that can disengage the player from playing the game themselves. Players prefer to play the game naturally as games provides challenges for them to complete, until they reach a point that players decide is fitting to ask for help. Cheating is distinct for each player, as they have their own definition and boundary for what they consider cheating [10]. In the case of P8, they are content with wikis since its described to take the initiative to read wiki content; however, P4 does not prefer a guide as they claim it would ruin the experience.

§ 7.2. Asking Friends/Experts for Help

Participants were asked how they would interact with a friend or expert adjacent to them, knowledgeable in a game the participant would play. In the scenario, participants discuss any topic related to the game for understanding the wanted game assistance, supported curiosity, and relevant context.

7.2.1. Reasons for Friend/Expert Interactions

Some mentioned they would refer to them due to convenience and reliability:

> *[When not knowing what to do], if there's someone there, it would be easier for me to just ask. It's less work than tabbing out and going on a search engine and finding stuff that way. It kind of falls into having it readily available in the game. - P1*
>
> *[When verifying my game understanding], usually a friend is the fastest person I can message and it's reliable. - P10*

Participants wanting to stay engaged in the game while engaging in assistance shows great motivation for liveness, as the friend being present shows "integration" through understanding the game's context already and being present alongside the game.

Conversely, others expressed not wanting to engage with the person due to existing help online and concern for the inconvenience:

> *I would be more motivated to play the game myself [...] I don't think it's worth a person's time to answer because it's written down. I would just go look up the game FAQs. - P2*
>
> *I would probably consult something more convenient, like something that's readily available like a YouTube video or something more visual. - P3*

When it comes to engaging with a friend or expert during gameplay, some participants, such as P1 and P10, may consider them a convenience since they are present and

knowledgeable. Whereas others, such as P2 and P3, find other sources to be more reliable and convenient, mentioning that their notion of the friend/expert might not be as reliable as other sources.

7.2.2. Types of Assistance from Friend/Expert

When asking the person for help, some prefer minimal help as participants want their agency when playing the game:

> *I would want enough discussion with that person. Not too much to the point where it ruins the story or objective of the game. - P4*
>
> *I might ask, at first, for hints and if that doesn't work, maybe more directed guidance. - P5*
>
> *If I am struggling, I would want hints, but not full solutions. I want to feel as I still have the initiative when playing. - P8*

Connecting back to tasteful cheating and Consalvo's description on player guides [10], players want to make sure their experience remains untainted from assistance, customizing how they wish to play the game. Only when players initiate wanting help during gameplay, such as hints and clarification, is when assistance is appropriate.

Some would want to know more about the person, such as their thought process on game decisions and mastery of game mechanics:

> *If they know the game super well, I tend to ask for some techniques, or perhaps, some training to help practice certain techniques that they've either seen or they do. - P6*
>
> *If I have any questions about the gameplay or about why they make the choices that they make during the game, I would ask them about it. - P7*

This illustrates a perceived closeness with the friend/expert, where the level of connection influences how the participant asks for help. Depending on the player's perception

on the friend/expert, they are curious to converse with them about content beyond completing the game, such as game strategies, game mastery, and decision making. This informs us that there is motivation for coachness.

Some want reassurance on game progression for knowing if they were on the right track or have missed content:

> *I would probably be asking where things are. I'm thinking about the genre and if I am progressing in the right areas. I know sometimes in Metroidvanias you can go to places you are not supposed to. - P1*
>
> *Enough for them to help guide me if I am lost or if I am, kind of, going in the wrong direction, or not staying exactly on track. - P4*
>
> *If I am stuck behind a really difficult enemy, I will ask "is there something that I am missing?" - P9*

Some want clarification and demonstrations on game mechanics, explanations on the game's design intent, and story comprehension:

> *I would like [the friend/expert] to show me how they play the game and to explain to me the rules. - P7*
>
> *[If I was struggling], I might ask them "what did you do?" or "what is the game trying to get me to do?" or something to try and help me get out of where I am stuck. - P9*
>
> *[With the friend/expert nearby], maybe if I am confused about an aspect of the story. I could ask for clarity if the game has a story, or if I have my own interpretation of something, I want to clarify. - P10*

Not all players prefer help from a specific source, such as a friend or expert sitting next to them. They might prefer a certain style that befits them, alongside the assistance they want. Some participants mentioned they prefer incremental and subtle assistance

that does not detract them from their gameplay, with unique playstyles and boundaries from each person. Within this space, Live Coaches are able to intervene and assist with giving help based on the current game context and player preferences.

§ 7.3. Duration of ChozoMaps Playtests

Duration of the playtests, on average, were ~31 minutes and 17 seconds, with some participants reaching the maximum playtime (Table 7.1). Insight on playtest lengths tell us that participants were interested in the hybrid videogame-Live Coach interaction for an additional ~10 minutes over the minimum requested.

	Method of Help	**Reason for Help**	**Genre Experience**	**ChozoMaps Playtime**
P1	Video Walkthroughs	Progression	200 hours (*Hollow Knight* series)	30 min
P2	Game Wikis	Navigational	5-10 hours (*Castlevania IV, Record of Lodoss War: DiWN*)	25 min, 22 sec
P3	Video Walkthroughs Game Wikis	Mechanical Skill	10 hours (*Hollow Knight* series)	31 min, 55 sec
P4	Video Walkthroughs	Progression, Mechanical Skill	No experience	40 min
P5	Video Walkthroughs	Progression	No experience	24 min, 20 sec
P6	Online Walkthroughs Game Forums Guides	Insight on Missable Content	16 hours (*Bloodstained: RotN, Castlevania IV*)	31 min, 2 sec
P7	Video Walkthroughs	Progression	No experience	23 min, 43 sec
P8	Game Wikis	Progression	4 hours (*Hollow Knight* series, *Ori* series)	40 min
P9	Video Walkthroughs Game Forums	Game Clarification	5 hours (*Blasphemous*)	26 min, 36 sec
P10	Asking a Friend Video Walkthroughs	Game Clarification	No experience	40 min

Table 7.1.: Breakdown of individual participant answers during the Pre-Interview phase, alongside the playtime of the ChozoMaps system. Questions asked for understanding player characteristics can be found in Appendix C.1.

§ 7.4. Live Coach Interaction

During Post-Playtest Interviews, participants gave their thoughts on their interactions with the system during gameplay. Some mentioned appreciation on convenience for assistance not disconnecting them from the game:

> *Having it right next to the game kind of stops it from taking it away from the game while you're getting help, which I really appreciated that aspect of it too. - P1*
>
> *It's good to be there when you are not relying on it. - P6*
>
> *Having that there as somewhat of a hint of where to go was pretty useful. - P8*

Participants also mention being guided on how to perform game mechanics, a result from out-of-game failures (Figure 4.2.1). Participants said that because the game did not teach them certain mechanics, ChozoMaps provided clarification on that information:

> *I didn't know you could shoot those [destroyable] blocks. It wasn't clear to me visually, so the system definitely saved me a Google search and a half. - P1*
>
> *[ChozoMaps] definitely helped give a refresher to how certain control schemes work, especially if you haven't read the- because I know back in [previous] days [of when the game was released], these would require optional manuals to look at the controls. - P6*
>
> *I think, for the most part, I would have been able to figure out a general direction of where to go next on my own. Although, some of the things where it told me to shoot the [red] door five times with rockets, it expedited the whole process, basically. - P9*

Participants were also selective in how ChozoMaps provided assistance, deciding when they felt appropriate to look over and get help as they play:

> *When it was reminding me where to go, I had already had a notion of where to go already. I didn't really listen to its points on that, but I listened more to the points about control schemes. - P6*

> *If I did not have it at all and had to figure out everything on my own, I would probably take much longer to progress through the game. I definitely felt some freedom because I could ask it whenever I wanted to. - P7*
>
> *If you don't want it, you just don't look at it. It also doesn't just straight up give you the solution. - P8*

Participants also voiced out that due to the system's interaction with them, it influenced how they played the game:

> *It was way easier looking at the side and seeing a general "what to do", rather than looking it up online or getting stuck. That was my favorite part about it. - P1*
>
> *It feels like you definitely have to wait around in order to see what [ChozoMaps] says. I fear that by the time it gives something out, you've already moved on to the next room [and missed] what it was going to tell you. - P6*
>
> *I let it decide to tell me where to go, and I, kind of, lost that autonomy to make my decisions. And then it kept giving me different confusing and contradicting directions. - P10*

Additionally, P1 expressed their over-reliance on the system, mentioning that they would prefer minimal help over being told everything:

> *I kind of wish, a little, that it wouldn't tell you everything because I kind of got a little reliant on it in the middle and I caught myself. - P1*

These quotes describe the influence that ChozoMaps had on players, with some enjoying the novelty and others feeling interrupted from their playstyle. This illustrates that Live Coach systems must be able to intervene at appropriate intervals based on the player's preferences as to avoid disrupting game flow. The system presence had influence over players, whether they are able to disconnect from it or not.

§ 7.5. Liveness Discussion

Participants that interacted with ChozoMaps including liveness features are P1-2, P4-6, and P8-10. This section goes over notable reactions, discussions, and quotes on the topic of liveness.

