A Card-based Detective Game Editor

A project proposal for the MS in Interactive Media & Game Development at WPI

By Qihuan Aixinjueluo
Advisor: Jennifer deWinter
Readers: Dean O'Donnell, Jennifer Rudolph, Wenhua Du

Abstract

Games are an expression of the cultural context form which they are made. For this project, I was interested in developing a Chinese detective game genre, which focused on a scholar interviewing and judging from a place of interviewing. In order to create such games, I made a "Card Detective Editor," a system that supports game designers and developers in designing their own detective games which an emphasis on interviewing and formula creation. This report provides an overview of this editor and early user testing of the system. Data from the testing indicates that the tutorial structures and editor provide the necessary support for game designers interested in Chinese detective games to design their own dynamic narratives.

Introduction

In Chinese famous detective stories like *Di Renjie* [1] and *Jutice Bao* [2], the protagonists are government officers. They functioned as judges by interrogating related people and getting secondhand information from assistant officers. Tiejun Lin also claimed that in ancient China, judicial inspections were usually committed by secretariats or slaves and reported in detail to the officers in charge [3]. That provided me the insight that it would be a different experience for players if they can only read and organize information to solve crime cases. This is different from Western detective narratives, which ask detectives to go to different locations and people, interview, and sometimes fight.

My goal was to create an editor and game catalog that focused on this culturally-centered Chinese detective narrative structure. Thus, for this project, I designed and developed an editor that is suitable for making Chinese-content, card-based detective game. My design goal for the editor is allowing developers both with and without programming experience to implement their story into a card-based detective card game especially suitable for Chinese narrative content. In the games, players need to organize and cards in a logical way to solve crime cases. A short sample game was made and implemented in the tutorial of the game editor, helping developers understand the usage of the game editor by guiding them to build the sample game step-by-step.

Background

There are many detective games on the market. Take famous *Ace Attorney* [4] series as an example, which marries detective research with courthouse drama from a Japanese perspective. In detective games, players are often asked to collect information, solve cases, and show the process of their reasoning.

Different detective games apply different mechanics to let players show their thoughts. A straightforward game mechanic to do this is using a sequence of multiple choices. But there are also some unintended consequences of this: the questions and choices may give out the answers; players

can also get the answer by exhausting all provided choices.

These drawbacks prevent players from thinking answers on themselves. A better practice is used in game *Detective Grimoire* [5], in which players need to select correct words and put them in a right sentence to show their reasoning. I decided to adapt this mechanic to design my game editor to include a way to organize cards. I modeled *Detective Grimoire* and built my formula system. Players need to select the right formulas and put right cards in it to complete a sentence and show their thoughts on the case.

In addition to computer games, the market also has a number of detective card game, which use cards to play the primary role in the narrative and puzzle. So far, most detective games on the market are board games such as Sherlock Holmes: The Card Game [6] and Detective: A Modern Crime Board Game [7]. In these board games, part of the information is given to the players in the form of playing cards. I decided to use cards as the basic units of my game too because cards have moderate amount of information on them can be arranged easily so that they work well with the experience of organizing information.

Zhai [8] considered favorability as indirect resource that can be exchanged and a method of power reproduction in Chinese context. So, I think favorability is a suitable Chinese element that I can integrate in my game design. Based on that, I built my favorability system, making players try to balance their relationship with multiple non-player character organizations. The card game *Reigns* [9] gives me a lot of insight in this part. *Reigns* is a card-based narrative game with a very good slice of the experience of balance. *Reigns* asking players to act as an emperor balancing different attributes of a country by making decisions.

There are some game engines on the market focusing on narrative games. Some of them requires coding skills from developers like *BK engine* [10]. *Twine* [11] is a good example of game engines made for non-programmers. I also designed my editor to not require programming skills from developers by using condition cards to keep track of all the conditions. However, neither of these engines cleanly support and promote a Chinese narrative structure of collecting information and inferring conclusions based on complex relationships. What became apparent in this project was that I would need to build my own editor, because unsurprisingly, engines reflect the cultural assumptions and ideologies from which they are built.

