追忆 · In Search of Lost Words
Specifications

Introduction 2
Context 2
Description 2
Gameplay 3
Chinese Characters 3
Magic 3
Numbers 3
Interactions 3
Types of Levels 4
Learning Levels 4
Dungeon Levels 4
Boss Level 4
Life and Game Over 5
Apparatus 6
Graphic Tablet 6
Joycon 6
Scenario and Level Design 7
Storyline 7
Level Designs and Flashbacks 7
Learning Levels 7
Dungeon 8
Boss Level 8
Chapter Sequence 9
Dialog 9
Character Recognition 10
User Model 11
Adaptation 13
Data Collection 15

Introduction

追忆 · In Search of Lost Words is a 3D first-person perspective game that follows the story of a young person on their journey to recover their mother tongue. The character goes through a series of challenges designed by scientists in order to relearn words, and rediscovers their memories along the way.

Context

The aim of the game is to help the player learn to write a set of basic Chinese characters. The primary audience will be people that have never learned to write Chinese. There will be no prerequisites to the game, so it can be used in parallel with a Chinese course or independently.

One of the difficulties in learning the Chinese language is learning the writing system, as there is no alphabet or reduced system that can be learned in order to write full words. In Chinese, each word is represented by a different character (logogram) or the combination of two or more characters. Most characters hold a meaning of their own, and most basic concepts can be represented with one character. As such, our aim is to provide a fun way to learn to write or practice some of these characters.

Description

The game is made up of chapters containing learning levels and practice dungeons, and an infinity mode. Learning levels and practice dungeons are interlaced so that the player has a chance to practice the new characters they have learned before learning new ones. The practice dungeons will contain puzzles and pathways where the player will be incited to use certain characters in order to interact with the environment without any pressure, and enemy rooms where the player will have to react quickly in order to fight various types of enemies by activating the appropriate type of magic using the right Chinese characters. Infinity mode will be unlocked once the player is ready, and will provide additional adapted practice to players who feel the need to review their skills, or those that want to challenge themselves for fun. In particular, it will be a way for players that have finished the main story to continue practicing.

The game will use a combination of machine learning techniques in order to analyze the characters written by the player as well as to provide adapted practice opportunities. The use of characters in the appropriate context, the use of hints and the correctness of the handwriting will be used to determine the mastery of the player. The player will be encouraged to practice their weaker skills through the generation of elements (objects or enemies) in the environment that require them to use those skills.

Gameplay

The player has 10 characters to learn and cast on the environment. The characters will be obtained as the levels progress.

Chinese Characters

There are two types of characters:

- Magic: 火 水 风 卫 弓
- Numbers: 一 二 三 四 五 (1-5)

Magic

Each type of Magic has its own functionality for puzzle solving as well as enemy elimination:

Magic	Meaning	In Puzzles	In Fight	Gesture to activate
火	Fire	Light up the environment Burn wood Melt ice	Kill wood enemies	Throw a ball
水	Water	Put out fire	Kill fire enemies	Throw a ball
风	Wind	Put out fire Dispatch light objects	Blow away dandelion enemies	Blow winds with a fan
弓	Bow	Shoot objects from above Carry fire to farther places	Shoot target enemies	Draw a bow
卫	Defense	Pass through spiky floor Protect	Defend against energy waves	Arms in front, as if guarding with a shield

Numbers

The numbers serve in the puzzles where the player will need to input the right number in simple math problems. In battle, the player can also cast a number along with 火水弓, to attack groups of enemies which would only be killed when the number and the magic are both correct.

Interactions

In order to play the game, the player would need to hold the stylus in their writing hand and the joycon on the other.

While moving, the player can use the joystick of the joycon to move around and use the tablet to look around (like a mouse).

To use magic:

1. The player needs to hold onto a button, and a canvas for writing Chinese characters will appear on the screen and time slows down.
2. If the player is not sure of the character, they can click on their grimoire containing all the characters they have learned thus far in order to see gifs of how to write each of them along with their meanings.
3. The player writes the character and the recognizer validates or invalidates the writing If the character is recognized, the magic appears on the player's other hand.
4. If there is a follow-up character (ex. 二火 or 火弓), the player can keep writing while holding down the canvas button.
5. Once the player has finished writing, they release the button and the game continues.
6. The player can then click a button and execute the corresponding gesture in order to cast magic.

(See which button to press in Apparatus)

Types of Levels

There are three types of levels in the game:

- Learning levels: Introduce characters for the first time
- Dungeon levels: Explore gameplay and train
- Boss level: The last level

Learning Levels

The player learns a character and practices in the environment. At the end, a flashback is unlocked recounting a memory linked to the word.