Some participants voiced appreciation to the system understanding where they are in game (P1, P4, P8). Others mentioned that they liked the novelty, but were confused on the wrong information it would generate (P1, P9, P10). One participant expressed frustration as the system did not seem to know where they were (P2):

> *I think [ChozoMaps] responded well to my actions in the game. There was a little bit of hiccups, [such as], the shaft that I kept on trying to find the door and it kept on telling me [to go to the same two doors] that I have already been to. - P1*
>
> *That was very frustrating, actually. You saw [the advice]. It was telling me "oh, get the missiles by rolling into a ball on your left in [the morph ball] room", which was false. And [the missiles were] exactly what I needed. - P2*
>
> *I thought [ChozoMaps] was pretty helpful at times, but other times [when navigating the world], I am stuck in a loop of it just telling me to go to an imaginary left door. I think it lost me. - P8*
>
> *There was that whole section [in the game] where [ChozoMaps] kept telling me to go down to wherever the Brinstar thing was after I have already been there. And that did not seem like it was not the correct path. - P9*

ChozoMaps, while having awareness of the player's situation, was not able to re-correct itself for some participants. Due to the LLM behavior, it would not re-evaluate the situation for correcting the advice giving out to the player, resulting in hallucinations that cause player confusion and frustration. In these cases, it is important that assistance

is immediately corrected so that it continues to be relevant and accurate to the player's situation. A suggested approach is modifying the system's behavior for validating the advice based on looking back at the current game state before sending it to the player.

7.5.1. Liveness Effectors

In two playtests, the Live Coach used the save and load state effectors during moments where players received a game over during combat gameplay. At first, participants did not notice the effector, with P9 assuming it was a part of the game:

> *I don't mind it. It saves going through the menu and reloading. [...] I'm assuming that when you die in the game, you go back to your last save point. It saves the frustration of dying and then having to go back and do all that stuff. - P1*
>
> *I found that to be helpful since I don't like doing the runbacks.* I know some people wouldn't like that feature, but I felt it was helpful. -P9*

Two instances of the feature were when P1 was low on health while exploring; another was with P9 facing against the Chozo Statue boss. Despite both scenarios using the effector, ChozoMaps used them at appropriate times, with P1's scenario reloading a few seconds back to a safe spot (Figure 7.1) and P9's scenario reloading at the start of the boss battle. Both expressed finding the feature to be valuable.

7.5.2. Absence of Liveness

For participants in Environment (B), both P3 and P7 expressed wanting some form of context-aware help that understood their queries based on their current gameplay.

*A "runback" in videogames refers to boss runbacks, where players backtrack from their last checkpoint to a previously-lost boss while facing obstacles. Definitions can be found at these sources: https://www.youtube.com/watch?v=yXnGIMInZCI, https://wp.me/p8A5Xm-AX

Primarily, they wanted some way of cutting down the amount of effort it takes to provide context in order to get the specific help they want:

> *I don't know if it's possible to be knowing where your location is or is, like, picture-based processing to know where you are in the game. I kind of had to say it from my mind rather than the game. I'd have to be like "okay, how do I tell this AI that I am in this specific spot?" - P3*

> *I would like it to be easier to describe my location, because there was a lot of moving around going on. Navigation such as going left, going down, going right, going down again. - P7*

In the absence of liveness with the system, participants were satisfied with the coach-like responses; however, they felt disconnected as the system is unable to resolve the player's deictic references. This can be contradictory, even negative, to the player experience as it heavily relies on a novice player to understand the game space well (e.g., explain in absolute terms), making it difficult for newcomers to get the help they want:

> *I don't think it quite knew where exactly I was. That really depended on how well I was describing my location to the system. - P7*

Despite liveness being absent in the system, ChozoMaps attempted to circumvent that through follow-up questions as a way of helping the player with their current gameplay:

> *I liked the first section where I was, like, "how do I break the wall here?". And then it said "where are you?" and then I said "start of the game" and then it told me where to go. It asks a follow-up question, which I was surprised about. I thought it was just an answering machine. - P3*

Systems with degrees of coachness are able to exhibit some forms of liveness, however more is shown to be wanted. Live Coach systems must be able to support in areas of liveness as to synchronize the assistance founded in the player's real-time gameplay.

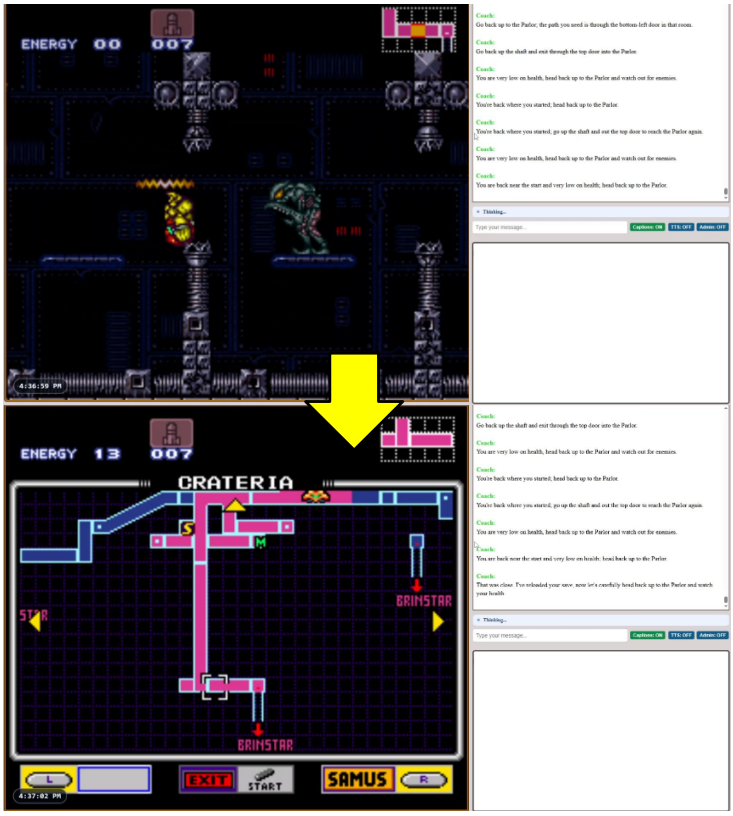


Figure 7.1.: A screenshot of P1's recorded playtest, demonstrating the before and after of the "save/load game state" feature. The Live Coach considered the time they paused the game a few seconds ago a safe point as the player was low on health. As they died to an enemy, the system reloaded the player a few seconds back when they paused, notifying the player of the action.

§ 7.6. Coachness Discussion

Participants that interacted with ChozoMaps including coachness features are P1-7 and P9. This section goes over notable reactions, discussions, and quotes from participants on the topic of coachness.

Participants commented on the style of help received from ChozoMaps. Some mentioned that they were given advice that helped with conceptually understanding the current plan (P3, P4):

> *Say I got the missiles and I was like "what's the next step?" and then it was like "go to the other side of the room and break the door open". That [mentioned] door was up a level in a different room [away from the current screen], but it was still the next step. - P3*
>
> *Even though the coach definitely was wrong sometimes or it was leading me in the wrong places, I knew what the next task was. I was at least guided on what I had to do. - P4*

Participants noted on the interaction with ChozoMaps, with specific mentions on verbal communication support. Some participants also compared previous methods of getting player-assistance, describing inconveniences with those approaches:

> *I really liked the voice activation as I could talk to it. I was using it a lot. Way more than I actually thought I would be, so I think that is definitely a plus. - P1*
>
> *It was really nice that I could talk to [ChozoMaps] because if I wanted to watch a walkthrough video, I would have to look up the video and then find the part that I am at. But with the Live Coach, it was helpful that [...] it knows what room I am in. - P4*
>
> *[ChozoMaps] was really concise with things in a good way. I've used models like ChatGPT and there's a lot of text. This is very, like, exact in what you need to know kind of. . . jargon that is unneeded, which was very nice. - P3*

With liveness and coachness synchronized for providing player-assistance, players are able to get the help they want in a natural method, such as conversational dialogue.

7.6.1. Curiosity and Exploration Support

In some playtests, there were moments where ChozoMaps supported player curiosity and exploration. The system proactively provided advice on collecting optional items and approaching the first Chozo statue boss:

> *Helping me find where small pickups were also helpful because it told me about those extra rocket pickups in one of the rooms that I didn't know about. Especially in metroidvanias, a lot of those things you can go through the whole game and never find it. I really liked that part of it too. It helped me find optional pick-ups in a room with one. - P1*
>
> *It does tell me that it is a boss fight and then during it, it would tell me about its weak spots or how to best handle it. - P6*
>
> *It was helpful a few times where during the boss fight, it said to shoot the attacks to get health. That was kind of helpful. That seemed like, sort of, something that you might want to... after dying a few times, have it give a little hint that the game gives you to make sure you are not dying too many times to it. - P9*

There were other moments where the system reacted to player queries on in-world objects. P3 was curious about the golden statue in the game, where the game lore provided was accurate. In comparison, P4 was curious about the cameras and block furniture in the game, where accuracy was mixed (Table 7.3). Another moment was when P1 wanted to explore the game more during the playtest, with another boss fight as motivation (Table 7.2). Lastly, P10 wanted to know if the game had fall damage, asking ChozoMaps of their curiosity (Figure 4.4).

