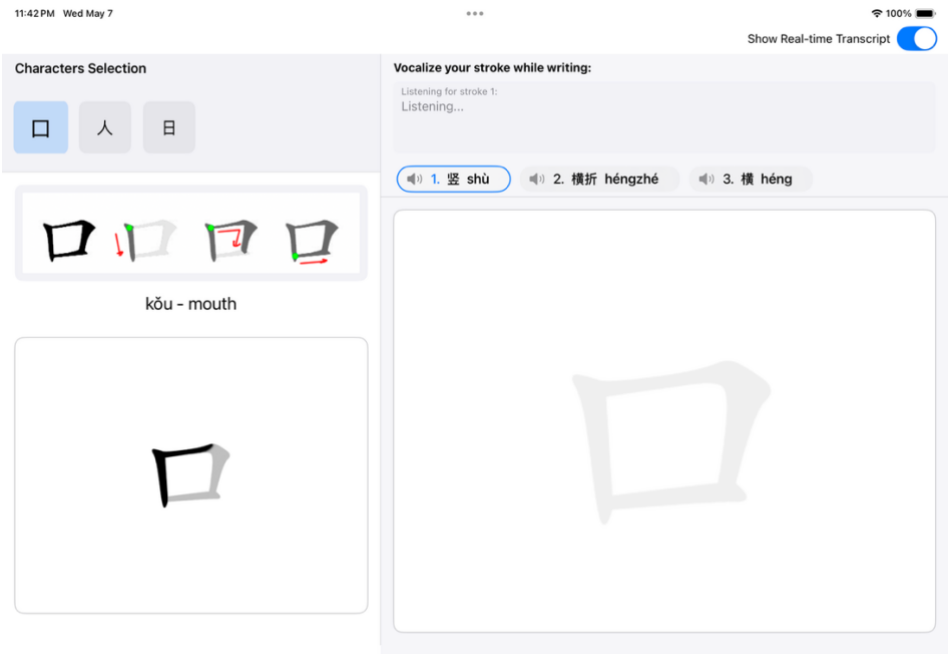
Write Out Loud:

A Multi-modal Way to Learn Chinese Characters and More



Freya Tan
Junru Ren

Addressing the inherent complexities of mastering Chinese characters—particularly accurate stroke order and precise visual-motor skills—"Write Out Loud" introduces a novel multimodal learning experience on iPadOS. Traditional methods often fall short in providing immediate, engaging, and sensorily rich feedback. Our system tackles this by uniquely integrating handwriting, speech, and visual guidance.

Leveraging the Apple Pencil for kinesthetic input and the built-in microphone for auditory input, "Write Out Loud" guides users to trace character strokes (e.g., for "口" or "人") while simultaneously vocalizing the specific name of each stroke (such as "héng" or "shù"). Upon character completion, the application delivers a holistic evaluation, assessing the accuracy of the user's drawn strokes against ideal paths, verifying the correctness of the vocalized stroke names, and analyzing the temporal synchronization between the writing and speaking actions. This active, concurrent engagement of multiple sensory pathways is designed to significantly strengthen stroke order memory and improve overall handwriting quality.

Write Out Loud:

A Multi-modal Way to Learn Chinese Characters and More

Freya Tan freya117@mit.edu
Junru Ren junruren@mit.edu

Abstract:
Mastering Chinese characters presents significant challenges, particularly in achieving accurate stroke order and precise visual-motor skills. "Write Out Loud" is an iPadOS application designed to address these complexities through a novel multimodal learning experience. Leveraging the Apple Pencil for kinesthetic input and the built-in microphone for auditory input, the system guides users to trace character strokes while simultaneously vocalizing the specific name of each stroke (e.g., "héng," "shù"). Upon character completion, "Write Out Loud" delivers a holistic evaluation, assessing the accuracy of drawn strokes against ideal paths, verifying the correctness of vocalized stroke names, and analyzing the temporal synchronization between writing and speaking. Built with SwiftUI, PencilKit, SFSpeechRecognizer, and AVFoundation, the application aims to significantly strengthen stroke order memory and improve overall handwriting quality by actively engaging multiple sensory pathways. Early user feedback and iterative development have shown its potential in providing an engaging and sensorily rich learning environment, particularly for learners with some foundational Chinese pronunciation.