Dungeon Levels

A dungeon consist of

- Lobbies: A big space where the player needs to solve puzzles in order to open the doors to the rooms, once all the rooms are open, the hallway to another lobby would open.
- Rooms: The rooms contain enemies generated at the moment the player opens the door, they would be adapted to let the player write the character they're the least familiar with.
- Hallways: The hallways link lobbies, which consist of obstacles which can be cleared with characters. Same as the rooms, the obstacles are generated adapting to the player skills, they provide a more laid back training environment.

Each dungeon would preferably have a theme, and a flashback that's based on the theme (ex. Throwing water balls everywhere - Flashback to playing water fights).

Boss Level

The final level of the game, which consists only waves of enemies approaching. The player would need to use the right magic and number to attack in order to survive until the end. The end condition is defined by whether the player has reached a certain percentage of competence in all characters. An exception will be made for the character 一 (one) because casting magic for one enemy doesn't require the use of 一 in front, and this character can be assumed to be mastered if the player masters the characters for two (二) and three (三).

Life and Game Over

All the attacks the player casts on enemies result in instant kills. The player has a fixed amount of lives that can be lost if attacked. When all lives are lost, the game prompts the player to go back to the main menu or restart the level. The player can also restart the level or reload the dungeon lobby whenever they want in the in-game menu.

Apparatus

The devices that we intend to use, other than a computer with keyboard and mouse, are: a Wacom graphic tablet and a Nintendo Switch joycon.

Graphic Tablet

In order to trace the handwriting of the player precisely, a type of interaction close to writing with a pen on paper is envisioned. We consider the graphic tablet to be the best option as it is used by artists for drawing with precision. The input of the tablet would be recognized as mouse input by Unity, which is enough as we only need the positions of the stokes

The input on the graphic tablet can be used for:

- Movement: Change viewing angles
- Handwriting

Joycon

The joycon has several types of inputs such as buttons, a joystick, an internal gyroscope and an accelerometer. It can also give haptic feedback using the HD rumble motor.

For the moment, we intend to use the following inputs and outputs:

- Joycon buttons: UI navigation, open canvas
    - ZL/ZR: open canvas
    - X/Y: open grimoire of gifs
    - L/R: start executing a gesture
    - A, B: confirm, cancel, ect.
    - +/-: open menu
- Joystick: UI navigation, movement in environment
- Gyroscope & Accelerometer: activating shield, drawing bow, shoot fire/water, etc.
- HD rumble: Tell the player if the system recognizes a character, show tension when drawing a bow, vibrate when attacked by an enemy, etc.

(If there isn't enough time to implement, the gestures would be simplified as a press of a button)

Scenario and Level Design

Storyline

The player takes on the role of a main character who is a native Chinese speaker. This character got into an accident and lost the ability to speak their native language as well as their childhood memories. The game happens in their brain during an innovative treatment in a laboratory with a neural scientist. The neural scientist thinks that, by relearning the language, the protagonist would also be able to recover their lost childhood memories, as language plays a big part of how we learn to see the world.

The scientist builds an interactive environment in the brain of the protagonist for them to explore and use characters to solve puzzles and defeat enemies. Everything goes well until in the end, when the system is hacked, and the enemies increase exponentially, threatening the learning progress of the patient. The protagonist manages to defend themselves and the game ends. The story will continue in the next game.

(Disclaimer: No fact checking has been done, and there is no scientific proof of the theories and technology claimed in the scenario)

Level Designs and Flashbacks

After competing, some of the levels, a flashback would come to a player, telling the lost memory from his childhood. The flashbacks are texts printed on a black screen with a simple typewriter effect. Some flashbacks ask the player to write a character to interact and continue the story (ex. Lit dad's beard on fire, draw 水 to put out the fire)

Learning Levels

Here are some examples of the levels and flashback for the characters:

火 (fire): You're in a cave, everything's dark, you learn 火, and you light up the cave.
Flashback: "There was a blackout, dad lit up candles, I felt safe and warm."