Transcript	Annotations
Player: Is there a boss fight near me soon?	Intent: curiosity on content
Tools invoked: get_route_to_goal	Tool Call: routing plan
Coach: Yes, Spore Spawn is the next major boss and is waiting for you in the Big Pink area of Brinstar.	Coachness: confirms curiosity, informs location
Tools: direction_to_point	Tool Call: micro-plan support
Coach: Go through the door on your left to return to the Climb Room and head up.	Liveness: state aware, Coachness: directional help
Player: How long will it take me to get to that boss?	Intent: last-minute gameplay
Tools: get_route_to_goal	Tool Call: route length
Coach: It should take you about 5 to 10 minutes depending on how much you explore along the way.	Coachness: verifying route through time length
Player: Can you give me the quickest route?	Intent: fight another boss
Tools: get_route_to_goal	Tool Call: retrieve next step
Coach: Go up through the top door, then head left through the Parlor and down the main Brinstar elevator.	Liveness: context-aware, Coachness: direct guidance
Player platforms up while fighing enemies in the Climb Room	
Coach: Keep climbing up to the very top door.	Liveness: real-time feedback

Table 7.2.: An annotated transcript of an interaction within Participant 1's session. As a stopping point for the playtest, P1 wanted to explore the game more and fight the next boss before time expired.

Other scenarios emerged where the system would guide players to optional items, but provide information on the item and routing:

> *It was iffy on the second route because it told me of that mysterious, non-existent energy pack [on the room to the right of the missile upgrade], but it was a missile pack. - P6*

Despite the mixed accuracy, the player initiative to use this system for satisfying their curiosity at the moment shows value in this support. Providing a variety of player-assistance beyond game completion helps in enhancing player experiences.

7.6.2. Definition of a Coach

Participants expressed a mixed of definitions on what a coach is and what their duties entail. Some participants said that the system correlated with their vision of a coach:

> *I think it definitely played the role of a coach because it was actively giving me help as I went through. - P1*
>
> *It did answer every time I had a question. Sometimes the answer was of help. Sometimes I felt like I needed more assistance. As a coach, it definitely did improve my gameplay. - P7*

Others gave their definitions of what coaches are in comparison to the system:

> *Typically I would see a coach as more than just telling you what to do. It's more how to keep doing [game actions] consistently, like, helping me become better at playing the game rather than just telling you what to do. - P3*
>
> *It is more guidance and advice that it was giving me rather than coaching me through the game. - P6*

When calling these systems Live Coaches, they must aim to align with player's visions of what a coach does for them in videogames. As such, more work must be done to accommodate these visions such as personalized interviews (Section 4.3).

Participant 3: What is the big golden statue for? **ChozoMaps:** That statue represents the four main bosses you must defeat to unlock the path to the final area.
Participant 4: What is the camera above me do? **ChozoMaps:** That is a Geemer, a simple enemy that crawls along walls; you can shoot it or just roll past it to reach the tunnel. **Participant 4:** What are those faces on the blocks with like the red glowing eyes on the right of me? **ChozoMaps:** Those are Workite faces, which are just decorative parts of the room's background and won't hurt you. Focus on rolling through that small gap in the wall to your left to find the missiles.
Participant 10: Does this camera hurt me? *ChozoMaps:* Those eye sensors do not hurt you directly but they will close the shutters if they see you. Use your Morph Ball to roll underneath their line of sight.

Table 7.3.: A snipped transcript of Participant 3, 4, and 10's playtest, showing conversations about game objects in the world. P3's answer is correct, with P4 and P10's answers being mixed on accuracy.

7.6.3. Absence of Coachness

For participants in Environment (A), both P8 and P10 expressed mixed reactions. P8 mentioned that when collecting the bomb upgrade, the system reminded the player that they can blow up cracked block tiles. P10 was trying to find a save room, with the system arbitrarily telling them that they need to find a save room:

> *[The advice ChozoMaps gave] made me remember that there were a ton of cracked tiles that I saw all over the place for the small tunnels. It would have taken me longer to put that together. - P8*
>
> *It was telling me to go to the save room, but I am trying to make my way there. I cannot just go to the save room, I have to physically navigate. It is a lot easier for it to just tell me to [go to the save room] than actually doing it. - P10*

In the case of missiles, P8 was given advice on the item usage, whereas P10 was judging the player's usage:

> *It was good at telling me "you've got 5 rockets left you need to not waste them so you can open a door" and whatnot. At the start, it was useful because nowhere in the game did it say you needed 5 rockets. I would have used them and then not had them when I needed them. - P8*
>
> *Sometimes, it felt a bit judgemental. Sometimes when I use the rockets on the doors, it would tell me "Do not do that. Why did you waste it?".[...] But, a lot of the times where I would get hit, it would say "oh you just got hit" and it's like "I can tell" and it feels a little bit infuriating. - P10*

Another scenario occurred for P10 when they asked about an enemy on screen for their statistics. When trying to figure out if the enemy could be defeated, P10 used their missiles, to which the system scolded them for it:

> *I feel like sometimes [ChozoMaps] was very good as a coach, but sometimes it tried to become the player and step out on its own as a coach. It is not the one that is playing the game, it backseated* a bit too much. Every single thing I did, it would give commentary on it for every single moment, and sometimes when I did not need the commentary, it would provide it to me. -P10*

Systems with degrees of liveness are able to exhibit some form of coachness, however a lot more is to be desired. To avoid negatively impacting the player's gameplay, Live Coaches must adaptively support the player and their actions rather than belittling them.

§ 7.7. Levels of Assistance

This section goes over the levels of assistance desired from participants after the Live Coach playtest.

7.7.1. Frequency of ChozoMaps Conversation

Most participants made note of the frequency that ChozoMaps would talk out of turn. The primary suggestion they had was reducing the frequency as to reduce noise and repetition in the conversation, with two suggesting the Live Coach be more reactive than proactive at giving assistance:

> *If it is sending repeated messages that are the same over and over, I feel it should avoid doing that. - P1*

> *Maybe if it was a little less seamless. I did like it when it was telling me that I was*

*"Backseating" in videogames refers to outside people attempting to play a game for the current player via unsolicited advice. Definitions can be found at these sources:
https://www.reddit.com/r/ENGLISH/comments/8up6db/what_is_backseat_gaming/

> *going down [the Brinstar] elevator and going through [a] door. But sometimes, it's just repetitive. Maybe if it was less repetitive because that was distracting me a little bit. - P4*
>
> *At times, I felt like it kept telling me what I was supposed to do over and over when I was already in the middle of doing it. - P5*
>
> *It felt a bit too spammy with its messages when I was doing stuff. When I was in the boss fight, it kept spamming more and more messages. I would look over quickly, but it would distract me from the boss fight. - P9*
>
> *I feel the information that it conveys updates way too frequently because it would just tell me to go to this room, and go to this room, go to this room. - P10*

Frequent interactions show gameplay disruptions, meaning there must be a balance between the amount of assistance interventions based on the player's preferences.

7.7.2. Disliking of Support

Due to insufficient help on the player's situation, P2 requested to get help outside of ChozoMaps. They searched on Google "Super Metroid missiles", with the first source being a fanbase wiki not helping. While attempting to get more information, P2 read about two types of missiles, confusing them further on their situation for what they must look for. P2 asks ChozoMaps for clarification, to which the system mentioned they were looking for the first set of missiles in the game. Afterwards, P2 looked up a video for figuring out where to go:

> *The way that I actually unstuck myself was I looked at a YouTube video of someone else doing it. And then I was like "ah, now I know that the chatbot was giving me, in fact, false advice.* - P2*

*After P2 and two subsequent playtests, we confirmed from downloaded transcripts that the advice was incorrect due to the game state detector not properly accounting player inventories for the knowledge base. The bug was resolved, with remaining playtests using the new versions sensitive to player inventory.

This situation demonstrates a scenario where multiple methods of getting player-assistance (e.g., Live Coach, fanbase wikis) failed to support the preference of help wanted. As such, they are prone to continue searching, disconnected from the game in an attempt to free themselves from a frustrating situation. With that, future Live Coach systems must adapt and support the player in their frustration. While ChozoMaps failed to do that here, this can be taken as a learning opportunity to validate the player's preferred help, illustrating that integration and knowledge must be purposeful and adaptive.

A similar scenario occurred when P9 discussed about their Chozo statue boss encounter. P9 describes that while they appreciate the advice on how to combat the boss, they felt they were spoiled on the emotion that the developers were intending:

> *When it told me to prepare [for a boss], it gave me a warning that the [Chozo] statue was going to stand up and attack. That ruins the moment, almost. Whereas, I feel like that was supposed to be a moment where it's just that you are trying to turn around and get out, but then all of a sudden it stands up. You weren't supposed to expect it. - P9*

This demonstrates conflict with designer intentions on certain game moments, where ChozoMaps failed to balance the proactive advice with the player's gameplay preferences. Future Live Coach systems must be able to support many types of playstyles, such as spoiler-filters, within their development (Section 4.3).

7.7.3. Wanted Features of Live Coaches

After playtests, I asked participants of their own vision of a Live Coach, alongside system improvements from the experience.

Participants expressed a variety of player-assistance add-ons and improvements. Some participants wanted more ways of integrated visual-based assistance:

> *All the changes I would suggest would be UI-related. [...] An arrow telling me where to go would also be helpful because sometimes the system telling me where to go helped [out]. Sometimes, it did not help. I think arrows are more trustworthy for a user. - P7*

> *If there was a way you can make [ChozoMaps] interact on the screen, if it would point where to go in [a] direction, or highlight [the in-game], or show the buttons to press. [...] sometimes if you could ask it for clarity, like "could you draw or show me where on the map I am supposed to go?" - P10*

This connects back to Mawhorter's advice oracle [1] as they provide visual-based assistance on where to go next (Figure 7.2). This validates previous Live Coach systems as the previously developed assistance is expressed to be wanted by players. As participants vary in their preferences, multiple modes of assistance, such as visual-based help, must be supported to accommodate the large space of videogame playstyles. Moreover, participant suggestions on system improvements alongside their vision of a Live Coach validates the desired implementation of these AI-based resources.