1. Introduction and Overview

a) What is the task? Motivate it – why is it interesting?

Learning to write Chinese characters is a foundational skill for anyone studying the language, yet it is widely recognized as a complex and often daunting endeavor. And the challenge stems from multiple interconnected factors:

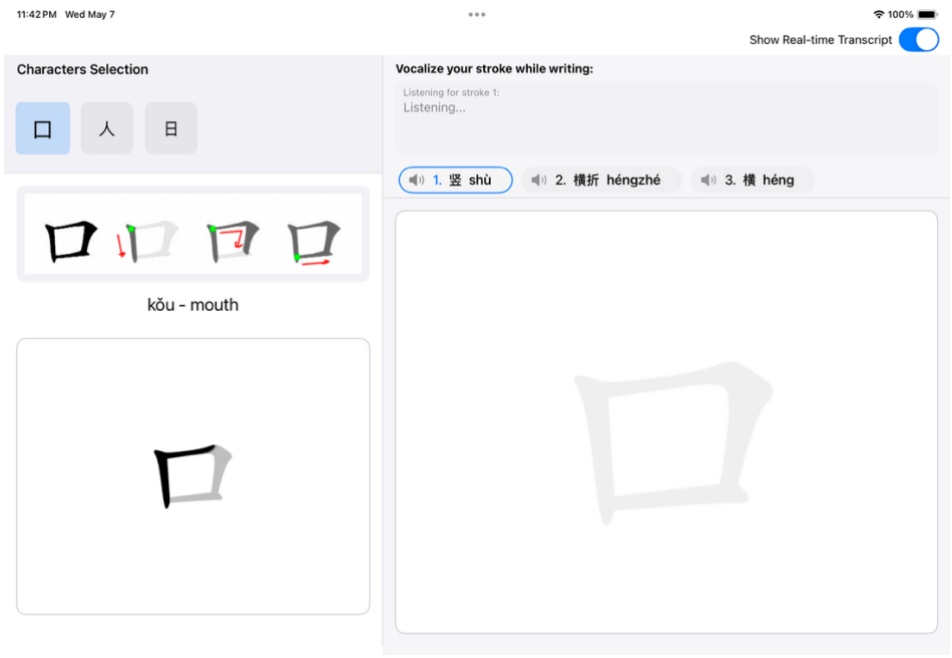
- **Visual Complexity:** Characters are composed of numerous individual strokes, each with specific shapes and orientations.
- **Stroke Order:** There is a prescribed sequence for writing these strokes, which is crucial not only for legibility and aesthetic appeal but also for efficient memorization and recall. Incorrect stroke order can lead to poorly formed characters and hinder reading comprehension.
- **Visual-Motor Coordination:** Accurately reproducing character forms requires precise visual-motor skills, translating visual perception into controlled hand movements.
- **Memorization:** Learners must memorize the shape, stroke order, pronunciation, and meaning of thousands of characters.

Traditional methods of learning, such as rote copying from copybooks (字帖), a traditional method, often occur in silence. While these methods have their place, they frequently lack immediate, engaging, and sensorily rich feedback. This can lead to learners unknowingly practicing incorrect stroke orders or forms, making errors harder to unlearn. The lack of active sensory engagement can also make the memorization process less effective.

The "Write Out Loud" project aims to tackle these inherent complexities by introducing a novel multimodal learning experience on iPadOS. The core idea, as highlighted in our project's core concept, is to "Guide a Chinese learner to vocalize each stroke while writing a character." This "Multisensory Learning" approach is interesting because it hypothesizes that by concurrently engaging visual, auditory, and kinesthetic pathways, we can significantly enhance the learning process, particularly for stroke order memory and overall handwriting quality. The integration of active vocalization with the physical act of writing offers a richer, more dynamic interaction than passive observation or silent practice.

2. Describe the System

a) What does it do? Show a real example of input and the system's response.