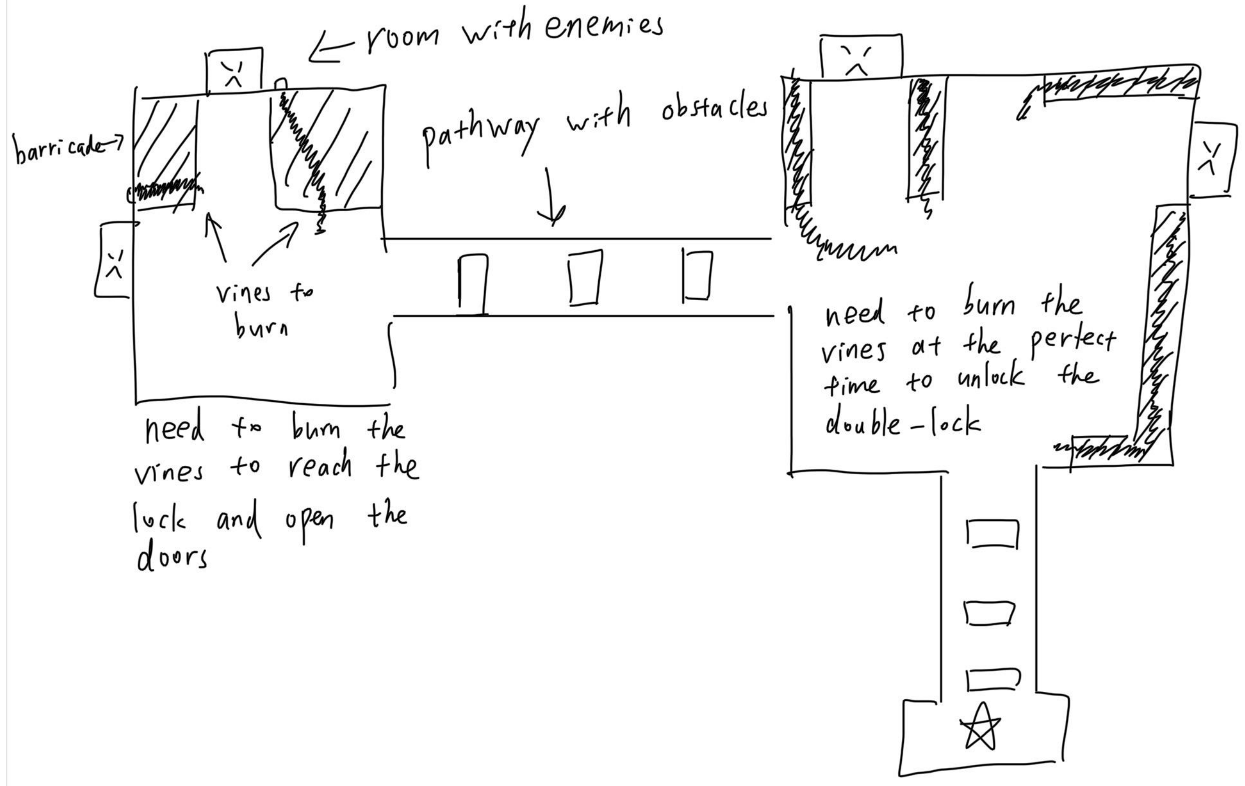
水 (water): There is a tree on fire, you learn 水, and put out the fire.
Flashback: "I lit dad's beard on fire, mom splashed her cup on him, I've never seen him being mad while laughing so hard. I was grounded for a month, but it was worth it."

风 (wind): There are lots of leaves on the ground, you learn 风 and manage to blow the leaves all around you.
Flashback: It was my birthday, Mom lit up the candles, they sang me birthday songs, I made a wish, and I blew the candles."

Dungeon

The dungeons consist of auto-generated obstacles in hallways and auto-generated enemies in rooms, as well as handcrafted puzzles in the lobby. The puzzles aim to combine the characters together in interesting ways in order for the player to memorize better the writing and the meaning.

Here is a possible level for the dungeons:


room with enemies
barricade
pathway with obstacles
vines to burn
need to burn the vines to reach the lock and open the doors
need to burn the vines at the perfect time to unlock the double-lock

An example of the playing order would be:

1. Draw 火 to burn vines and open doors
2. Kill enemies in the rooms
3. Go through the pathway
4. Draw 火 and burn the vines with the correct timing
5. Kill enemies in the rooms
6. Go through the pathway
7. Reach the end and possibly get a flashback

Boss Level

The boss level is the final level where everything is auto-generated

Upon completing the boss level, an infinite mode will be unlocked in the menu for players to train further.