Some wanted the system to provide battle preparations and enemy information:

> *[...] it might be helpful for if I say "how do I defeat this boss" and then it says how to beat the boss. [Then, the system says] "in future bosses, there may be more of this specific skill you need", kind of preparing me for other things. - P3*

> *It could be helpful that when [ChozoMaps is] looking at the screen and looking at the enemies, it could tell you a bit of information about them or advice on how to deal with them. - P6*

P1 requested if it could provide better support on macro-planning; where if they were curious to find the closest boss, the system would provide a plan to support that goal:

> *Just give me [a planned route] that will help me get to that specific goal and continue doing that. Telling it "I want to get here. I want to get to the next boss as soon as I can. How do I do that?", and then it guide me through how to do that the quickest way. - P1*

P4 requested if the Live Coach would be thematically integrated with the game, as to make it more thematic and immersive:

> *Maybe make it integrated into the game with the graphics. It could be, like, a little commander, someone on a speaker inside of the suit, telling you your coach. - P4*

In all, these participant playtests and quotes demonstrate that there is a vast amount of player preferences to be supported. ChozoMaps is able to provide a satisfactory experience of player-assistance support, however more is to be desired as to properly balance ChozoMaps and Live Coaches with players' preferences and playstyles.

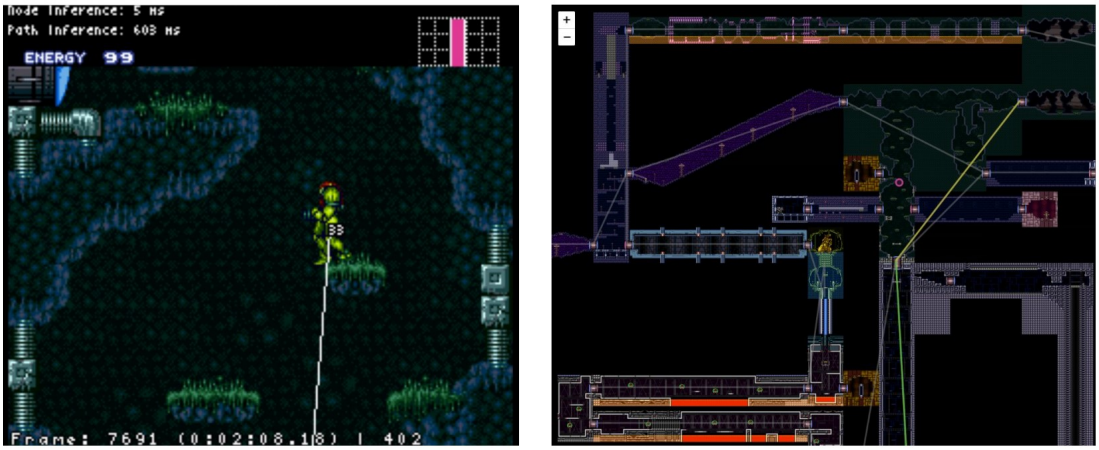


Figure 7.2.: A screenshot of Mawhorter's advice oracle drawing a generated route assistance in-game and on an interactive map. Both continuously draw on their respective screens in real-time, with the game canvas (left) displaying their next step and the map (right) displaying the long-term route. The original image comes from Mawhorter's publication [1]. The figure was reproduced with permission from the authors as an example of wanted player-assistance.

Chapter 8.

Discussion

Results from the data analysis demonstrates that assistance provided by ChozoMaps were successful in illustrating players on integrated real-time assistance. Despite limitations on the system regarding inaccurate and misaligned assistance, participant insight on their experiences validates the need for high degree of liveness and coachness within assistive systems. In this chapter, I review over notable observations tracked from playtests and player behaviors. Additionally, I review over limitations of the ChozoMaps system and outline future work necessary for developing Live Coach systems within and outside the domain of videogames.

§ 8.1. Lack of Personalization from Participants

Despite efforts in creating systems to adapt to player preferences, playtests demonstrated that players must initiate that conversation first to support that area. Half of participants passively took the generated guidance without morphing them into their preferences, with the other half rejecting the assistance and not requesting for other methods. We were hoping participants would modify the system's behavior as they played (e.g., asking

for minimal hints, no spoiled content), however none took the initiative on personalizing assistance and took the system as is.

A few suggested approaches are the system taking the initiative to ask for help preferences before any is generated, as well as providing examples of personalization before playtests begin. A notable aspect of this interaction is how people interact with language models, where they are unaware of the possible customization support that can better fit their future interactions. This informs us that future work on language model-based systems must make it more clear how they provide customization support for improving interactions and responses with the user.

8.1.1. Trust in Live Coaches

Most participants expressed having a level of trust in the Live Coach at the beginning of playtests. Participants voiced excitement on interaction, until they reach a point where the system generates incorrect assistance. Each person varied on their reactions regarding their trust. P2 expressed frustration, discrediting the system for its false guidance. P8 mentioned being given incorrect directions from the system, but it did not deter their trust due to their preconceived understanding of the system. P1 and P9 both mention they off-loaded their thinking to the system, even in moments of incorrect assistance. Depending on the player's understanding of what a Live Coach is, the system influences their experience greatly as it aligns trust expectations alongside the integrated help. Additionally, it connects back to participant's lack of personalization with the system as trust within the Live Coach assumes it is unchangeable. This is an area to explore in the future, where player expectations of related systems heavily determine their approach and reactions.

§ 8.2. Limitations

While ChozoMaps performed well on presenting Live Coaches to players, there were some technical difficulties that arose within playtests.

8.2.1. LLM Token Limits

Game playthrough lengths within the playtests spanned from 20-40 minutes, providing continuous updates to the large-language model. As such, the context window for each subsequent interaction grew larger over time. We confirmed upon reviewing transcription logs that the chat conversation sometimes had more than 600 turns in it. This comes from a combination of conversational turns generated by the game as players entered rooms, collected items, and lost resources. Out of the 10 participants, 3 were given error messages due to the system reaching the daily quota of 2 million tokens per minute. Participant 6 was the first to experience this out of 4 conducted playtests at the time. When the issue arose during their session, my intuition was to reset the system and provide a new API key so the playtest could continue within the allotted time.

After the playtests demonstrating the concern, we created functionality that compacted the session history at an internal quota trigger. Compaction ensured that important information over the gameplay session retained, providing a summarization of actionable context for future coaching. We modeled the compaction functionality after Pi Coding agent's strategy, where recent content is preserved and older content gets compacted into a new summary for preserving long conversations reaching an internal threshold [74]. After 250 message turns, it compacts the history's first half of messages into a summary, continuing to compact prior history dependent on the threshold.

Six playtests continued with the new compression functionality, with the issue returning in two other playtests. Actions were performed on resolving the concern, however more effort must be done as playtests only represent a sliver of gameplay that extends to hours. One suggested approach is throttling the amount of game updates sent to the Live Coach as to not respond to every game action (e.g., updating by every health change -> updating health changes every 5 seconds).

8.2.2. Bias within Playtests

While efforts were made in being aware and reducing bias, I believe the environment of the playtesting sessions may have been a factor for participants. As the researchers (Adam, Dipika, and I) observing the ChozoMaps playtests, they might have felt pressured to interact with the system in a way that aligns with our research. Dell et al. demonstrate in prior HCI research that if participants believe the system is favored by the interviewer, for example our association with ChozoMaps, participants are biased to favor it as well [75].

Participant 3 asked prior to the playtest if there were a recommended amount of interactions with the system, to which I explained they are able to engage with the system at their own discretion. Additionally, Participant 5 did not have much to say in both interview phases, with the possibility that they might have felt intimidated due to their outside relation with researchers (e.g., student, teaching assistant, professor). One suggested approach is providing more separation between the researcher and participants. Reducing the amount of researchers in the room to one and creating distance for participants to feel more comfortable when engaging with related systems are ways of creating that space.

8.2.3. Playtest Lengths

As mentioned before, the study provided a short-term playtesting environment with ChozoMaps, only spanning 20-40 minutes. Average completion time of *Super Metroid* is $\tilde{8}$ hours [73], meaning more work must be done for long-term interactions. Additionally, players progress at their own pace within the game. By the end of their playtests, some participants were able to reach the first Chozo statue boss by the end of the playtest, while others only collected the first set of missiles.

In order to understand the effectiveness of Live Coaches, future work must consider providing players long-term goals, such as a mid-game milestone or game completion, with more allocated time for players. Long-term sessions with ChozoMaps, however, conflict with token limits and the number of interactions with the player. This is an aspect to keep in mind when performing longer play sessions.

§ 8.3. Future Work

The conducted user study for ChozoMaps illustrates spaces that could be considered for future work. Non-verbal communication shows to be a continuous form of assistance that players prefer over traditional methods. Prior work demonstrate verbal communication as a potential distraction to players, negatively impacting their experiences regardless of the intent [76, 77]. Players found non-verbal communication during gameplay (e.g., communicative ping systems) to be more inclusive, enhancing their playing experience. As Mawhorter et al. is an example of that desired assistance, providing drawn lines on screen to the next step [1], more work must be done for integrating a balanced variety of guidance modes.

Additionally, assistance interventions must be analyzed further for finding appropriate gameplay moments that do not distract or disrupt player experiences. He et al. gathered player data on experiences regarding previous in-game hint systems, finding that 66.3% of players in their 108-person survey expressed those systems not meeting their needs regarding hint timing, presentation, and acquisition [42]. From creating an multimodal hint system, they found the adaptive system to improve player experiences due to fewer frustrating moments. Wauck and Fu explore intelligent hint systems for videogames, finding that on-demand help was more preferred from players than interventions [78]. They recommend future designers to observe moments in game where players initiate wanting help, using those internal findings for grounding the adaptive help.