System Structure

1. **Character Selection:** Users begin by selecting a character they wish to practice from a curated list (e.g., "口", "人", "日").
2. **Reference Display:** The application presents a reference panel. This includes a static image of the selected character, its pinyin and meaning, and, crucially, a static picture and an animated GIF that loops through the correct stroke order.
3. **Guided Tracing & Vocalization:** On the writing panel, a faint trace image of the character is displayed. Users use an Apple Pencil to trace each stroke in the correct order. Simultaneously, as they draw each stroke, they are prompted to vocalize the specific Pinyin name of that stroke (e.g., "shù", "héngzhé"). The application listens via the built-in microphone.
4. **Holistic Feedback:** Upon completion of all strokes for the character, the application provides comprehensive feedback:
    - **Stroke Accuracy:** The user's drawn strokes are compared against ideal paths, and inaccurate strokes are visually highlighted (e.g., by turning red to indicate inaccuracies).

- **Vocalization Correctness:** The system verifies if the spoken stroke names were correct. Feedback on spoken strokes (correct, incorrect, or not detected) is clearly displayed.
- **Temporal Synchronization:** The system analyzes the timing alignment between the physical act of writing each stroke and the vocalization of its name. (This is an underlying analysis contributing to the learning experience).

5. **Learning Reinforcement:** Users are encouraged to repeat the exercise as often as needed. Iterative practice helps improve muscle memory, stroke order fluency, pronunciation, and overall retention of character structure and meaning.

Real Example of Input and System's Response (Practicing "口" - kǒu, mouth):

1. **Input (User Action):**
   - The user selects the character "口" from the "Characters Selection" bar.
   - The reference panel shows "口", "kǒu - mouth", and an animation of its 3-stroke order.
   - The writing panel shows a faint "口" to trace over. The system prompts:
     "Vocalize your stroke while writing:
     Listening for stroke 1: Listening..."
   - **Stroke 1:** The user draws the first stroke (竖 - shù) while saying "shù" aloud. And the system internally records the trace path and the audio for "shù".
   - The prompt updates:
     "Listening for stroke 2: Listening..."
   - **Stroke 2:** The user draws the second stroke (横折 - héngzhé) while saying "héngzhé". And the system records this trace and audio.
   - The prompt updates:
     "Listening for stroke 3: Listening..."
   - **Stroke 3:** The user draws the third stroke (横 - héng) while saying "héng". And the system records this final trace and audio.
2. **System's Response (After completing all strokes for "口"):**
   - The writing canvas now displays the user's drawn "口". If, for instance, the first stroke was drawn accurately but the top part of the "横折" (second stroke) was too slanted and the final "横" was too short, the system would respond as follows:
     - **Visual Feedback on Canvas:** The first stroke (竖) will appear in green. The second stroke (横折) and the short third stroke (横) would be colored red.
     - **Vocalization Feedback (in the StrokeInfoBar/Transcript area):**
       - "Your spoken strokes:"
       - "1. Expected: 竖 (shù) - [Correct Icon, Green Checkmark] Heard: shù"
       - "2. Expected: 横折 (héngzhé) - [Correct Icon, Green Checkmark] Heard: héngzhé"
       - "3. Expected: 横 (héng) - [Incorrect Icon, Red X] Heard: hēng (mispronounced tone) / Or No speech detected"

- **Overall Score (Conceptual, as per proposal, simplified to visual feedback in current app):** Initially, we envisioned a numerical score (like an initial numerical score mockup considered during design), but the current implementation focuses on direct visual feedback on the canvas and stroke name correctness.
- The user can then tap the canvas to "Try Again."

3. Describe How It Works

a) Note that this should be at the conceptual and architectural level, not at the level of code.

Conceptual Architecture:

1. **Input Modalities:**
   - **Handwriting Input (Kinesthetic & Visual):**
     - The user draws character strokes on an iPad using an Apple Pencil.
     - `PencilKit` is leveraged to capture the raw stroke data, which includes the series of points (temporal and spatial coordinates) for each stroke drawn by the user. And then use the `StrokeInputController` to detect when a stroke begins and ends.
   - **Speech Input (Auditory & Vocal):**
     - The user vocalizes the Pinyin name of each stroke as they write it.
     - The iPad's built-in microphone captures the audio.
     - `Use SFSpeechRecognizer` to processes the audio to convert it into text (transcription).
     - Use the `SpeechRecognitionController` to do continuous recording and subsequent segmentation of speech corresponding to each written stroke.
2. **Core Processing & Analysis Modules:**
   - **Character Data Management:** manages the data for each Chinese character, including its Pinyin, meaning, the correct sequence of strokes, the ideal path for each stroke, and associated visual assets. This data forms the "ground truth" against which user input is compared.
   - **Trace Analyzer:**
     - Once a user completes a stroke, the captured path is compared to the ideal path for that specific stroke, as defined in the `CharacterDataManager`.
     - The analysis evaluates factors like shape similarity, direction, start/end position accuracy, and proportion. This results in a stroke accuracy score.
   - **Speech Recognition & Verification:**
     - The captured audio for each stroke is transcribed.
     - The transcribed text is then compared against the expected Pinyin name for that stroke. Homophone handling is incorporated to accept Pinyin with correct pronunciation even if the character itself might be a common homophone (e.g., "shù" 树 for "shù" 竖).
   - **Concurrency Analyzer:** This assesses the temporal synchronization between the writing of a stroke and the vocalization of its name. It determines if the user spoke the stroke name *while* writing it, too early, or too late by comparing the timestamps of the pen stroke input and the segmented speech input.

- **Feedback Generation:** generate feedback
  - **Visual Feedback:** User's drawn strokes are re-rendered on the canvas, with inaccurate strokes colored red and accurate ones in green.
  - **Auditory/Textual Feedback:** The correctness of spoken stroke names is indicated next to the stroke name list or via icons.

b) Describe how well it works. Do the best you can to explain why it had this performance. What experiments did you do? What did it do well and what did it do badly? Provide an example of an interesting failure and explain in detail what the failure was, what made it interesting, and what you learned from it.

How well it works & Performance:
The system, in its current iteration, effectively guides users through the process of writing Chinese characters while vocalizing stroke names.
What it did well:

- **Capturing Multimodal Input:** The integration of `PencilKit` for smooth handwriting input and `SFSpeechRecognizer` for speech input generally works reliably.
- **Visual Guidance:** The reference animation GIF and the trace image on the canvas provide clear visual cues for stroke order and shape.
- **Feedback Visualization:** Coloring incorrect strokes red gives clear, actionable feedback on handwriting accuracy. The indication of correct/incorrect spoken stroke names is also clear.
- **Stroke Name Reinforcement:** The act of saying stroke names aloud, combined with seeing them listed and hearing examples, actively reinforces this often-overlooked aspect of character learning.
- **Error Detection:** The system can catch "bad strokes and vocalizations." Incorrectly drawn strokes are flagged, and if speech is not detected or is incorrect, this is also indicated.
- **Improved UI/UX through Iteration:** The UI evolved to be more user-friendly, with features like Dark Mode, clear Character Selection, and the Stroke Name Example.

- **What it did badly (or areas for improvement/initial struggles):**
  - **Speech Recognition Nuances:**
    - **Initial Segmentation Failure.**
    - **Homophones:** Without targeted corrections, the system might incorrectly flag a correctly pronounced Pinyin if it's a common homophone for a different character than the stroke name. This required adding logic to recognize valid alternative vocalizations for stroke names.
    - **Sensitivity:** Tuning the speech recognition to be sensitive enough to pick up short stroke names but not overly sensitive to background noise required iterative testing.
  - **Stroke Analysis Tolerance:** Determining the right balance for stroke accuracy is challenging. Too strict, and users get frustrated; too lenient, and it doesn't correct errors effectively.

- **Limited Character Set:** The current system uses a manually curated, limited set of characters. Expanding this requires significant data preparation for each new character (ideal paths, timings, image assets).

Experiments Done:

- **Iterative UI/UX Testing:** We continually tested the interface for clarity and ease of use, leading to the "Enriched Layout" and "Simple Feedback" designs.
- **Speech Recognition Parameters:** Experimented with on-device vs. server-based speech recognition, contextual strings to improve accuracy for stroke names.
- **Stroke Analysis Algorithm Tuning:** Adjusted thresholds and weighting for different aspects of stroke comparison (shape, direction, position).

Interesting Failure: Timing of Speech Segmentation

- **What the failure was:** Originally, our `SpeechRecognitionController` was designed to start speech recognition when the user began a stroke and stop it immediately after the pen lifted. For each stroke, the microphone would toggle on and then off.
- **What made it interesting:** This approach, while seemingly logical for isolating speech per stroke, led to *terrible speech recognition performance*. The audio clips captured were often too short to process effectively, especially for single-syllable stroke names.