Chapter Sequence

The progression of the levels would be a mixture of learning some characters and then some levels to practice, until reaching the final level. The game is separated into 4 parts:

Chapter 1: Learn 火, 水 -> Dungeon practice
Chapter 2: Learn 风, 弓 -> Dungeon practice
Chapter 3: Learn 卫 and numbers -> Dungeon practice
Epilogue: Boss level (Infinite Mode unlocked at the end)

Dialog

Here are some examples of the dialog the scientist can have with the player:

Transmitting message:
C-c-can you hear me?
Listen, you've been in a car crash, and it seems that it has damaged the part of your brain that processes your maternal language and your childhood memories. We've been trying a new experimental method that puts your brain in a simulation which will help you recover the damage. We need you to use the knowledge you've picked up so far and use it.

Transmitting message:
Good! You're doing very well, I think with this speed, we can...
Wait what? How is this possible?
OK, hold on, we're detecting some interference in our system...
How can it...
Can you hear me? It seems someone has been attacking our system, they changed the level difficulty and danger to maximum. If you lose here, we don't know if you'll come back from your coma. Someone really doesn't want you to recover. OK, I've unlocked all the characters for you, I'll help you however I can, but it's up to you to defend yourself!

Voice-over:
You've defended yourself and recovered some of the characters and memories. How much longer will it take you to recover your maternal language skills? Who was the person that tried to sabotage you? Guess you'll find out soon.

Character Recognition

The handwriting of the player will be inputted through the use of a graphic tablet. Unity will treat the input as it does input from a mouse. This is suitable for our use as we will only consider the coordinates of the strokes.

In order to identify the characters written by the players, as well as the quality of the handwriting (order that the strokes were written in and distance from a standard), we will need to have two recognition systems.

Before inputting the handwriting into the recognition systems, we will first have to normalize the data so that it is comparable. This will consist of sampling a fixed number of points from the written character and adjusting the coordinates such that the considered characters are always approximately the same size.

The first system will be a supervised learning model that is trained on a database of all the characters written by different people of different profiles (beginner, advanced, proficient) several times. If we find an external API that is of good enough quality, we may consider switching to that option in order to focus on other parts of the game.

The second system will be a self-programmed stroke recognizer that will be loosely based on the One Dollar Recognizer that we learned and implemented in our Human-Computer Interaction labs last year. It will work based on the number of strokes used and a comparison of the relative position of different strokes. For example, the character 弓 may be written by a beginner in one stroke, but it is actually composed of three.

User Model

In order to best support the player in their learning, the game will adapt the enemies and objects in the environment to help them practice their weaker skills.

To achieve this, we will need to recognise situations in which the player confuses words or doesn't write a word correctly and update the user model accordingly. There will be three levels of mastery checking :

- The first level consists of checking whether the character the player wrote is recognisable by the system. If the character isn't recognised, we will give the player a chance to consult their grimoire of magic in order to review the character that they intended to write.
- The second level consists of checking whether the player wrote the appropriate character in response to the context. For example, if the player uses the character for fire when faced with an enemy that is only susceptible to water attacks, we will use that as an indicator that the player may have forgotten the right character, or confused the two.
- The third level consists of checking whether the player wrote the character with the correct stroke order, as well as whether the handwriting was far from a defined standard. These will be used as weaker indicators that the player still needs to practice to master the character.

The traces that will be necessary for the above are as follows:

Level 1: Recognition of the character	● If the system fails to recognise a character, we will not need to trace anything specifically as the system will trace the player's use of help. ● If the system recognises the character but it is far from the standard (either a low confidence in the prediction or using the mathematical distance of the points to the standard writing), we will trace that.
Level 2: Correct character choice	● In order to trace this, we need to trace the following pair: (intended character to be practiced, actual recognised character). The character that is meant to be practiced will be deduced from the action that the system has sequenced (for example, if the system spawns an enemy that is susceptible to fire).
Level 3: Correct character handwriting	● The trace for this will be a boolean indicating whether the strokes of the character were written in the right order.

Other	● In addition to the above, we will trace the player's consultation of their grimoire. When they open the grimoire to view a character, we will trace this action and pair it with the character that the player writes afterwards. We will consider this an indication that they have not yet mastered this character and will update the user model accordingly.

We will use Bayesian Knowledge Tracing (BKT) as our user model. Each skill will correspond to a character, and our estimation of the player's mastery of each skill will be a value between 0 and 1. We may separate the mastery of meaning (using the right character in the right situation without considering the order of strokes and quality of writing) and mastery of handwriting if we observe it to be interesting or necessary, but for the moment we intend to only use one skill for each character.