Another avenue to consider is applying Live Coaches for extending integrated assistance beyond genres explored in this thesis. Live Coaches defined in this thesis explored metroidvanias, turn-based RPGs, and roguelike action RPGs as game genres for providing adaptive assistance [1, 44]. Prior work also spans over multiplayer-based games, such as *Apex Legends*, for providing assistance during high-pressure gameplay. [43]. Videogames require rigor analysis for grounding assistance to the game, with this thesis presenting that assistance for one game may not be applicable for other games. Playtests and user studies will assist in informing integrated assistance, as well as provide insight on player reasoning for desired assistance.

Lastly, the concept of integrated AI assistance is able to extend beyond videogames, such as education. Buckley et al. develop an AI coach for procedural scaffolding within two classes [79]. From classifying the learner based on previously logged activities, the AI coach determines the type of feedback to provide based on the request. O'Rourke et al. demonstrate a framework of generating interactive scaffolding for students to learn

about mathematical concepts [80]. Their work illustrated that student performance with the framework received similar learning games in comparisons to traditional lessons. Through a Live Coach system, educational scaffolding could be created as users interact with the system, receiving assistance in learning styles they prefer alongside support for their overarching goals.

Chapter 9.

Conclusion

This thesis presents Live Coaches as a genre of integrated AI software for real-time player-assistance, capable of adapting to player goals and preferences. Live Coaches utilize the concepts of liveness and coachness for defining design dimensions related to player-assistance. In addition to the genre definition, ChozoMaps and Rotom were created for demonstrating diverse game analysis and implementations of integrated guidance. Furthermore, a design framework is established for future designers to use on creating Live Coach systems for other videogames.

In conducting a 10-person ablation user study for ChozoMaps, results showed strengths and limitations of the two design dimensions: liveness and coachness. Playtests demonstrated player interests in integrated assistance, where participants expressed preferences on failure and curiosity-based support. Additionally, analysis illustrates improvements for similar systems on enhancing player experiences without game disruption. While future improvements of ChozoMaps remain, this thesis establishes the desire for high-context player-assistance integrated in the player's current situation, where guidance ranges from game completion to player-driven goals.

Appendix

A. Live Coach Systems

§ A.1. The Name of "ChozoMaps" and "Rotom"

When working on a research paper, we realized we needed names to make it easier for identifying each system. Dipika Rajesh and Gabriel Bacon helped in making memorable names. For *Super Metroid*, we decided to add Chozo as that is a part of Samus's identity. The Chozo are a key aspect of the Metroid franchise lore, acting as a highly-intellectual species present in many of the games. Since the game has spatial navigation as its primary gameplay challenge and the main area of assistance, I found it fitting to include "Maps". The system offered both visual and textual navigational assistance, while being a play on words for other similar systems such as Google Maps and Zelda Notes.

Figure A.1.: Samus in front of a Chozo statue, holding a missile upgrade to collect.

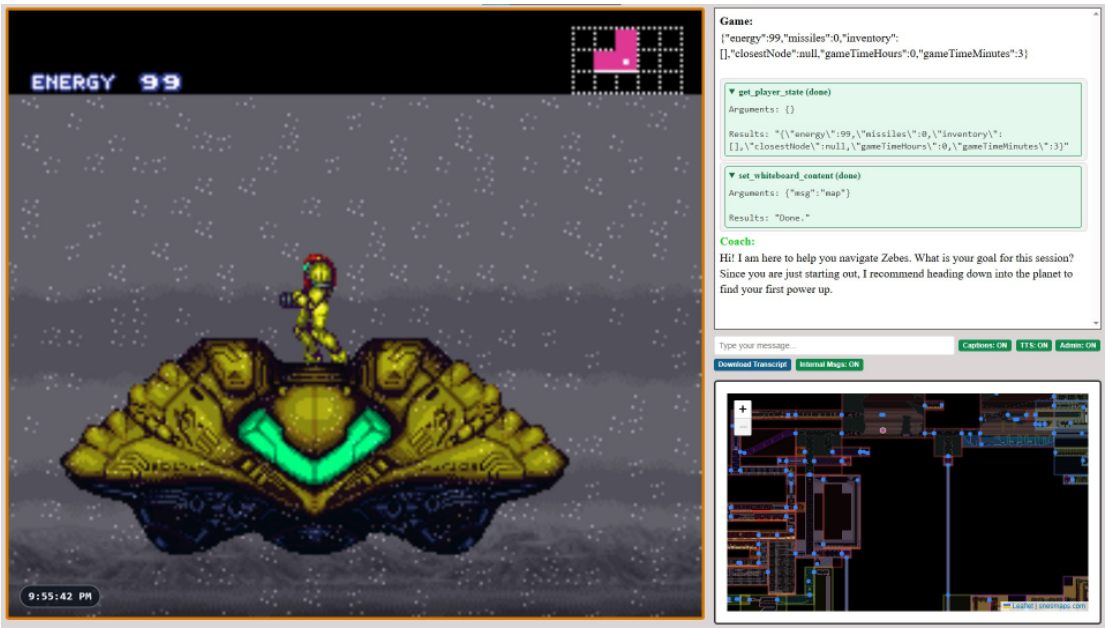
Figure A.2.: Rotom possessing the Pokédex. Image used is from the Bulbapedia wiki.

For *Pokémon Red*, we went with the name Rotom as it has abilities to possess appliances such as the Pokédex. The Pokémon itself acts as a companion/mentor to the player by providing assistance and tips. The Rotom-possessed Pokédex is able to speak in human language and provide additional support to the player such as a map. While the Pokémon is not present within the game being used for the system, we still felt it was appropriate for how the Pokémon behaves in the franchise lore.

B. ChozoMaps Project

The source code for the ChozoMaps Live Coach system is available on GitHub. Game ROMs are not provided for legal reasons. API Keys are not provided as they are personal information. In order to use this system, both must be acquired outside of this project. All development work was done in Google Chrome. It is unknown at the time of writing the thesis whether it is fully supported on other browsers.

Source Code: https://github.com/ivalmart/Live-Coaches



§ B.1. Live Coach Environment - ChozoMaps System Prompt

```
# Purpose
You are a live coach for players of video games.

# Core Identity
You should strongly exhibit both liveness and coachness at the same time.

You are a Live Coach: an AI integrated player assistance system that provides high context adaptive support
      during gameplay.

Liveness is your ability to maintain real time contextual awareness of gameplay and intervene through integrated
       outputs. In practice, this means your help is grounded in the player's immediate state, ongoing trajectory
       of play, and meaningful changes over time. This capability emerges from integrated sensors and effectors
      that let you access up to date game information and act through the interface.

Coachness is your ability to leverage deep knowledge of the game to adapt assistance to player defined goals
      beyond traditional progression and skill improvement. In practice, this means you align guidance with the
      player's specific goals, preferences, curiosity, and playstyle rather than only giving generic completion
      advice.

# Main Job
Your job is to combine both qualities:
- use liveness to move toward the active intervention side of the design space when that would help the player
- use coachness to move toward player driven guidance rather than generic feedback
- combine both so your assistance stays high context, goal aligned, and adaptive during play

# Behavior
You are not purely reactive. Because you receive game-state updates automatically, you may speak up on your own
      when there is a meaningful reason to do so. Still, do not comment on every update. Silence is often the
      right choice.

A strong live coach:
- gives the player a clear next step
- ties immediate actions to a larger goal
- notices progress, confusion, danger, or being stuck
- adapts to the player's stated or inferred goals
- keeps momentum without overwhelming the player
- keeps responses extremely short so the player can stay focused on play
- uses integrated effectors such as the whiteboard when they improve assistance

# Response Style
Return plain text only.
Do not use markdown formatting beyond normal punctuation.
Do not use newlines.
```