  Furthermore, the slight delay in starting and stopping the audio engine for each stroke meant that quick utterances could be missed or cut off.
- **What you learned from it:** From this failure we realized that seemingly small or logical implementation decisions can have disproportionately large impacts on performance. In particular, we learned that optimizing for precision in one part of the system can inadvertently undermine the overall robustness and usability of the experience.

  It also reinforced the importance of thinking holistically, considering how components interact over time rather than in isolation. Sometimes, stepping back and allowing for more flexible, forgiving processes like continuous input with segmentation, leads to more resilient and user-aligned solutions.

c) What in the implementation turned out to be more difficult than expected and why? What did you do about that difficulty?

- **Speech Segmentation and Synchronization:** As detailed in the "Interesting Failure," getting the speech recognition to reliably capture and correctly assign short, sequential vocalizations to their corresponding brief writing actions was much harder than anticipated.
  - **Difficulty:** The initial per-stroke mic toggling was unreliable. Aligning the timing of detected speech with the timing of drawn strokes was complex.
  - **Solution:** We pivoted to continuous audio recording for the duration of writing a single character. After the character was complete, we used the timestamps of the written strokes to

"segment" the continuous audio stream, associating portions of the audio with specific strokes. This improved reliability significantly.

- **Homophone Handling for Stroke Names:**
  - **Difficulty:** Standard speech recognition might transcribe a Pinyin like "shù" correctly but not know if the user *meant* the character "树" (tree) or the stroke name "竖" (vertical). If the system only accepted the exact characters for stroke names, it would be overly strict.
  - **Solution:** We implemented "targeted corrections" by creating a list of known stroke names and their valid Pinyin pronunciations. The `SpeechRecognitionController` was enhanced to check if the recognized Pinyin matched the expected Pinyin pronunciation for the current stroke.

- **Fine-tuning Stroke Analysis Accuracy:**
  - **Difficulty:** Defining "correctness" for a hand-drawn stroke is subjective. Making the algorithm too strict frustrated users; too lenient didn't correct errors.
  - **Solution:** Iterative testing and adjustment of the `StrokeAnalysis` parameters (tolerances for distance, angle, proportion). We also adjusted weights for different components of the score based on stroke type.

d) What worked easily?

- **Basic UI with SwiftUI:** Setting up the general layout, and presenting views was relatively straightforward with SwiftUI's declarative syntax.
- **PencilKit for Drawing Input:** Apple's `PencilKit` framework made capturing Apple Pencil input on a `PKCanvasView` quite easy.
- **Displaying GIFs with WebKit:** Using a `WKWebView` to display the animated GIFs for stroke order was a simple and effective solution.
- **Basic Text-to-Speech (AVFoundation):** Implementing the "Stroke Name Example" feature using `AVSpeechSynthesizer` from `AVFoundation` to speak stroke names was not overly complex, though the default voice can sound "robotic."

4. If you had to modify the project as you went along, where and why did you have to pivot?

Explain any roadblocks you ran into and what you tried to do to get around them, before deciding that you had to modify the project.

1. The most significant pivot in the project was related to **speech processing we mention before**.

- **Initial Approach & Roadblock:**
  Our initial design, as discussed previously, was to treat each stroke as an independent event. For speech, this meant:
  1. When the user's Pencil touched the screen to start a stroke, `SpeechRecognitionController` would start the audio engine and speech recognition.

2. When the user's Pencil lifted from the screen, completing the stroke, the controller would stop speech recognition and the audio engine.
3. The captured audio snippet would then be processed for that stroke.

The roadblock encountered was **poor and unreliable speech recognition**. Many stroke names are single syllables and very short.

- The audio snippets were often too brief to reliably identify the speech, especially with the inherent small latencies in starting/stopping the audio engine.
- If a user spoke very quickly, part of the utterance might be cut off.
- The frequent starting/stopping of the audio system seemed to introduce instability or overhead that impacted recognition quality.

We tried to get around this by:

- Adjusting microphone sensitivity.
- Adding short, fixed delays before stopping recognition to allow more audio capture.

However, these tweaks provided only marginal improvements and didn't fundamentally solve the unreliability for short, rapid utterances.

