Escaping Ancient China VR

Marina Weber
marina.weber@tum.de

Chennan Zhou
elisazhou0406@gmail.com

Olli Ullgren
olli.ullgren@aalto.fi

***Abstract*—Escaping Ancient China VR is a virtual reality game made in Unity. The project aims to teach the player about ancient Chinese history, focusing on various cultural changes occurring in China during different dynasties. The player learns about different historical objects within the game by solving puzzles related to them. Books and hints throughout the game provide the player with the necessary information to solve the puzzles. Moreover, the player can choose to interact with AI to retrieve more information about a certain object, be it hints to solve the puzzle or additional historical information. These game mechanics create an interactive game environment that engages the player and encourages them to learn more about the history of China.**

***Index Terms*—serious games, extended reality, virtual reality, puzzle, escape room, china, ancient china, chinese dynasty, artificial intelligence**

I. Introduction

This paper describes a course project done in the practical course *Serious Games in Extended Reality* in the summer semester of 2024. The game, made in Unity and using Varjo extended reality equipment, teaches the player about some of the most famous ancient Chinese dynasties. It is an escape room-style game, using puzzles to showcase various aspects of ancient China.

A. Serious Games

The concept of a *serious game* is at the center of the course. According to [1], serious games are ”learning environments that use entertainment and engagement aspects of computer games to attract users to the educational content.” These aspects mean that serious games while being games, are not fully just for entertainment, but they should also include some learning aspects, making the games ”serious.” Unlike other kinds of edutainment, though, they should do this meaningfully – the learning has to be embedded in the gameplay itself instead of having an explicit sequence of ”fun – engage – learn” [1]. This game avoids these explicit quizzes by implementing the learning part within the puzzles and using ChatGPT to make the experience more accessible and immersive.

B. Extended Reality

According to [1], *extended reality* (XR) is an ”umbrella term which covers augmented reality (AR), mixed reality (MR) and virtual reality (VR).” In augmented and mixed reality, virtual elements overlay the real world, whereas virtual reality consists of an entirely virtual setting. Usually, extended reality requires dedicated equipment, which is also the case with this project. The game uses virtual reality, achieved with the Varjo VR headset and HTC Vive controllers, to fully immerse the player in the ancient Chinese escape room.

C. Escape rooms

Escape rooms – virtual or real-life – are games where the main objective is to figure one’s way out of a room. The games are usually completed by solving puzzles in the room, which gives the player ways to advance. The puzzles may be related to each other or be more standalone, and there may be a theme to them, relating to the theme of the escape room game itself. In this game’s context, as mentioned before, the theme is Chinese history, and thus, the game’s puzzles are themed around it.

D. Puzzles in Video Games

Puzzles are commonly found in various types and genres of games, be it extended reality, traditional video games, or even physical games. In short, a puzzle is an interactive part within a game that needs to be solved, usually with the player needing to use either their logical thinking or other skills relevant to the game. In the case of this project, the puzzles require knowledge about Chinese history. The knowledge is learned throughout the game, making the puzzles function as ”quizzes” in disguise, providing the player with learning opportunities suitable for a serious game.

II. Game Description

The following section gives an overview of the idea and concept of the game. The game is an escape room; thus, the plot revolves around the player trying to escape a room they find themselves trapped in. Escape rooms typically provide a reason for players to find themselves in this situation. As the goal of this game is to teach the player about ancient Chinese history, in our game, the player finds themselves trapped in a room full of old Chinese objects. The player learns about Chinese history by examining and interacting with these objects. Moreover, the synopsis described in II-A provides a reason for the player to engage with the topics provided in the game. In this section, there is a brief description of different concepts introduced in the game to make the idea of the game more comprehensible.

A. Synopsis of the Plot

The synopsis of the plot of the game is as follows: The player is a university student who stole a handheld device capable of time travel out of the university laboratory. Because they did not know how to use this device properly, they accidentally teleported themselves into ancient China. On arriving in those foreign lands, they are locked by two strangers into a room, not knowing what those strangers are planning to do to the player. The goal of the game is for the player to find a way out of the room and escape to the present. Since the player's time travel device does not have much energy left, they can only teleport a couple of hundred years back to the future at a time until they find a new way to charge the device. Thus, they will be introduced to different periods and dynasties of ancient China.