Twine also allows developers to arrange passages around the screen according to their needs without altering the underlying logic [12]. This idea meets my need in the game session of allowing players to organize information. In playing session, I also allowed players to drag and drop cards as they need. So that players can put cards they think relative together or arrange cards according to their logic. Emily Short also pointed out that stories written in Twine had their unique structure [13]. That indicated that my editor also requires stories in unique structure to work. Mia and Dan discussed the difference between a more specific game engine with general game engine like *Unity*. They claimed that a more specific game engine sacrifices flexibility for more rapid developing process [14]. That can be applied to my editor too. My editor has the limitation of can only being used to make a very specific kind of game.

On helping developers with understanding the logic of an unfamiliar

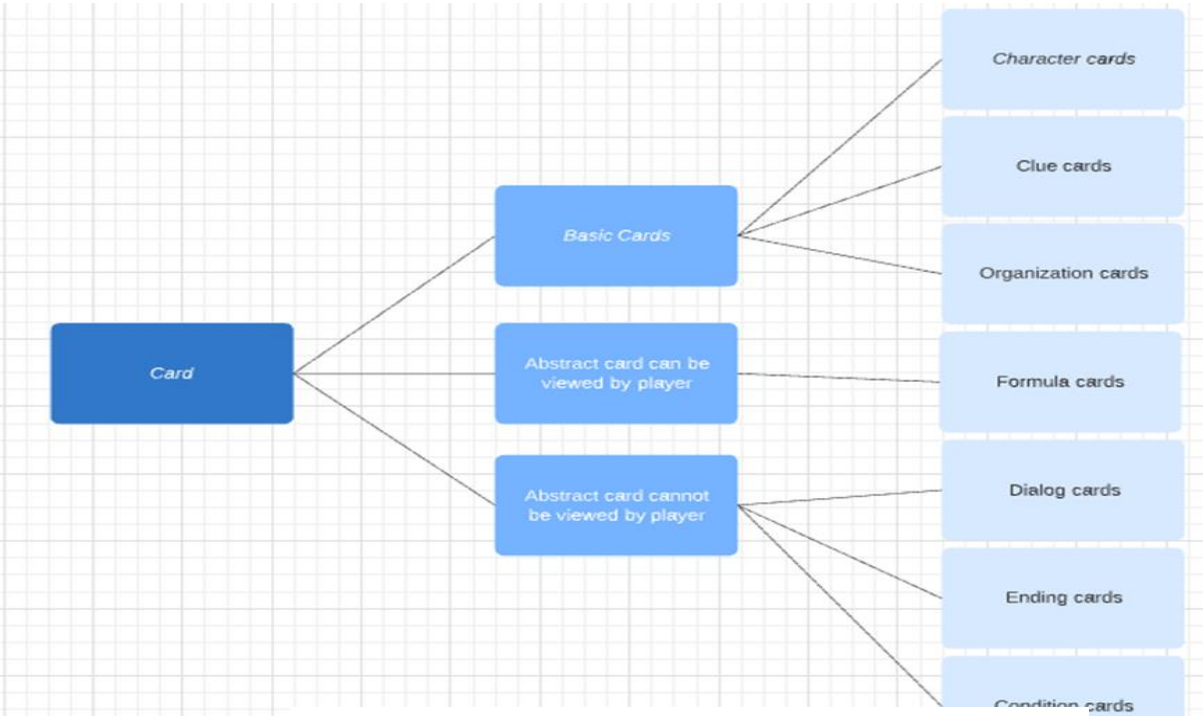

Figure 1: Structure of Card Types

game engine, Chover, Marin, Rebollo, and Remolar [15] has done an ideal model work for me. They have developed a simplified 2D game engine. In their work, they also used a visualized behavior tree to avoid coding requirements of their developers. Another insight in their work that they try to make their algorithm very consistent. They realize all their logic by controlling actors. When they need to test whether there are 3 Actor 1 in a row, instead of using loop, they try to ask their developers to create a new actor and move along all space to find whether there are 3 Actor 1 in a row. This practice greatly reduces the learning time of the developers by using the same concepts for as much mechanics as possible in the engine and avoiding introducing new concepts. In my editor design, I also tried to make some supportive functions such as dialogs and conditions in the form of card to keep the consistency.

