MultiModalMan: An Interactive Hangman Game

Andrea Tarricone
tarricone.1888181@studenti.uniroma1.it
University ”La Sapienza”
Rome, Lazio, Italy

Lucian Dorin Crainic
crainic.1938430@studenti.uniroma1.it
University ”La Sapienza”
Rome, Lazio, Italy

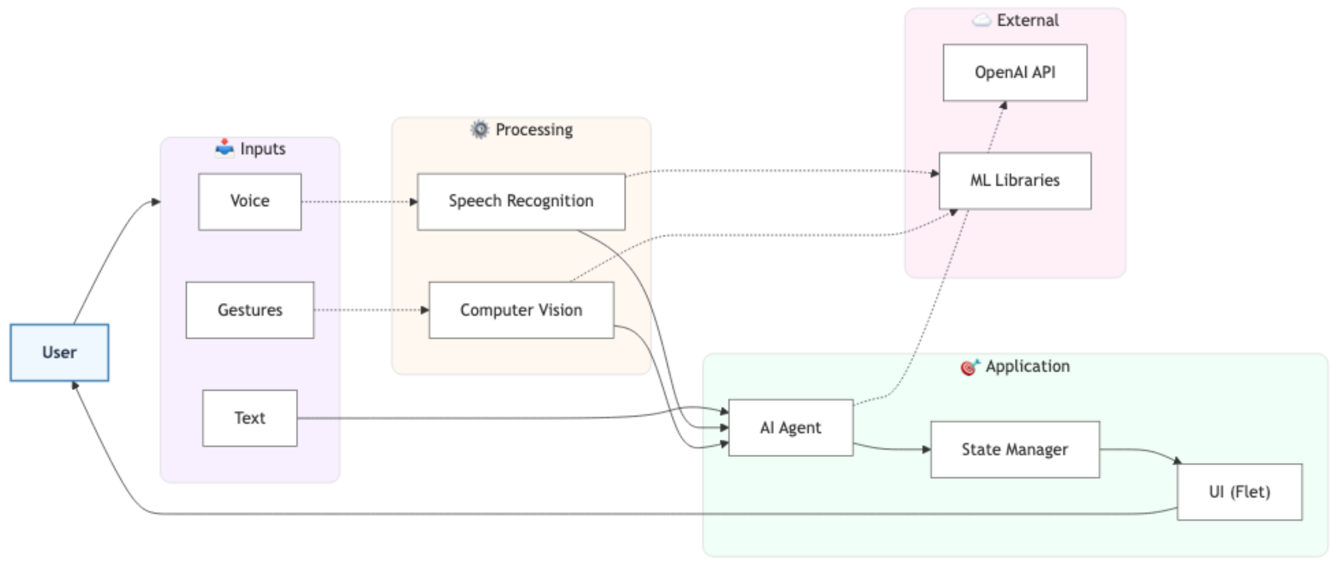

Figure 1. High-level overview of the multimodal hangman game system showing the flow from user inputs through processing layers to the core application.

Abstract

MultiModalMan is a re-imagining of the classic hangman game that lets players guess letters and control the interface by speaking or using hand gestures as well as by typing. The system is built in Python around three modules: a state manager that synchronises all inputs, an AI agent for natural-language processing, and a computer-vision pipeline for real-time gesture recognition. Users can switch freely between modalities during play, and tests show the approach improves accessibility and user satisfaction without sacrificing speed or accuracy. Future work will explore multilingual vocabularies, spoken feedback, and cooperative multiplayer play.

1 Introduction

Digital games have become a mainstream medium for both entertainment and learning, yet they remain unevenly playable. Industry reports estimate that *20–30 %* of the global player base lives with at least one disability, and almost half of those players already engage with games despite a range of access barriers [1]. Ensuring that games are accessible is therefore both an ethical imperative and a clear educational and commercial opportunity.

Research in multimodal human–computer interaction (HCI) shows that combining complementary input channels—speech, gesture, gaze, or haptics—lowers access barriers and improves task performance. A recent survey notes consistent reductions in error rate, task time, and cognitive load when users can blend modalities rather than rely on a single channel [2]. Game-specific studies echo these findings: gesture-based controllers enable players with motor impairments to execute directional actions that would be impossible on a traditional keyboard [5], and experiments that fuse voice commands with hand gestures yield the highest efficiency and perceived naturalness among the tested conditions [3]. Meanwhile, advances in AI speech processing continue to broaden accessibility by adapting to diverse accents and speech differences [4].

MultiModalMan responds directly to this research agenda. The project re-imagines the classic Hangman game as a language-learning tool that can be played interchangeably through:

- spoken commands processed by an AI speech pipeline;
- hand-gesture input captured via a webcam and MediaPipe;
- conventional keyboard typing.