B. The Rooms

A crucial concept of the game is the rooms in which the game takes place. Each room represents a particular imperial dynasty and period of China, except for the tutorial room in the beginning and the ending scene. There are puzzles in each room that focus on a specific aspect unique to that period, like the clothing, government system, or inventions of that time. The rooms are arranged chronologically: once the player completes the puzzles in a room, they can travel a couple of hundred years into the future. A more precise description of the rooms can be found in III.

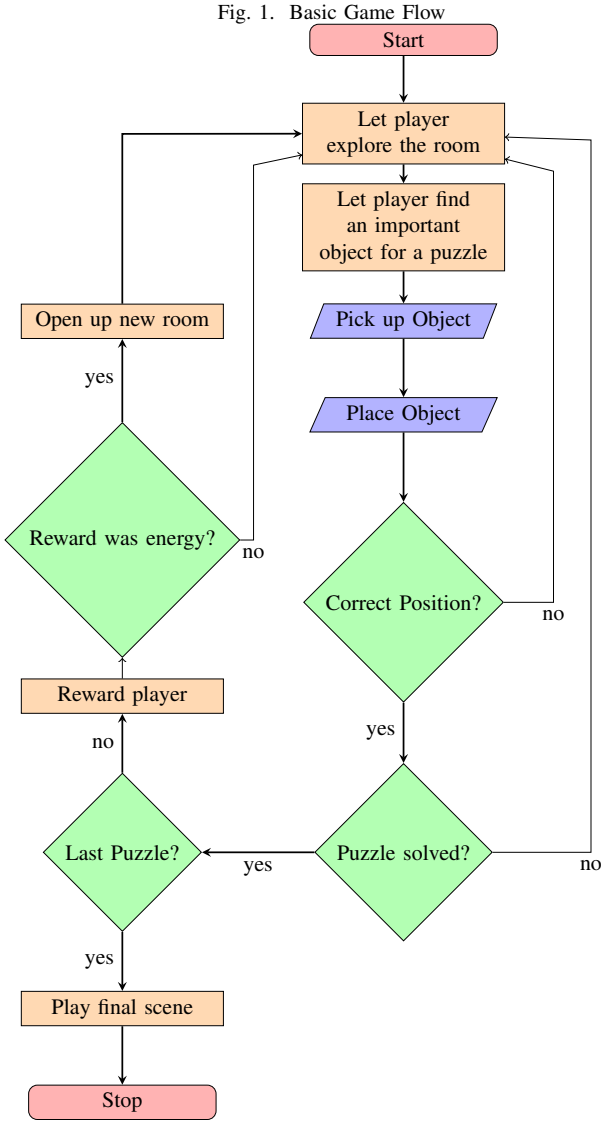
C. Game Flow

In the game, the player explores the rooms and searches for objects that could be relevant to solve a puzzle. The player can solve the puzzles with the found objects in the room. Once they solve a puzzle, there are three possible outcomes:

1) A new, up until then inaccessible area opens up to the player within the room by opening a locked door
2) Furniture and other objects within the room open or start to move, making new objects accessible to the player
3) The player receives energy, which allows them to time travel and thus progress to the next room; see II-G.

The player will continue solving more puzzles using the new unlocked elements until they have completed the game. The player can solve the puzzles by placing objects in a pre-specified correct location. Placing an object in its correct location works slightly differently regarding each puzzle: Sometimes, there are assigned slots in which the objects need to be placed, and sometimes, some objects are in the wrong room as they belong to a different period, and thus the player needs to move them to the correct room. Sometimes, the objects are 2D graphics on a wall that must be rearranged. A graphical description of the primary game flow can be seen in 1. The three reward possibilities that could occur after solving the puzzle are summed up as "reward player" in the flow chart.

Fig. 1. Basic Game Flow



D. Beginning & End of the Game

At the beginning of a game, the player can gain a sense of the controls by playing through the tutorial. This tutorial introduces all the controller inputs the player can use and the core gameplay mechanics. While teaching little about Chinese history, the tutorial room lets players enter the game quickly.

A final cut scene plays at the end of the game. This scene clarifies that the game is over and the player has completed it.