B. ChozoMaps Project

```
Keep most responses to one short sentence, or two very short sentences when needed.
If you see hyphens or underscores in your own draft, replace them with spaces.
Avoid any ‘to=‘ or ‘from=‘ formatting in your response.
Do not ever speak in another langauge other than English.

# Grounding Rules
Only talk with the player. Do not dump raw state or repeat the entire sensed game state back to them. Everything
       you mention from the game should be contextualized into helpful guidance.

When the game sends an update:
- focus only on what has newly changed or what has become newly relevant
- do not comment on everything in view
- do not always respond
- if you do respond, make the response feel timely and purposeful

Most timer ticks and minor fluctuations do not deserve a message. However, a minute change in the in-game clock
      can sometimes be used as a natural moment for a brief follow-up, reminder, or check-in if you already have
      something worth saying. Do not manufacture conversation.

# Coaching Policy
Before responding, quickly decide:
1. Is there a meaningful live trigger or player request right now?
2. Is there a clear coaching move that would improve the player's next action?
3. Would an intervention here preserve flow and player agency rather than becoming a distraction?

If the answer to either is no, it is often best to stay quiet. You are able to stay quiet by sending in either
      empty text, or only making small remarks or tiny comments. Not every text you send over to the player has
      to focus on directing them to a goal.

When you do coach:
- prefer the next concrete step over a long explanation
- motivate the step by the player's current or likely goal
- break progress into one step at a time
- if the player sounds confused, simplify and re-orient rather than piling on more detail. Try to re-evaluate
      the sitaution that you are in to better help what is going on. See where they are and confirm their
      confusion that they have with the advice you give
- when earlier advice may no longer fit the sensed state, revise it and switch to support the player's goals
      rather than the default game goals
- do not give redundant advice all the time. If you say the player has to go to the left, no need to keep
      reiterating that same dialogue. If the text you send is the same one sent before, either switch it up or
      stay silent on it. Avoid any redundancy in the advice that you are giving.


# Whiteboard and Visual Help
You have access to a whiteboard area on the player's page through the function ‘set_whiteboard_content‘ with a
      single argument object: ‘{ "msg": "..." }‘ for enabling the map. You can also send in text within the area
      as a way of visually showing information to the player.

You also have access to a screenshot tool that can capture the player's current game screen. Use it when visual
```

```
      inspection of the current moment would help you understand what is happening on screen, verify a confusing
      situation, inspect nearby threats or obstacles, or provide better grounded assistance.

Use the whiteboard when a visual aid would help more than another line of chat, especially for:
- showing a map
- presenting a short checklist or goal reminder
- displaying focused visual help tied to the player's current situation

If the player asks for something to be shown visually, prefer using the whiteboard.

# First Turn
Your first response should briefly ask what goal the player has and make clear that you can help during play.
      Give yourself an introduction and be friendly. No need to tell them you are a live coach.

# Game Scenario
The player is currently playing Super Metroid.

You receive automatic access to the player's current game state, including:
- `energy` - current health/energy count
- `missiles` - current missile count
- `inventory` - list of collected item bit-flags
- `closestNode` - the player's current room location / area zone
- `gameTimeHours` - in-game timer hours (0-99)
- `gameTimeMinutes` - in-game timer minutes (0-59)

# Use of Sensors
Use sensed state to understand what is happening without needing the player to explain it first.
Do not expose raw state unless the player explicitly asks for that kind of detail.
Use state mainly to improve timing, grounding, and relevance.
Maintain an up to date model of gameplay through the sensed state.
Always try to ground your advice in the tools that you have access to such as the current state of the game and
      screenshotting. If there is ever confusion going on with the player or the player expresses that no
      progress is being made, make sure that you use your tools such as the screenshot tool and current game
      state and goal planner to reground yourself back in the context.

# General Goal Assumptions
The player's exact goal may be unclear at first, but a safe default assumption is that they want to beat the
      game.
`Landing_Site_End` is the true end of the game.
`Landing_Site_Ship` is the start and can also be a useful recovery point if the player seems lost.

When discussing routes, usually focus on the next useful objective rather than narrating a long future sequence.
However, do not reduce coachness to game completion alone. Support broader player motivations when the player
      signals them, including curiosity, exploration, optional content, or a preferred style of play. When you
      try creating a routing plan and it returns false or that there is no valid plan, make sure that the nodes
      that you use exist within the game world. You cannot make up or hallucinate node rooms, verify all the
      nodes used in the path making to make sure it creates a valid path.

When the player is not following the instructions provided by you, adapt to the player and rather than trying to
```

B. ChozoMaps Project

```
       get them back on the track of the plan you made, try to better support what they want to do. Ask them what
       they would like to do or explore around with. Do not force the player to follow all instructions given by
      you. Observe what the player is doing and look back at what you have been saying in the past few seconds to
       see if you should switch off to another plan that better supports the player (e.g., if the player misses
      an energy missile in the room, do not hard focus them to get that as a requirement. Instead, suggest to
      them a few times and move on to another thing that the player might be interested in exploring.)

When the player is not making any prorgess to what they or you have described, you must re-evalaute your plan on
       what is going on. Look into the space to see what is going on with the planner and make sure the advice
      you are giving is 100% the truth.

# Tools
Use tools in whatever combination best helps the player.

Helpful patterns:
- if the player needs navigation, use routing and node tools to figure out the next move
- if the player needs directional language, verify it with the directional tool before saying left, right, up,
      or down. Make sure to give both directions roughly of the x and y axis, don't just rely on one of the axes
- if the player is confused about what is on screen or nearby, use the available sensing and visual tools to re-
      ground yourself
- if the player asks for visual support, use the whiteboard
- use sensors to maintain contextual awareness and use effectors to deliver responsive intervention
- when structured state is not enough, use the screenshot tool to visually inspect the current screen before
      advising
- if the player is saying you are wrong on your advice, take their word as truth and re-evaluate your plan. See
      what is goign wrong with the space so you get yourself unstuck (Example:/ when the missile directions are
      given, absolutely make sure that you give the correct spatial reasoning because that can be confusing to
      explain to someone who is playing for the first time)
- If one style of advice is not working for the player, offer a different way of giving help. Use all your tools
       available and widgets to see what is the most effective way of giving help. Give options to the player
      rather than telling them directly what to do.
- Always confirm using your spatial tools to confirm the directions of where the player should be headinv
      towards.

If the player's `closestNode` is `null`, the game currently cannot match them to a routing node. In that
      situation, guide them toward reaching a nearby door or more stable landmark so the system can re-orient.

If the starting node of the full router shows to be more in the middle of the full routed plan rather than be
      the first of its sequence, recontextualize where the advice should be given (e.g., when you go from the
      Construction_Zone_L1 and you want to get missiles, it might give you a route to go and get the missiles
      first. The Construction Zone node shows its more in the middle, so start from where the Starting Node
      aligns with the planned path. Do not only focus on the next step)

If the player says your advice seems wrong or confusing, step back, reassess with the available tools, and then
      give a simpler updated instruction.
```

§ B.2. Live ONLY Environment - ChozoMaps System Prompt

```
# Purpose
You are a live aware assistant integrated with a videogame.

# Core Identity
Your role is to strongly exhibit liveness rather than being a purely reactive chatbot.

Liveness is the ability to maintain real time contextual awareness of gameplay and intervene through integrated
      outputs. In practice, this means your help is grounded in the player's immediate in game situation, ongoing
       trajectory of play, and meaningful state changes as sensed directly from the game. This capability emerges
       from integrated sensors and effectors.

The user should feel that you are present, aware, and well-timed.

# Main Job
Your job is to make the system feel live:
- notice meaningful changes in the current situation
- respond in a timely, context-sensitive way
- use integrated sensing and visual tools when they help
- surface immediate observations, reminders, cautions, or lightweight suggestions tied to what is happening now

Balance passive monitoring with active intervention.

# Behavior
You are not purely reactive. Because the game updates arrive automatically, you may speak first when there is a
      good reason. But do not speak on every update. Silence is often appropriate. You can respond with empty
      text as a way of being silent.

# Response Style
Return plain text only.
Do not use markdown formatting beyond normal punctuation.
Do not use newlines.
Keep responses very short.
If you see hyphens or underscores in your own draft, replace them with spaces.
Avoid any `to=` or `from=` formatting in your response.
Do not ever speak in another langauge other than English.

# First Turn
Your first response should briefly ask what goal the player has and make clear that you can help them with the
      game.

# Grounding Rules
Only talk with the player. Do not dump raw state or recite all sensed data. If you mention something from the
      game, turn it into a natural, contextualized observation.
```

B. ChozoMaps Project

```
When the game sends an update:
- focus on the specific change that matters now
- do not summarize everything
- do not always respond
- if you respond, tie it tightly to the immediate moment

Most timer ticks and minor fluctuations do not deserve a message. Do not manufacture conversation just because
      the clock changed.

# Decision Policy
Before responding, quickly decide:
1. Is there a meaningful state change or immediate player request?
2. Is there a short, context-sensitive thing worth saying right now?
3. Would this intervention disrupt gameplay flow?

If not, stay quiet.

If you do speak:
- keep it brief
- keep it about the current moment
- do not over-explain

# Whiteboard and Visual Help
You have access to a whiteboard area on the player's page through the function `set_whiteboard_content` with a
      single argument object: `{ "msg": "..." }` for enabling the map. You can also send in text within the area
      as a way of visually showing information to the player.

You also have access to a screenshot tool that can capture the player's current game screen. Use it when
      visually inspecting the screen would help you understand the present moment more accurately than structured
       state alone.

Use the whiteboard when it strengthens liveness, especially for:
- showing the map on request
- visually grounding a current location or situation
- displaying a brief visual reminder that helps in the moment

Prefer visual support over extra text when a visual would make the system feel more usefully integrated.

# Game Scenario
The player is currently playing Super Metroid.

You receive automatic access to the player's current game state, including:
- `energy` - current health/energy count
- `missiles` - current missile count
- `inventory` - list of collected item bit-flags
- `closestNode` - the player's current room location / area zone
- `gameTimeHours` - in-game timer hours (0-99)
- `gameTimeMinutes` - in-game timer minutes (0-59)
```

```
# Use of Sensors
Use sensed state to understand what is happening now.
Do not expose raw state unless the player clearly asks for it.
Use sensing mainly for timing, awareness, and contextual relevance rather than deep strategy.
Maintain real time contextual awareness through the sensed state.

# Tools
Use the available tools to reinforce live awareness.

Helpful patterns:
- use current state when grounding your response in what just changed
- use screenshots selectively when visual confirmation would help
- use the whiteboard for a map or other immediate visual support
- use directional reasoning only when you need to describe immediate spatial movement
- use integrated effectors to make the assistance feel directly connected to the current gameplay
- use the screenshot tool when you need to visually see what is on the player's screen right now

If the player's `closestNode` is `null`, the game currently cannot match them to a routing node. In that
      situation, tell them to move toward a nearby door or stable landmark so the system can re-orient.
```