An embedded AI agent maintains the dialogue flow and adapts hints in real time, while a finite-state machine keeps all three modalities synchronised so players can switch fluidly between them without losing context. The project pursues three goals:

1. demonstrate how a lightweight multimodal pipeline can be integrated into a small educational game;
2. provide meaningful accessibility options for users with motor or speech restrictions;
3. offer a concrete test-bed for the design claims found in multimodal HCI literature.

The remainder of this report details every phase of the project—from refined requirements to architecture, multimodal input handling, interface design, testing, and future work—so that our design decisions remain transparent and reusable.

2 Project Outline

This report is organised into nine chapters, each detailing a distinct stage in the development of *MultiModalMan*. Together they guide the reader from the earliest requirements to the final evaluation and future prospects.

3 Aim and Scope

MultiModalMan pursues a twofold mission: to illustrate how multimodal interfaces lower accessibility barriers and to turn that insight into an engaging, language-learning game. Traditional mouse-and-keyboard designs routinely sideline entire user groups. Blind players cannot perceive graphical feedback; Deaf players can read on-screen text but gain little from audio narration; users with motor impairments may struggle with precise key presses. By blending speech, gesture and text input, our project offers redundant—and therefore more inclusive—paths through the game.

3.1 Accessibility Objectives

Multiple channels
Voice, hand gestures and on-screen buttons are interchangeable so that no single sense or motor skill becomes a blocker.

Low reading load
Visual icons and spoken prompts replace verbose text wherever possible.

Complementarity
When one modality fails (e.g. speech in a noisy room) another can take over without interrupting play.

3.2 Educational & Inclusive Goals

Broad reach
Children, language learners and users with motor or speech constraints can all follow the same storyline.

Multisensory learning
Players reinforce spelling via simultaneous visual, auditory and kinaesthetic cues.

3.3 Technical Targets

Accurate input
Real-time recognition of spoken letters and hand-drawn glyphs.

Context-aware logic
An AI agent that interprets free-form utterances and drives adaptive hints.

Unified control
A central state manager that keeps the three modalities synchronised and responsive.

4 Requirements

The following tables specify the essential capabilities (*Functional*) and quality targets (*Non-Functional*) that *MultiModalMan* must meet to deliver an inclusive, responsive and maintainable experience.

Table 1. Functional requirements

ID	Description
F1	The user can start a new game at any time.
F2	The system selects a random word or User suggests a word to the AI agent.
F3	Letters can be guessed through voice commands, hand gestures or keyboard input.
F4	The interface shows the current word state and the number of wrong attempts.
F5	The system announces the outcome (win or loss) when the game ends.
F6	A new round can be launched without restarting the application.

Table 2. Non-functional requirements

ID	Description
NF1	The user interface must be clear and readable for all age groups.
NF2	Gesture recognition covers the full 26-letter English alphabet
NF3	The game logic remains consistent when inputs are invalid or ambiguous.

5 Multimodal Interaction

Multimodal interaction is the cornerstone of *MultiModalMan*. Instead of locking players into a keyboard-and-mouse loop, the game accepts spoken commands, hand-drawn letters captured by a webcam, and classic text input. A Python state manager keeps these channels in sync, while an OpenAI-powered agent interprets free-form utterances and adapts feedback to the current game phase.

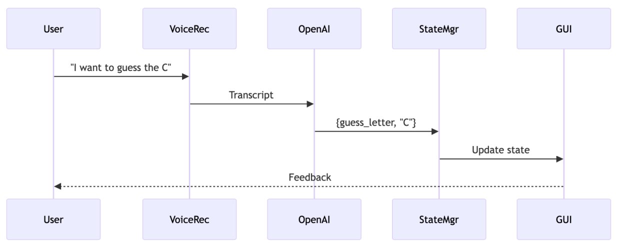

Figure 2. State machine diagram for the `GameStateManager` class, showing the transitions between different game states based on user inputs and actions.

5.1 Voice Interaction

When the user speaks—"start game", "guess C", or "exit" for instance—the audio stream is transcribed in real time (primary engine: `speech_recognition`; fallback: Whisper or the Google Speech API). The recognised text is passed to the state manager, which checks whether the utterance is valid for the current state (setup, guessing, or end-of-game) and routes it accordingly. Natural phrases such as "how many errors do I have?" are interpreted by the OpenAI agent, which converts them into concrete game actions or informational responses.

5.2 Gesture Interaction

A webcam continuously detects the user's hand. OpenCV isolates the region of interest, normalises it, and forwards the image to a lightweight CNN trained on finger-spelled letters. The predicted character is then forwarded to the state manager, which decides whether to accept, reject, or request clarification—always providing on-screen feedback so users understand what was recognised.