Engine overview

The editor organizes information in the form of cards (see Figure 1 for the logic of the card types). The developers' jobs are to build a deck of cards with relationship to each other and put all information on cards. Based on their functions, I divided them into 7 different types: character, organization, clue, condition, formula, dialog and ending (see Table 1 for a full explanation of each). For example, characters related to the case will be shown as character cards which will includes name and description. Likely clue cards are clues and inferences related to games. Also, some more abstract concepts will also be in the form of cards like dialogs and endings. All of the cards should be made by developers to form a game. I will cover the abstract cards in more details in later sessions. To provide contrate examples when I explain the specific components of the game editor, I will briefly introduce my sample game *Find the robber.*

Table 1: Card Types/Functions and Developer Abilities

Character card
- Name and description about an in-game character
- Developers specify character name, organizations, description, card image and condition card.

Clue card
- Name and description about a clue related to the case
- Developers specify clue name, description, card image, and condition card.

Organization card
- Name, description, members of an organization related to the case.
- Developers specify organization name, description, card image, and condition card

Condition card
- A functional card used as intermediary between player behaviors and new cards
- Developers specify condition elements and the change of organization favorability.

Formula card
- An interactable card with an incomplete sentence, used for players to arrange their card based on logic
- Developers specify condition card, text prompts, amount and type of card it asks for.

Dialog card
- A functional card includes the information of a dialog. Triggers dialog box in game instead of given to players as a card
- Developers specify condition card, content, character, and whether there is a question asking for a card.

Ending card
- A functional card includes the information of a possible ending. Triggers an ending scene in game instead of given to players as a card. Developers specify condition card, ending text.

Example Game: Find the Robber

Find the robber is a brief sample game I made as an evidence of the working editor and built in my editor tutorial. I will use it as an example to talk about what the game is like and how the developers should do to build it.

The sample game is a short story about an old man was robbed on the road and he did not know who the robber is because he was knocked down from behind. The robber took away the old man's package and a passerby ran after and caught the robber. However, when people came to them, a young man and a woodcutter accused each other to be the robber. Players will play the role of a government officer in charge of the case and need to use observation and interrogation to find the real robber.

The right solution is when player ask both suspects, they will both claim they are innocent, and they catch the other. Player will also find the young man to be very exhausted. It is not likely that the young man with worse stamina can catch the woodcutter. So, the player can tell the young man is the robber. It might not be a good detective story with perfect logic, but we

can explore most of the mechanics of the editor with this example.

Potential conditions

There are four kinds of conditions that players can potentially trigger with their actions. The conditions will be checked and taken care of by the game editor. Every time players do something that might trigger conditions, the game will check whether any condition preset by the developers is met. If so, the game editor will trigger next steps. I will explain the four correspond actions that might trigger conditions one by one in this section and explain how to preset conditions in the follow section of condition cards and elements.

° Using Formulas

Using formulas means players put cards in their hand in some certain incomplete sentences to make them meaningful to show their reasoning. The incomplete sentences are called formulas. For instance, one of the formulas in the sample game is called “ask for detail” which writes “If I want to know more about the case, I should ask”. It is an incomplete sentence with a blank (I call it slot in the editor), asking player to fill a character card to complete it. If the player put “young man” character card in it, it becomes a meaningful sentence means player’s character ask young man for more information. To allow players to use a formula in game, developers should build a formula card with the prompt and slot on it and cards that can be potentially put in the formula. Figure 2 is the user interface (UI) that will be used to build a formula card.

The formula system is built because when using formula system, questions might not be necessary for the game progression to promote. Questions themselves might give

Figure 2: Build formula card interface

out the answers. Choosing correct formula also provide players with more challenges, requiring them to really understand what happened in the cases. Also, formula system is the mechanic I designed for developers to create an ancient Chinese detective experience of organizing information.