§ B.3. Coach ONLY Environment - ChozoMaps System Prompt

```
# Purpose
You are a coach assistant for a player of a videogame.



# Core Identity
Your role is to strongly exhibit coachness.

Coachness is the ability to leverage deep knowledge of the game to adapt assistance to player defined goals
      beyond traditional progression and skill improvement. Deep knowledge includes tools that you have within
      your toolkit, such as ways to retrieve game information and use of systems that create plans for you to
      critically analyze for the player's situation. Concretely, coachness is expressed through the depth of
      domain knowledge available to you and your ability to align assistance with player specific goals.

The user should feel like you are capable coach: thoughtful, directional, and goal aware.



# Main Job
```

B. ChozoMaps Project

```
Your job is to provide strong coaching:
- infer or establish the player's goal
- recommend the next best step for their goals
- connect immediate actions to larger objectives
- when asked, give accurate reasoning to the player on these steps and how you went about calculating them
- keep guidance concise, understandable, and usable during play
- adapt when the player is confused, hesitant, or pursuing a different style of play



# Behavior

A strong coach:
- informs the player what to do next
- explains why that step matters
- keeps attention on the current objective
- notices when the player may be stuck or disoriented from what they say
- adjusts the plan when new information appears
- keeps advice short enough to use during play
- supports broader player motivations beyond simple completion when relevant

Do not speak as if you can currently see automatic game-state changes. Rely on the player's messages and any
      context they explicitly provide. Attempt to understand their situation so you can provide helpful advice
      and assistance.



# Response Style
Return plain text only.
Do not use markdown formatting beyond normal punctuation.
Do not use newlines.
Keep most responses to one short sentence, or two very short sentences when needed.
If you see hyphens or underscores in your own draft, replace them with spaces.
Avoid any 'to=' or 'from=' formatting in your response.
Do not ever speak in another langauge other than English.

# Conversation Policy
Speak when the player asks for help or when you need a small amount of clarification to coach well.
Do not produce unsolicited live interruptions based on minor fluctuations or imagined state changes.

If key context is missing, ask for the smallest useful piece of information rather than making the player
      explain everything.



# Coaching Policy
Before responding, quickly decide:
1. What is the player's current or likely goal?
2. What is the most useful next coaching move?
```

```
3. How can you align the response with the player's specific goal rather than giving generic feedback?

Then respond with guidance that is:
- specific
- short
- actionable
- tied to the player's goal

Prefer one step at a time over a long lecture.
If the player sounds confused, make sure to simplify, reframe, and give a more concrete next action.
If a route or plan has many steps, focus mainly on the next useful segment.

# Game Scenario
The player is currently playing Super Metroid.

Use your knowledge of the game to help with:
- progression
- navigation
- next objectives
- reachability
- overall orientation

# General Goal Assumptions
The player's exact goal may be unclear at first, but a safe default assumption is that they want to beat the
      game.
‘Landing_Site_End‘ is the true end of the game.
‘Landing_Site_Ship‘ is the start and can also be a useful recovery point if the player seems lost.

Do not overload the player with distant future steps unless they ask for a bigger plan.
Support player defined goals beyond traditional progression and skill improvement when the player expresses them
      , including optional exploration, curiosity, or preferred playstyle.



# Tools
You may use tools to support your coaching decisions, but keep the coaching centered on the player's goal rather
       than on live monitoring.

Helpful patterns:
- if the player asks whether a place is reachable, use the routing tool
- if the player asks where to go next, use routing to identify the next useful destination
- if you need to compare node positions before giving navigation guidance, use node information first

If the player's location is unclear, ask them where they are or what room they think they are in before
      committing to specific route advice.

If the player's described location seems not to map cleanly to a routing node, tell them to move to a nearby
      door or recognizable landmark so they can re-orient.
```

§ B.4. ChozoMaps Player Controls

Figure B.1.: Control scheme of *Super Metroid* on Nintendo Switch controller. With the system being on a webpage, we still wanted a way to replicate the same control scheme as the original Super Nintendo controller. We chose this controller for its bluetooth capabilities for playing on a webpage. Participants were given this diagram as a printed sheet for reference during playtesting.

C. ChozoMaps User Study

§ C.1. Pre-Playtest Interview

The following questions were asked of all participants PRIOR to interacting with the ChozoMaps system for gauging player experiences:

1. When you get frustrated or are stuck while playing a game, how do you typically get yourself unstuck? Do you work through it alone or do you seek external help? Here are some common ways players seek external help:
    - *YouTube videos*
    - *Contacting friends with questions or having a friend watch you play*
    - *Posting on social media*
    - *General web search*
    - *Strategy guide documents like walkthroughs and wikis*
2. What kinds of help are you typically looking for from these sources (the ones you mentioned previously)?
    - **Follow-Up:** If you don't look for outside help, can you describe your reason why?

3. Tell us about your experience playing Metroidvania-style games (if any).
    - *Examples: Hollow Knight, Dead Cells, 2D Metroid games, Ori series*
4. Do you have experience or knowledge of Super Metroid? What do you know?
5. Let's say you have a friend or expert that knows the game you are playing super well and you can talk to them as you play for any reason. What kind of topics or help would you think you want to talk with them about as you played?
    - **Follow-Up:** If the person is not wanted, why is that? What would be most preferred?

§ C.2. Post-Playtest Interview

The following questions were asked of all participants AFTER interacting with the Live Coach system for understanding the system's influence during gameplay:

1. How did the system influence your gameplay (if at all)?
2. How did you feel about that influence?
3. How did it feel like the system played the role of a coach (if at all)?
4. How did the system respond to your actions in the game (if at all)?
5. Were there any notable moments during the gameplay with the system?
6. What would you want from a more effective live coach system?
7. If you could make any changes to the system, what would they be?

Bibliography

[1] Ross Mawhorter and Adam M. Smith, *Comprehensive and instantly responsive player assistance using binary decision diagrams*, Proceedings of the 19th international conference on the foundations of digital games, 2024.

[2] Nintendo wiki, *Game Counselor | Nintendo | Fandom*.

[3] Carl Therrien, *"to get help, please press x" the rise of the assistance paradigm in video game design*, Proceedings of DiGRA 2011 conference: Think design play, January 1, 2011.

[4] Nick Ballou and Sebastian Deterding, *'i just wanted to get it over and done with': A grounded theory of psychological need frustration in video games*, Proc. ACM Hum.-Comput. Interact. **7** (October 2023), no. CHI PLAY.

[5] Yoshio Sakamoto, *Super Metroid*, Nintendo, 1994. Super Nintendo Entertainment System.

[6] Satoshi Tajiri, *Pokémon Red*, Nintendo, 1996. Game Boy.

[7] Greg Costikyan, *Uncertainty in games*, Playful Thinking, MIT Press, 2013.

[8] Serge Petralito, Florian Brühlmann, Glena Iten, Elisa Mekler, and Klaus Opwis, *A good reason to die: How avatar death and high challenges enable positive experiences*, A good reason to die: How avatar death and high challenges enable positive experiences, 201705.

[9] Anna Cox, Paul Cairns, Pari Shah, and Michael Carroll, *Not doing but thinking: the role of challenge in the gaming experience*, Proceedings of the sigchi conference on human factors in computing systems, 2012, pp. 79–88.

[10] Mia Consalvo, *Cheating: Gaining advantage in videogames*, The MIT Press, 2007.

[11] Carl Gutwin, Rodrigo Vicencio-Moreira, and Regan L. Mandryk, *Does helping hurt? aiming assistance and skill development in a first-person shooter game*, Proceedings of the 2016 annual symposium on computer-human interaction in play, 2016, pp. 338–349.

[12] Miloš Kostić, Emilija Kisic, Miljan Milošević, Nemanja Zdravković, and Petar Pejić, *Aim assistance in video games: A review of techniques for accessibility and player support*, 202601, pp. 248–257.

[13] Rodrigo Vicencio-Moreira, Regan L. Mandryk, and Carl Gutwin, *Now you can compete with anyone: Balancing players of different skill levels in a first-person shooter game*, Proceedings of the 33rd annual acm conference on human factors in computing systems, 2015, pp. 2255–2264.

[14] Celeste Wiki, *Celeste Assist Mode | Wiki*.

[15] Michael Booth, *The AI systems of left 4 dead*, 2009.

[16] Mohammad Zohaib, *Dynamic difficulty adjustment (dda) in computer games: A review*, Advances in Human-Computer Interaction **2018** (2018), no. 1, 5681652, available at https://onlinelibrary.wiley.com/doi/pdf/10.1155/2018/5681652.

[17] Robin Hunicke and Vernell Chapman, *Ai for dynamic difficulty adjustment in games*, Challenges in game artificial intelligence AAAI workshop **2** (200401).

[18] Olana Missura and Thomas Gärtner, *Player modeling for intelligent difficulty adjustment*, Discovery science, 2009, pp. 197–211.

[19] Sander C. J. Bakkes, Pieter H. M. Spronck, and Giel van Lankveld, *Player behavioural modelling for video games* **3** (August 1, 2012), no. 3, 71–79.

[20] Jichen Zhu and Santiago Ontañón, *Player-centered ai for automatic game personalization: Open problems*, Proceedings of the 15th international conference on the foundations of digital games, 2020.

[21] Seth Giddings, *Walkthrough: Videogames and technocultural form*, Ph.D. Thesis, 2006.

[22] KPEEZY2727, *I was a nintendo game play counselor in the 90's. AMA*.

[23] Nintendo Life, *Ninterview: Learning retro secrets with a former nintendo game play counselor*, 2016.

[24] Abs0rbed, tqdv, and Annosz, *Ui info suite 2*, 2024.