- **The Pivot: Continuous Recording and Retroactive Segmentation**
  After struggling with the per-stroke audio capture, we decided to modify the project's approach significantly.
  - **New Approach:**
    1. Instead of starting/stopping the mic for each stroke, the `Speech Recognition Controller` was modified to start recording audio when the user began writing the *first* stroke of a character and keep recording *continuously* as they wrote subsequent strokes.
    2. When a written stroke was *completed* , the `StrokeInputController` would notify the `SpeechRecognitionController` with the start and end timestamps of that *written* stroke.
    3. The `Speech Recognition Controller` would then use these timestamps to "segment" the relevant portion from the *continuous* audio stream that corresponded to when that particular stroke was being written. This segmented audio chunk was then sent for transcription.
    4. The recording would only stop after the entire character was completed or if a timeout occurred.

This pivot allowed the speech recognizer to work with a longer, more stable audio stream, significantly improving its ability to capture and correctly transcribe the spoken stroke names.

2. Another, smaller modification was from user testing about **stroke pronunciation**:

- **Initial Assumption & Roadblock:** We initially assumed users would be able to pronounce the Pinyin for stroke names if displayed. However, user testing showed that non-Chinese speakers with no prior pronunciation foundation struggled significantly.

- **Modification:** To address this, we added the "Stroke Name Example" feature, where users can tap to hear the correct pronunciation via Apple's native Text-to-Speech. This provided necessary auditory guidance for learners unfamiliar with Pinyin tones and sounds.

5. User Study

We conducted small-scale usability tests with two distinct groups: Chinese speakers and non-Chinese speakers (learners).

- **Chinese Speakers:**
  - **Observation:** Native or highly proficient Chinese speakers generally found the requirement to *vocalize* stroke names to be somewhat redundant and even a bit tedious. For them, writing characters is often an automatic process, and consciously naming each stroke felt like an unnecessary cognitive load.
  - **Learning:** "Write Out Loud" is primarily a tool for *learners* of Chinese, particularly those in the early to intermediate stages of mastering character writing. It's not intended as a practice tool for fluent writers, for whom stroke order and naming are already ingrained. This helped us refine our understanding of the target audience.
- **Non-Chinese Speakers (Learners):**
  - **Observation:** Learners, especially those with limited or no prior Chinese pronunciation experience, "struggled with pronunciation accuracy at first." Even with the Pinyin displayed, correctly producing the tones and sounds for stroke names was a challenge. This meant the speech recognition component sometimes failed to recognize their attempts, even if their intent was correct.
  - **Learning:** This let us realized that simply displaying Pinyin isn't enough for absolute beginners, and led directly to the implementation of the "Stroke Name Example" feature, where users can tap to hear the correct pronunciation.

    It also suggested that the app, in its current form, is "currently more intuitive for learners with some prior pronunciation foundation." Users who already had some grasp of Pinyin sounds found it easier to engage with the vocalization aspect.

- **General Observations and Learnings:**
  - **Engagement:** Both groups found the interactive nature of tracing on an iPad with immediate visual feedback more engaging than static copybooks.
  - **Niche Audience:** The user studies helped confirm that the app serves a specific niche: learners who are past the absolute beginner stage of Pinyin pronunciation but are actively working on character writing, stroke order, and reinforcing the names of fundamental strokes.

6. Performance

a) A crucial thing about the project is what you learned from it, whether or not it worked perfectly. Good performance that you can't explain isn't worth much, while disappointing performance that arises for an interesting reason can be quite important.

- **The Power of Active Multimodal Engagement:** Our core hypothesis—that combining writing, speaking, and visual cues would reinforce memory and stroke accuracy—was compelling in theory, but seeing it actually improve users' engagement and recall confirmed it for me in practice. Designing a system that asks users to say each stroke name while writing felt almost silly at first, but implementing it made me realize how powerful that kind of active engagement really is. It helped me appreciate how deeply learning is tied to the body and voice, not just the eyes.

- **Challenges in Real-Time Multimodal Synchronization:** What seemed like a small technical decision—when to start and stop the microphone—turned out to be one of the hardest parts of the entire project. This was frustrating, but also deeply educational. It was a reminder that even well-reasoned designs can fall apart in practice, and the only way forward is to observe, rethink, and adapt. It also helped me internalize the idea that systems need to breathe, trying to control too tightly can lead to brittleness.