We consider that the skills are independent from each other, because although certain characters may be in part composed of other more basic ones, it is helpful but not necessary to know them in order to be able to learn the composite character. The mastery of meaning and writing of a composite character that contains a more basic character also does not guarantee that the user will realize that the composite character contains a basic one, or know its meaning.

The user model will be updated each time the player uses the canvas to write a character. If we can, we will also implement an update to represent the probability that a player may have forgotten a character over time. The amount by which a value is increased or decreased will have to be decided according to our judgment and experience after testing, as we will not have enough data to learn these values.

Adaptation

In the practice dungeons and in infinity mode, the game will spawn different objects and enemies according to the player's weaknesses.

We will first need to create a way to link these objects and enemies to the characters that they make the player practice.

In order to sequence these spawns, we are considering several options:

- The first option would consist of a rule-based system where we draw random characters to practice that are below a certain level of mastery, and alternate that with characters that the player masters well in order not to forget them and also to give the player a confidence boost.
- The second option would be similar, but draw characters proportionally to their level of mastery, with an applied factor that we will have to identify so that the player will not be too frustrated.
- The third option would be to use a Deep Reinforcement Learning algorithm to identify the best objects/enemies to spawn in order to increase the player's mastery. For this, we will have to design an appropriate reward function that will be based on the increase of mastery. The algorithm is likely to constantly recommend characters where the player is weakest and thus increase frustration. If we find that this is the case, we can define a sequence whereby we draw characters using the RL algorithm interspersed with characters that we know the player masters.

We will choose one of the above according to the time we have at our disposal as well as results we may obtain. For example, if we cannot get satisfactory results with the Deep RL algorithm because of poor/insufficient training data we may switch to another option.

For the Reinforcement Learning algorithm, our situation corresponds to a Partially Observable Markov Decision Process (POMDP), as we have no way of accurately obtaining the knowledge state of the player. As such, we will estimate this state using BKT. Thus, the observations will consist of a vector indicating the estimated mastery level of the player for each skill. The actions will correspond to the various objects and enemies that the system can spawn.

One foreseen difficulty with the algorithm will be that the action space will have to be adapted according to the unlocking of characters after each chapter. To overcome this naively if we do not find a better way, we may redraw a character ourselves if the algorithm recommends a character that has not yet been introduced.

In addition to adapting the sequence of characters practiced, we will also adapt the difficulty in two ways:

- The main way will be by adapting the help the player has while writing. When the player’s mastery is low, we will show a gif of how the character should be written (with stroke order) while the player is writing the character. When the player’s mastery is intermediate, we can remove the gif and only display an image of the character (the player will have to remember the stroke order on their own). The last level of difficulty is when no more help will be provided, the player will have to remember the character on their own. They will be able to choose to look at a hint if they like. It will give us an indication that they haven’t mastered the character yet and we will update the user model accordingly, which will in turn automatically lower the difficulty.
- The second way we may adapt the difficulty is by giving the player more or less time to write the character before an enemy hits them and they lose a life. This can be a mechanic that is added only once the player has fully mastered a character.

Data Collection

We will need to collect data or draw from expert knowledge for several parts of the program:

- to recognize handwritten characters,
- to initialize BKT,
- to train the RL algorithm if it is implemented.

For the recognition of characters, we intend to collect samples from people who have never learned to write Chinese, both before they are shown the stroke order and after. We will also collect samples from people who are currently learning Chinese, and experts (native speakers). We think we can collect 5 samples of the first category, and 5 samples from experts. It will be more difficult to find Chinese learners from our entourage so we will aim for 2.

For the initialization of the BKT, we intend to assign learning and forgetting probabilities based on our own judgment. We will initialize the priors either by collecting data from different categories of people with a version of the game that does not use BKT, or by allowing the player to select on a home menu the characters that they know. This would provide the basis for the initial probability of mastery of those characters, the others would be initialized with 0. The latter would be preferred as it will personalize the initialization of the estimated values. If we cannot achieve this either, we will initialize all probabilities to a low value (for example 0.2) after the player has done the chapter where the character is taught, and learn as the player practices.

Finally, we will need data to train the Deep RL algorithm on. This will be very challenging as it may require the learning of hyperparameters as well as weights for the neural network. It is difficult to estimate the amount of samples we will need for the algorithm to perform in a satisfactory manner, and we will look into the latest algorithms that are sample-efficient, but we have a fallback solution as described in the previous section if this fails. It may not be very problematic in itself if the algorithm doesn't perform in an optimal way, as the game should be interesting for the player even if the character chosen to be practiced is totally random, but it would defeat our objective of the game being adaptive.