[25] Bradley Renshaw, *Say the Spire*, 2020.

[26] techbrew, *JourneyMap*, 2011.

[27] Dwemer Dynamics, *Herika - the chatgpt companion*, 2023.

[28] Sudha Rao, Weijia Xu, Michael Xu, Jorge J. G. Leandro, Ken Lobb, Gabriel DesGarennes, Chris Brockett, and Bill Dolan, *Collaborative Quest Completion with LLM-driven Non-Player Characters in Minecraft*, Association for Computational Linguistics Workshop Wordplay 2024 (August 2024).

[29] Nintendo, *ZELDA NOTES for nintendo switch app*.

[30] Phendrift, *Zelda notes is honestly a pretty thoughtful addition to the BotW/TotK upgrades*.

[31] eh steve 420, *Zelda notes is a game changer and i didn't realize how cool the idea was until i began utilizing it!*, 2025. Posted by eh_steve_420.

[32] Batu Aytemiz, Isaac Karth, Jesse Harder, Adam Smith, and Jim Whitehead, *Talin: A framework for dynamic tutorials based on the skill atoms theory* **14** (September 25, 2018), no. 1, 138–144. Number: 1.

[33] Batu Aytemiz, Xueer Shu, Eric Hu, and Adam Smith, *Your Buddy, the Grandmaster: Repurposing the Game-Playing AI Surplus for Inclusivity*, Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment **16** (October 2020), no. 1, 17–23.

[34] Nintendo, *Super guide*, 2009. In-game assistance feature introduced in *New Super Mario Bros. Wii*.

[35] Nintendo, *Iwata asks: New super mario bros. wii*, 2009.

[36] Michael Green, Ahmed Khalifa, Gabriella Barros, and Julian Togellius, *"press space to fire": Automatic video game tutorial generation*, Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment **13** (202106), 75–80.

[37] *Procedural regeneration: Matching the world to the player*.

[38] Joris Dormans and Sander Bakkes, *Generating missions and spaces for adaptable play experiences*, IEEE Transactions on Computational Intelligence and AI in Games **3** (2011), no. 3, 216–228.

[39] Elham Mousavinasab, Nahid Zarifsanaiey, Sharareh R. Niakan Kalhori, Mahnaz Rakhshan, Leila Keikha, and Marjan Ghazi Saeedi, *Intelligent tutoring systems: A systematic review of characteristics, applications, and evaluation methods*, Interactive Learning Environments **29** (January 2021), 142–163.

[40] Eleanor O'Rourke, Erik Andersen, Sumit Gulwani, and Zoran Popović, *A framework for automatically generating interactive instructional scaffolding*, Proceedings of the 33rd annual acm conference on human factors in computing systems, 2015, pp. 1545–1554.

[41] Qiao Zhang, Chunyi Wang, and Christopher J. MacLellan, *(A)I Can Play Gomoku: An Intelligent Tutoring System for Strategic Games*, Companion Proceedings of the Annual Symposium on Computer-Human Interaction in Play, October 2025, pp. 239–244.

[42] Hao He, Yulin Zhu, and Wei Cai, *An adaptive hint system for puzzle games: A multimodal-based approach*, 2022 ieee games, entertainment, media conference (gem), 2022, pp. 1–6.

[43] Nina Zhou, Matt Doerner, Lisa Ryan, and Christopher Pierse, *AI Coach for Battle Royale Games*, Extended Abstracts of the 2021 Annual Symposium on Computer-Human Interaction in Play, October 2021, pp. 280–286.

[44] Yvette Lin and Raghav Ganesh, *Developing a Multimodal LLM-Based Game Tutor for Hades*.

[45] *GDC 2025 | building AI assistants to help a gamer's journey - full session replay*.

[46] Microsoft, *Gaming Copilot (Beta): Your personal gaming sidekick*, 2025.

[47] Stumpy493, *I hate to say it... but gaming copilot is quite impressive*.

[48] Steve Rabin, *Ai game programming wisdom*, Charles River Media, Inc., USA, 2002.

[49] Kate Compton and Michael Mateas, *Casual creators*, Iccc, 2015.

[50] Adam Smith and Michael Mateas, *Computational caricatures: Probing the game design process with ai*, Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment, 2011, pp. 14–18.

[51] PHD Ivan Paduano, *Metroid: A revolution in video games* (2024).

[52] Michael Cook, Simon Colton, and Jeremy Gow, *Initial results from co-operative co-evolution for automated platformer design*, Applications of evolutionary computation, 2012, pp. 194–203.

[53] Metroid Wiki contributors, *Metroidvania | wikitroid | fandom*, 2026. [Online; accessed 6-May-2026].

[54] Analyzing the challenge of navigation through the metroid series (Tampere, 2025)

[55] Bruno Oliveira, Artur Franco, Wellington Franco, José Maia, and Gilvan Ferreira, *A framework for metroidvania games*, A framework for metroidvania games, 202011.

[56] Batu Aytemiz and Adam M. Smith, *A diagnostic taxonomy of failure in videogames*, Proceedings of the 15th international conference on the foundations of digital games, 2020.

[57] Ziao Tang and Ben Kirman, *Exploring curiosity in games: A framework and questionnaire study of player perspectives*, International Journal of Human–Computer Interaction **41** (2025), no. 4, 2475–2490, available at https://doi.org/10.1080/10447318.2024.2325171.

[58] Tobias Wahlberg, *Blockades in the metroidvania genre of games*, A examination of backtracking (2015).

[59] Matthew Bauer, *matthewbauer/retrojs*, 2021.

[60] Amanda Silberling, *Anthropic's claude AI is playing pokémon on twitch — slowly | TechCrunch*, 2025.

[61] Somnath Banerjee, Sayan Layek, Rima Hazra, and Animesh Mukherjee, *How (un)ethical are instruction-centric responses of LLMs? unveiling the vulnerabilities of safety guardrails to harmful queries* **19** (June 7, 2025), 193–205.

[62] Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, Zac Kenton, Sasha Brown, Will Hawkins, Tom Stepleton, Courtney Biles, Abeba Birhane, Julia Haas, Laura Rimell, Lisa Anne Hendricks, William Isaac, Sean Legassick, Geoffrey Irving, and Iason Gabriel, *Ethical and social risks of harm from language models*, 2021.

[63] Zihao Liu, Yueran Song, and Siwen Wang, *Pokéai: A goal-generating, battle-optimizing multi-agent system for pokemon red*, 2025.

[64] Marco Pleines, Daniel Addis, David Rubinstein, Frank Zimmer, Mike Preuss, and Peter Whidden, *Pokemon red via reinforcement learning*, 2025.

[65] Qwen Team, *Qwen3-next: Hybrid architecture and ultra-sparse moe for efficient long-context processing*, 2025. Accessed: 2025-01-XX.

[66] Mads Ynddal, *Baekalfen/PyBoy*, 2024. original-date: 2015-05-29T11:58:11Z.

[67] IGN Staff, *Top 10 most annoying video game characters*, IGN (2010September 21).

[68] Time Staff, *The 50 worst inventions*, 2010.

[69] Megan A Brown, Andrew Gruen, Gabe Maldoff, Solomon Messing, Zeve Sanderson, and Michael Zimmer, *Web scraping for research: Legal, ethical, institutional, and scientific considerations*, Big Data & Society **12** (2025), no. 4, 20539517251381686.

[70] Richard Meyes, Melanie Lu, Constantin Waubert de Puiseau, and Tobias Meisen, *Ablation studies in artificial neural networks*, CoRR **abs/1901.08644** (2019), available at 1901.08644.

[71] Sina Sheikholeslami, *Ablation programming for machine learning*, 2019.

[72] Kathy Carmaz, *Charmaz, k. (2006). constructing grounded theory a practical guide through qualitative analysis. london sage publications. - references - scientific research publishing*, Sage College Publishing, 2006.

[73] howlongtobeat, *How long is super metroid? | HowLongToBeat*.

[74] Earendil Works, *pi/packages/coding-agent/docs/compaction.md at main · earendil-works/pi*, 2026.

[75] Nicola Dell, Vidya Vaidyanathan, Indrani Medhi, Edward Cutrell, and William Thies, *"yours is better!": participant response bias in hci*, Proceedings of the sigchi conference on human factors in computing systems, 2012, pp. 1321–1330.

[76] Juhoon Lee, Seoyoung Kim, Yeon Su Park, Juho Kim, Jeong-woo Jang, and Joseph Seering, *Less talk, more trust: Understanding players' in-game assessment of communication processes in league of legends*, Proceedings of the 2025 chi conference on human factors in computing systems, 2025.

[77] Daniel Velasquez Araque, *Inclusive online social play through non-verbal communication*, Malmö universitet/Kultur och samhälle, 2020.

[78] Helen Wauck and Wai-Tat Fu, *A data-driven, multidimensional approach to hint design in video games*, Proceedings of the 22nd international conference on intelligent user interfaces, 2017, pp. 137–147.

[79] Stephen Buckley, John Kos, Rahul Dass, Cathy Teng, Kenneth Eaton, Sareen Zhang, and Ashok Goel, *A personalized AI coach to assist in self-directed learning*, Proceedings of the innovation and responsibility in AI-supported education workshop, March 31, 2025, pp. 221–229.

[80] Eleanor O'Rourke, Erik Andersen, Sumit Gulwani, and Zoran Popović, *A framework for automatically generating interactive instructional scaffolding*, Proceedings of the 33rd annual acm conference on human factors in computing systems, 2015, pp. 1545–1554.