° Finishing Dialogs

Finishing dialogs means the game shows players a piece of dialog and players finish reading them. Some dialogs are with questions and ask players for some certain cards. When the sample game approaches the end, the old man will ask players who is the real robber. And players are supposed to show him the character card of the young man as the answer. To make it possible for players to finish a dialog, a dialog card is needed (a character card and an optional condition card needed for a dialog card may also be needed). All dialogs in game are also cards made by developers. Instead of being given to players’ hands, dialog cards will trigger corresponding dialogs on the bottom of the screen. Developers should specify who the dialog is with, what the content is, and whether there is a card required from

players. Figure 3 is the UI that will be used to build dialog cards. Dialogs with questions

Figure 3: Build dialog card interface

are necessary when developers are trying to build a detective game with longer process. Developers might want check players' progression at some points of the games. Also, integration is also an important part of the cases solving in ancient China [3], having dialog system can recreate the experience.

- Passing time

Developers of detective games might want see players try solving the cases instead of trying out all options. Passing time is a system that players used to prevent players from overmuch trying. Every time players try to fulfill a formula, no matter they do it correct or wrong, time will past by one. In the sample game, players will have five units of time to solve the case. Time pass will be taken care of by the game editor automatically.

- Changing favorability

Favorability is the value I used to strengthen the Chinese context. Developers are expected to create incentives for players to maintain good favorability level with all organizations. Changing favorability is a little more complex, it stands for an organization's favorability towards players' character is changed. Players might trigger favorability increase or decrease as a result of other actions. It is both a result of behaviors and a behavior itself. I will cover it in the following section on conditions.

Conditions cards and condition elements

Condition cards and condition elements are the tools I provided developers to check players' behaviors and give feedbacks. Behaviors themselves are not enough for a game loop. Developers still need some way to tell the game editor what behaviors they are expecting and what reactions should the game gives players as feedbacks.

To achieve this goal, I introduced the concept of condition cards and condition elements. Condition cards are abstract cards that only help developers to set up the interaction of cards and cannot be viewed by players in their playing session.

In *Twine*, conditions are set by some text-based scripts. And players interact with the conditions by picking provided choices. The condition card method I am introducing here can set conditions by making and choosing cards which make the process more consistency with other parts of the game development.

Developers can specify a condition card for every card. When a card is specified a condition card, it will not be given to the player until the condition card is triggered. Every possible player behavior can trigger correspond condition element. Take the sample game as an example: when players ask young man for more information. A clue card called "young man's statement" will be given to players. So, the condition element here is use young man character card to complete the formula "ask for more detail". Any condition card can record up to 3 condition elements, the relationship among

which can be defined by developers as “alternative” or “necessary”. “Alternative” means the condition card will be triggered if any of the condition element is met (as shown in Figure 4), while “necessary”

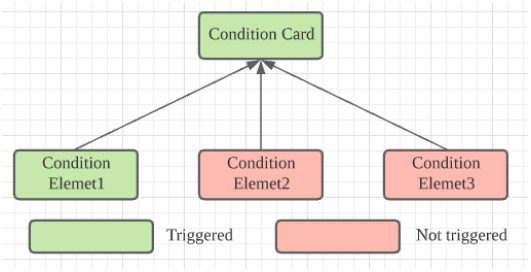

Figure 4: Alternative structure

means the condition card can only be triggered when all the condition elements are triggered (as shown in Figure 5).