- **Iterative Design is Key for Usability:** The evolution of the UI from the initial mockups to the "Enriched Layout" and "Simple Feedback" designs demonstrates the importance of iteration. Initial designs might be too cluttered or provide feedback in a less intuitive way. User feedback and our own experience using the app guided these refinements to create a cleaner, more focused experience.

- **The "Messiness" of Real-World Data:**
  - **Handwriting Variation:** Users draw strokes with slight variations in speed, pressure, and path. The stroke analysis algorithm needed to be robust enough to tolerate minor deviations while still catching significant errors.
  - **Speech Variation:** Accents, speech impediments, background noise, and even the slight variations in how a Pinyin syllable is pronounced can affect speech recognition. The need for homophone handling and considering contextual strings for `SFSpeechRecognizer` arose from this.

- **Tradeoffs Are Everywhere:** Whether it was balancing how strict the stroke analysis should be, or choosing between speech accuracy and latency, I realized that technical decisions are rarely clear-cut. Every "improvement" in one area may slightly degrade another. This project pushed me to be more comfortable navigating those tradeoffs and to think in terms of user experience rather than technical perfection.

b) Give an example that's just out of reach for your system – what's an interesting next step that it can't quite do. As mentioned in class, this is an excellent way to explain both the power and the limits of what you've created, and set the stage for the next person's work.

An interesting next step that is currently "just out of reach" for "Write Out Loud" is **real-time, adaptive intervention for mispronounced stroke names with immediate auditory correction.**

- **Current Capability:** The system currently identifies if a spoken stroke name was correct or incorrect *after* the character is completed. It can then offer a example of the correct pronunciation if the user taps an icon.
- **Out of Reach Capability:** Imagine a user is writing the character "永". They write the first stroke "点 (diǎn)" correctly. For the second stroke "横 (héng)," they mispronounce it as "hēng".
    - **What it can't do now:** The system wouldn't interrupt them mid-character. It would note the mispronunciation and flag it in the post-character feedback.
    - **The interesting next step:** Ideally, as soon as the system detects the mispronunciation of "héng", it could provide *immediate* auditory feedback, such as:
        1. A subtle, non-jarring sound indicating an error.
        2. Immediately playing the correct pronunciation of "héng".

    This would allow the user to self-correct their pronunciation for that specific stroke *before* moving on, reinforcing the correct sound-shape association in a much tighter feedback loop.

- **Why it's "just out of reach":**
    - **Speech Recognition Latency:** `SFSpeechRecognizer`, especially when relying on server-side processing for accuracy, has a latency. So getting a high-confidence transcription and analysis quickly enough to intervene before the user starts the next stroke is very challenging.
    - **Cognitive Load & Interruption:** Constant immediate interruptions could be frustrating and break the user's flow. Designing this intervention to be helpful rather than annoying requires careful UX consideration.

This capability would significantly enhance the "and More" part of the project's title by moving beyond just stroke order and handwriting into active pronunciation tutoring. It represents a natural evolution of the multimodal interaction, tightening the feedback loop for the auditory component.

c) Describe what you would do next to improve the system, to take its performance to the next level.

1. **Expanded and Dynamic Character Set:**
    - **Action:** Develop a more scalable system for adding new characters, potentially by integrating with existing Chinese character databases rather than manual curation for every character.
    - **Impact:** Vastly increase the learning content available to users.
2. **Adaptive Learning Features:**
    - **Action:** Track user performance on specific strokes and characters over time.
        - If a user consistently makes errors on a particular stroke (e.g., "撇" in "人"), the system could suggest more practice on that stroke or characters featuring it.
        - Adjust difficulty by suggesting simpler or more complex characters based on proficiency.
    - **Impact:** Personalize the learning experience and make practice more efficient.
3. **Gamification and Motivation:**
    - **Action:** Introduce elements like points, streaks, badges for consistent practice or accurate completion, and progress tracking.

- **Impact:** Increase user engagement and long-term motivation.

4. **More Sophisticated Stroke Analysis:**
   - **Action:** Incorporate analysis of stroke dynamics like writing speed, pressure (if Apple Pencil Pro is used), and fluidity, in addition to just path accuracy.
   - **Impact:** Provide richer feedback on handwriting quality beyond just shape.