E. Talking Portraits

Talking portraits are an optional gameplay element the player does not need to engage with to complete the game. This gameplay element aims to aid the player. They can use it if they are stuck in a specific part of the game or want to learn more about a particular object. A talking portrait

provides information on how to solve a puzzle, the history of an object, or how the object relates to the broader cultural context in the game. A talking portrait in this game is a portrait attached to a wall. In front of the portrait is a table where objects can be placed. In each room, the portraits that appear to be paintings depict an influential historic Chinese emperor. By placing an object on the designated table in front of them, the portrait comes to life and starts talking about the placed object. This way, the player can gather more information about the object to learn more about the object or receive hints on how this object could be used to solve a puzzle. Figure 2 displays such a talking portrait as it appears in the game.

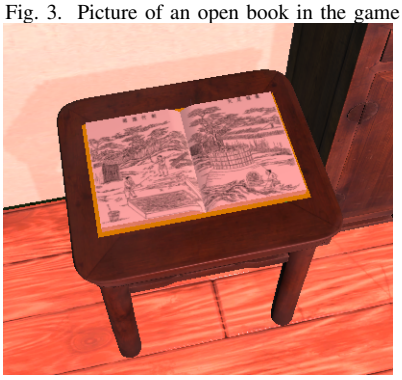
While not necessary to play through the game, this gameplay mechanic is crucial to the game as it allows the player to learn more about the aspects of Chinese history according to their liking. If a different object or topic catches their attention, it provides simple interaction to learn more. Suppose a player is interested in something other than a specific subject. In that case, they can choose not to use the talking portrait game mechanic and learn enough to solve the puzzle using other game mechanics.

Fig. 2. Picture of a talking portrait

F. Books

Books in the game are a gameplay element designed to aid the player. The game has two kinds of books: open books and closed books. The player can interact with open books by turning their pages. They contain helpful information about the puzzles and also general historical knowledge. In every room, the player can find books to guide them through the puzzles. This information aims to teach the player more about the background of the objects in the room. An example of such a book can be seen in figure 3.
Closed books are the second type of book. Unlike open books, closed books cannot be opened but can be picked up. The purpose of these books is to activate a talking portrait by placing the book in front of the portrait. When activated, the portraits tell the player more about the topic described by the book's title.

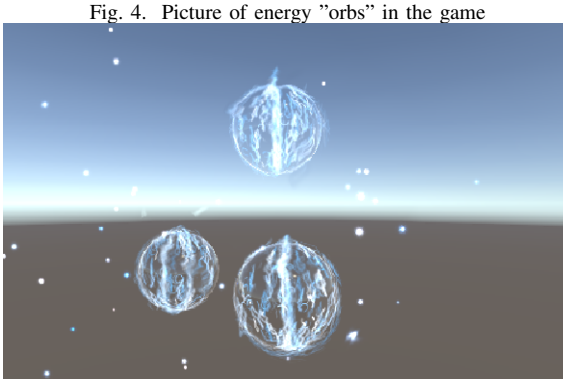
Fig. 3. Picture of an open book in the game



G. Energy

Energy is a game object that the player can absorb and then use to travel through time. The game rewards the player with energy after successfully solving certain puzzles. Energy is a game object that the player can interact with – the interaction being absorbing the energy. With this energy, the player can use their time travel device to shoot a portal to the future. This action is again done using the basic interaction patterns: when standing in front of a predefined, suitable slot for a portal, the player can interact with the slot. This interaction causes an animation to trigger where the player shoots a laser with their time travel device toward the slot. This laser opens up the portal to the next room. A description of the portals follows in section II-H.

Fig. 4. Picture of energy "orbs" in the game



H. Portals

Portals allow the player to move from one room to the next. When first entering a room, the player can see big scroll paintings unrelated to any other game aspect. Those big paintings hide the perfect spot to create a time portal, as they have the ideal size and location. Thus, once the player acquires enough energy to create a portal, they can walk towards a painting. Once they stand close enough, they can shoot a laser, which causes the painting to transform into a time portal. An example of such a transition can be seen in figure 5. Opened portals show a distinct swirly blue glow to visualize traveling in time while passing through the portal.