5.3 State Manager

The `GameStateManager` class acts as the game's traffic controller. It tracks a finite set of states (initialisation, waiting for input, verifying, end of game) and exposes two public methods—`handle_voice_input()` and `handle_gesture()`. Because every modality funnels through this single point, conflicts are resolved centrally: if a gesture and a voice command arrive at the same instant, the manager applies simple time-stamping to decide which one to honour first. It also filters commands that make sense only in a given phase (e.g., "new game" is accepted only after a round has finished).

5.4 OpenAI Agent

The agent turns raw language into game intents. It maintains short-term context—current word, remaining attempts, last hint delivered—so that a query like "remind me what I've tried" produces a targeted answer. Outputs are JSON objects containing an `action` field (e.g. `"guess_letter"`) and an optional `value`. This structure lets the state manager treat the agent as just another input device, keeping the architecture loosely coupled.

The agent that interprets free-form utterances is created once at start-up. For security, the OpenAI key is injected via the `OPENAI_API_KEY` environment variable (e.g. `export OPENAI_API_KEY=`⟨key⟩ on Unix).

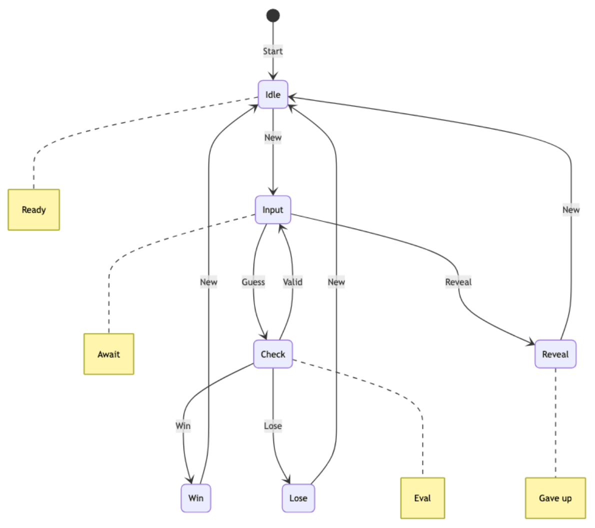

Figure 3. State machine diagram for the `GameStateManager` class, showing the transitions between different game states based on user inputs and actions.

Listing 1. Essential agent scaffold

```
# expects OPENAI_API_KEY to be set in the shell

agent = Agent(
    name="hangman_game_agent",
    instructions="""High-level rules and prompts (
        truncated for brevity)""",
    tools=[
        ("welcome_agent", "Greets and explains
            rules"),
        ("wordsetter_agent", "Chooses the word"),
        ("letter_guesser_agent", "Processes a
            letter guess"),
        ("game_restarter_agent", "Starts a new
            round"),
        ("sync_agent", "Checks active game state")
            ,
    ],
)
```

6 Conclusion and Future Work

6.1 Key Achievements

MultiModalMan re-imagines the classic paper hangman as an inclusive digital game: players can now interact by speaking, gesturing, or typing, while an AI agent orchestrates the state machine, understands natural-language commands, and adapts hints in real time—making the experience accessible to a far wider audience than the original pencil-and-paper version ever reached.

6.2 Future Directions

F1. Expand the word set and add text-to-speech for fully voice-driven, multilingual play.

F2. Boost gesture accuracy with larger, augmented datasets and improved CNNs.

F3. Introduce cooperative and time-trial multiplayer modes.

F4. Provide an interactive onboarding sequence for first-time users.

F5. Store user profiles and match history to track learning progress.

References

[1] AbleGamers Foundation. 2024. *How the Gaming Industry Is Adapting to the Needs of Gamers With Disabilities*. https://ablegamers.org/how-the-gaming-industry-is-adapting/ Accessed 18 Jun 2025.

[2] Muhammad Zeeshan Baig and Manolya Kavakli. 2020. Multimodal Systems: Taxonomy, Methods, and Challenges. *arXiv preprint arXiv:2006.03813* (2020). https://doi.org/10.48550/arXiv.2006.03813

[3] Lizhou Cao, Huadong Zhang, Chao Peng, and Jeffrey T. Hansberger. 2023. Real-Time Multimodal Interaction in Virtual Reality: A Case Study With a Large Virtual Interface. *Multimedia Tools and Applications* 82, 16 (2023), 1–22. https://doi.org/10.1007/s11042-023-14381-6

[4] Meredith Ringel Morris. 2019. AI and Accessibility: A Discussion of Ethical Considerations. *Commun. ACM* (2019). https://www.microsoft.com/en-us/research/wp-content/uploads/2019/08/a14a-ethics-CACM-viewpoint-arxiv.pdf Viewpoint column, in press.

[5] Atieh Taheri, Ziv Weissman, and Misha Sra. 2021. Design and Evaluation of a Hands-Free Video Game Controller for Individuals With Motor Impairments. *Frontiers in Computer Science* 3 (2021), 751455. https://doi.org/10.3389/fcomp.2021.751455