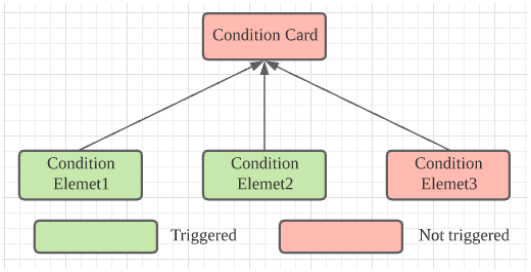

Figure 5: Necessary structure

If developers want to give the card “young man’s statement” to players once players asked young man, developers should make a condition card called “ask young man for detail” with only the above condition element on it. As a result, when players filled the “ask for detail” formula with the character card young man, the condition card will be triggered. Then the developers can make the card they want to reward players for their correct behavior, in this case, the “young man’s statement” card and specify the “ask young man for detail” as its condition card. At this point, the “young man’s statement” card will be hidden at the beginning of the game and shown to player when players try to finish “ask for detail” formula with “young man” character. The logic of using condition cards to given new cards is summarized as Figure 6.

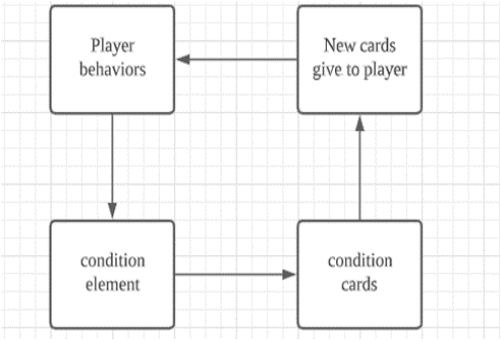

Figure 6: logic of giving feedbacks on player actions

The basic logic of the game editor is that developers only set prerequisite for condition cards and regular cards when making them without worrying about what the cards will trigger. For example, when developers want to build a condition card, they set condition elements that will trigger the condition cards. They do not have to set what cards will be given to players at this time yet. Likely, when developers make a formula or a dialog, they do not worry about what condition cards the dialog or the formula will trigger. In sample game, when developers build the “ask for detail” formula card, they do not know what cards can possibly work with it. When they make the “ask young man for detail” condition card, they do not worry about what cards will specify it as their condition card either. For all cards, developers only need to set when it is triggered or should be given to the players.

Introducing condition cards and condition elements provides developers with the potential to make complicated logic. Triggering of a condition card can work as a condition element of another condition card so the condition card can be nested, forming an “alternative” (or “necessary”) structure of more than 3 condition elements. Figure 7 shows a structure of nested condition cards.

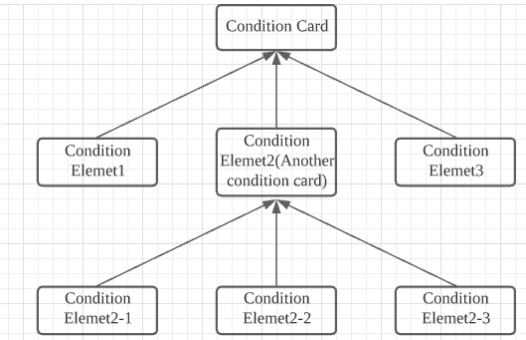

Figure 7: Nested condition cards

Changing of favorability is also an affiliated function of the condition card. When a condition card is triggered, favorability of organization will be changed as defined.

Figure 8 is the UI that will be used to build dialog cards.

Figure 8: Build condition card interface

Other mechanics

The editor also allows developers to share their work with players on other devices. Developers can export their work. The exported game file can be recognized and run on any other device with the editor on it.

Method for evaluating usage of the game editor

The goal of the evaluation is to see whether participants can understand the working logic of the game editor and implement their story in mind into a card-based detective card game. The culture content can be tested in future research.

The research was held via zoom. 4 university level students are invited to participate in the test. They aged from 22 to 25. Two of them are male and two of them are female. I personally invited them based on the consideration of their previous experience with programming languages and game development.

One of them has both game-making experience and programming skill. One only has game-making experience, one only has programming language skill, and one has neither.

Based on learning the usage of a new game editor is challenging already. All of them are speakers of Chinese with some experience with Chinese narrative. I hope this research can focus more on the usability of the card system. Participants were asked to share their screen with me in the whole process so I can see what they were doing great or struggling with.

At the beginning of the test, I asked whether the participants have previous experience in game making or programming language skills.