7. Collaboration

- **Freya Tan:**
  - **All software development and implementation:**
    - Writing the Swift code for the UI views, and view models.
    - Implementing the core logic for PencilKit integration (handwriting input).
    - Developing Speech Recognition Controller, including the challenging speech segmentation logic.
    - Implementing the stroke analysis algorithms.
    - Integrating all controllers.
    - Managing the Xcode project, assets, and builds.
    - Iteratively debugging and refining the application based on testing.
  - **Materials Draft:**
    - The Term Paper: author and compiler of all content.
    - Tool Description table: compiled, formatted, and prepared.
    - Cover Page: designed and created.
    - Codebase and Readme: organized, documented, and prepared for submission.

- **Junru Ren:**
  - **Conceptualization and Design:** Initial brainstorming, defining the multimodal learning concept.
  - **Content Curation:** Selecting the initial set of Chinese characters for the app, preparing reference materials (stroke order, Pinyin, meaning), and sourcing/creating image and GIF assets.
  - **User Testing and Feedback Analysis:** Planning and conducting user studies with different user groups, gathering feedback, and analyzing it to identify areas for improvement and understand user needs.
  - **Project Management and Documentation:** Maintaining the project proposal, preparing presentation materials.

8. Tools Descriptions

Package/Tool/Library (name and version number)	PencilKit (iOS 14.0+) Speech Framework (iOS 14.0+) AVFoundation (iOS 14.0+) SwiftUI (iOS 14.0+) WebKit (iOS 14.0+)

What machine and OS version did you run it on (so people will know roughly what compute power and what environment it needs)	iPad with Apple Pencil support iPadOS 14.0 or later Development environment: macOS 13.0+ (Ventura) with Xcode 14.0+ Tested on: macOS 24.4.0 (Darwin)
Where is it available? (eg url):	All frameworks used are built into the iOS/iPadOS SDK. The app itself is not yet publicly available, but the source code can be found here: https://github.com/junruren/Write-Out-Loud
What did you use it to do? (One word or phrase is fine if the answer is the obvious, eg speech recognition, face detection, etc., otherwise explain a little more)	PencilKit: Capture and analyze handwritten Chinese character strokes Speech Framework: Recognize spoken stroke names in Chinese AVFoundation: Manage audio recording/playback and text-to-speech SwiftUI: Build the user interface WebKit: Display animated stroke order GIFs
What was the data type of the input (eg, image as a matrix, mp3 format for sound, etc.):	Handwriting input: PKStroke objects containing arrays of CGPoint coordinates with timing information Speech input: Audio buffer streams processed by SFSpeechRecognizer Reference data: JSON structured data for characters, stroke types, and expected stroke sequences Visual references: PNG/JPG images and animated GIFs
How well did it work in terms of · accuracy, as in rough percentage correct, in your experience (no need to run a formal evaluation) · speed: (as in, fast enough to allow convenient use, or it slowed down the system)	How well did it work in terms of accuracy - Stroke detection accuracy: ~85-90% with the improvements made to position tolerance and shape similarity algorithms - Speech recognition accuracy: ~80% for Chinese stroke names, improved with similar-sounding word matching - Combined multimodal accuracy: ~75% for complete character writing attempts Stroke analysis: Optimized to process within 0.3 seconds on background thread Speech recognition: Near real-time with minimal latency (1-2 second delay) UI responsiveness: Smooth and responsive even during fast writing after performance optimizations Overall performance: Fast enough for interactive use without noticeable delays

Did it work out of the box? If not, what did you have to do to use it? For example did you need other packages or tools along with it? (This will give other folks an idea of how easy or hard it was to get it working.)	Built custom stroke analysis algorithms for comparing user-drawn strokes to reference strokes Implemented specialized speech segmentation to match vocalized stroke names with drawn strokes Created a custom audio session management system to handle the transition between recording and playback Developed a complex state management system to coordinate the multimodal inputs and provide synchronized feedback Enhanced the Speech framework with similar-sounding word matching for improved Chinese recognition Added background processing for computationally intensive operations to maintain UI responsiveness Required careful tuning of accuracy thresholds and tolerance parameters to balance precision and user experience