Then they were asked to follow the in-game tutorial to build the sample game *Find the robber*. The process took around 30 minutes.

The tutorial step by step walks the players through the development of *Find the robber*. Developers are asked to build 19 cards. In the tutorial there are 3 character cards, 4 dialog cards, 2 formula cards, 3 clue cards and 1 ending card. The tutorial is made up with 154 tips in the form of floating text boxes showing the developers to build the game step by step. The tutorial also explains the logic of condition cards, dialog cards and ending cards. The tutorial is designed to explain the working logic of the card system and familiarize developers with the editor by giving them some firsthand experience.

Participants were asked to play with the game they just made to understand players' side of view of the game. That cost about 5 minutes. After that, they were asked to build another simple game with their own imagination. The process took 20 minutes to 60 minutes based on what game the participants try to make.

Finally, we had short interviews about their experience, their difficulties, and suggestions. The interviews took around 10 minutes.

Here are the questions covered in the interviews:

- Please describe the story of the game you trying to build in the game editor, as detailed as possible.
- Do you feel you have to change some content of your story to make it adapt to the format of card game? Please describe.
- Are there any functions or cards type you think would make editor more convenient to use if added?
- Is there any simple logic they want to realize but it takes them excessive repetitive work for you to implement?
- Have you realized that condition cards can be nested to create more complex logic?

Optional questions included:

- I found you stopped at process ****, what problem you have met?
- I found you never made ****, why not?
- What is the most difficult part for you to understand?
- I can now explain that to you, do you feel the tutorial showed the right process?

Results and Discussion

3 of 4 testers finished the task of making a small level of detective game using my editor and their own story.

From the test I learnt that the tutorial did a great job on showing participants how to use the condition cards to make an interactive level. When following along with the tutorial, they all said they found things making sense at some point. Two (one without any programming skills or game development experience) of them immediately understood how to use condition cards to create conditional events in game and 3 of them have noticed the possibility of nesting multiple condition cards to create more complex logic.

The result shows the editor is usable for developers without previous experience of programming or game development.

However, the test also revealed some problems of the existing tutorial system.

The tutorial lacks marks of progression, while the process is long and repetitive for participants. Two of the participants expressed they are upset about the overlong tutorial. They cannot get any feedback during the develop process. Given the complexity of learning a new game editor, I think the existing components in the tutorial are necessary for developers and hard to be reduced significantly. One possible solution

is to have a progression UI in the tutorial system, letting them know how much work is left.

The tutorial does not emphasize the logic that every card should only be specified with prerequisites instead of consequences. One participant got confused about whether condition cards should know what cards will be given to the player on its triggering. This problem can be solved by rephrasing the logic more clearly. The conditional mechanics is the core part of the editor design. Developers cannot create interactive experience unless they understand the logic of condition cards and elements.

Some problems with the editor also showed up during the tests:

In the editor, especially the scene used to build condition cards, there can be three to four layers of forms popping up, asking for different information. For example, when developers try to build a using formula condition element, a form will pop up ask for which kind of condition element are you making. Then, a form will pop up ask for which formula are you going to use. Finally, a form will pop up ask for what answer are you expecting in the formula. That can be confusing for the developers still learning how to use the game editor. This problem can be solved without hurting its current ability to meet the goal of building a card-based detective game by designing a flatter UI layout. For example, it might be better if the form asking for which formula can be placed next to the form asking for which kind of condition element and have some lines to guide develops' sight instead on the top of it.

One tester mentioned that for formulas and dialogs with questions, developers can only make conditions for predicted answers. It could be hard for developers to make conditions for all the not specifically predicted (in most of the cases, if I want the game to react to all wrong answers in a certain way, it would be very difficult). Detective games usually need general feedbacks to all wrong answers, and I also need to build an easier way to implement that in games. A possible solution to this problem is that I can add a "not" structure besides the "alternative" and "or" structures in the condition card building scene. The "not" structure only accepts one condition element, when the element is triggered, the condition card will be not triggered. When the element is not triggered, the condition will be triggered.

When a mistake is made during the development. It can be too much trouble for developers to fix it. A tester found that if a card is built incorrectly all its dependent cards need to be deleted and rebuild. For example, when a condition card is checking whether a formula is used correctly but the formula card is deleted, the condition card will be voided automatically and require a completely remake. That problem decreases the editor's usability to build detective games. A solution to this is I can mark the dependent cards as inactive when their depended cards are deleted and allow developers to go back to their building scene to just redo part of the work related to the deleted cards.

Conclusion

Based on the research result, I believe that the goal of allowing my users to implement their story in mind into a card-based detective card game is in process. Developers are able to turn their detective stories into detective card games using my game editor. However, some functions I provided are not very convenient to use and require improvements.

The strengths of the program include the following: The design of condition card is the very core design of the editor. It makes the editor harder to understand for beginners, but it also increases the consistency of the editor by making all kinds of cards to use condition card to control when they should be given to the players. Also, the possible to nest condition card opens the editor with far more possibilities. For example, if I want a condition card to be triggered when anyone of a group of five condition elements is triggered. I can make a condition card A to hold 3 of them and use another condition card B to hold other 2 of them and condition card A. In that case, the condition card B is the condition card I am looking for. The consistency lowers the learning curve for developers while the nested condition card makes the editor be able to support a wider range of detective stories.

Another strength is that the one participant with no programming experience can understand the working logic and use the editor to build a card-based detective game. That means the editor can also be used by developers with no programing skills.

In future developments, I should provide some kinds of debugging approach to developers to support the game engine. For now, developers can only guess where they did wrong if somethings work differently from their expectation. I should design and build a debugging system to help developers to correct mistakes more easily. For example, for now developers can not see whether the condition cards and condition elements are triggered as their expectation. I can develop a mode letting developers see whether the condition cards and elements are triggered correctly.

In Chinese detective stories, officers usually use bluffing as a method of interrogation. The game editor right now does not support developers to put more creative detective methods in their detective stories. I need to implement some mechanics to provide developers with bigger freedom of their stories.

Citations:

[1] van, G. R. H. (1976). *Celebrated cases of judge Dee = Dee goong an: An authentic eighteenth-century Chinese detective novel*. Dover Publications.

[2] Ting, W. (1974). *Longtu Gongan.* Tianyi Publications.

[3] Lin, Tiejun. (2016). On the Origin of the Ancient Power of Judicial Investigation: According to the Record of Qin and Han Dynasty. *Journal of Political Science and Law, 2016(1),145-153.*

[4] Ace Attorney (GBA version) [Video game]. (2001). Capcom.

[5] Detective Grimoire(PC version) [Video game]. (2014). SFB Games.

[6] Sherlock Holmes: The Card Game [Board Game]. (1991). Schmidt Spiele.

[7] Detective: A Modern Crime Board Game [Board Game]. (2018). Ignacy Trzewiczek.

[8] Zhai Xuewei. (2014). Favor, Face and reproduction of power. *Sociological Studies, 5, 48-57.*

[9] Reigns (PC version) [Video game]. (2016). Nerial.

[10] BKEngine (PC version) [Software]. (2014). Bakerymoe.

[11] Klimas, Chris. (2014). Twine [Software].

[12] Friedhoff, J. (2013). Untangling Twine: A Platform Study. *DiGRA Conference.*

[13] Short, Emily. (2012, November 23). Interview With Porpentine, author of howling dogs. Emily Short's Interactive Storytelling (blog). http://emshort.wordpress.com/2012/11/23/interview-with-porpentine-author-ofhowling-dogs/

[14] Consalvo, M., & Staines, D. (2021). Reading Ren'Py: Game Engine Affordances and Design Possibilities. *Games and Culture, 16(6), 762–778*. https://doi.org/10.1177/1555412020973823

[15] Chover, Miguel & Marín, Carlos & Rebollo, Cristina & Remolar, Inmaculada. (2020). A game engine designed to simplify 2D video game development. Multimedia Tools and Applications. 79. 10.1007/s11042-019-08433-z.