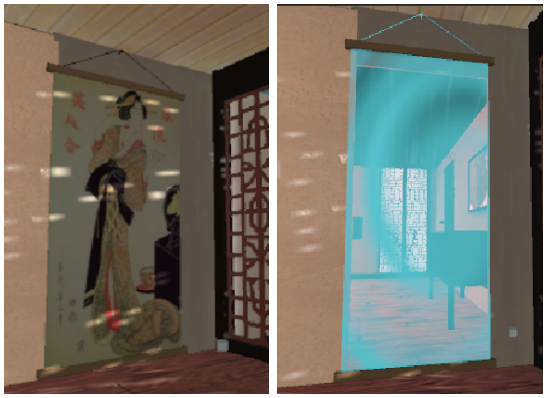


Fig. 5. A painting before and after it became a portal

III. History of China

China has a rich history and a culture that dates back millennia. Over thousands of years, numerous imperial dynasties have ruled over the area that is now China, and numerous inventions, such as gunpowder or paper, emerged there. Chinese history is often neglected in Western school curricula, making the topic very interesting for any learning application. However, the topic is immense; thus, this project only tries to cover a small fraction. The topics covered in the game are taught via the puzzles, which are all centered around a particular aspect of historical China, such as Chinese characters or the evolution of the borders of China over time.

Chosen Aspects of Chinese History

The game world is divided into three rooms, each representing a different period in ancient China. In the following section, we talk about which dynasties are implemented into the game. Furthermore, we elaborate on the chosen emperor representing the dynasty used in the game. These emperors help the player in various ways, as explained in section II-E.

A. First Room: Qin Dynasty and the Beginnings of China (Before 200 BC)

The Qin dynasty, ruling around 200 BC, was the first imperial dynasty of China [9]. Being a minor state for most of its history, the Qin united China under one empire. While the Qin dynasty was the first imperial dynasty, it does not mean that the history of China started from there; the first semi-legendary dynasties are thousands of years older. The first room in the game thus showcases the significant technological developments, such as the changes in weaponry that happened during the millennia. The chosen emperor to represent the dynasty, Qin Shi Huang, was the founder of the Qin dynasty and the first leader of China to adopt the title of emperor, *huangdi* [15].

B. Second Room: Han dynasty (200 BC-200 AD)

The Han dynasty, reigning for approximately 400 years around year 0, saw significant advances in economic prosperity and technological innovations in China [10]. The Han era is widely thought to have been a golden age in Chinese history, and it significantly shaped Chinese identity: the endonym of the Chinese people, *Han*, comes from the name of this dynasty. The second room showcases the innovations of the time, such as paper. The chosen emperor for the room, Emperor Wu, ruled over China for 54 years, one of the most extended times for an emperor of China [12].

C. Third Room: Tang Dynasty (600-900 AD)

The Tang dynasty, the most recent of the chosen dynasties, ruled for approximately 300 years. The Tang era is also considered a cultural golden age in Chinese history. During this time, the bureaucracy of the imperial state evolved considerably in complexity. [11] For this reason, the puzzles in the room are about the government of that time. The chosen emperor for the room, Wu Zetian, was the only reigning empress of ancient China widely regarded as legitimate [13]. She was technically not a part of the Tang dynasty, constituting her own dynasty, but she was both preceded and succeeded by a Tang emperor, making the time period correct.

D. Tutorial Room

The tutorial room is not about any specific period of ancient China but instead focuses on teaching the player how the game works. The chosen person in the talking portrait is not even an emperor of China; instead, being the philosopher Confucius. His philosophical teachings, Confucianism, have greatly affected Chinese cultural heritage and way of thinking [14].

IV. Description of the Puzzles

As mentioned in previous sections, the gameplay consists of several puzzles. This section briefly describes the mechanics and historical background of these puzzles and further outlines the expected learning outcomes for the player.

A. Clothing Puzzle

Over time, clothing styles and fabrics change in any culture. For this reason, we designed a puzzle to introduce the player to the typical garments worn in different periods in China. The player first unlocks a closet in the game containing three pieces of clothing. Every piece of clothing belongs to a different period. In each room, there is a clothing hanger. To solve the puzzle, the player must hang each piece of clothing onto the hanger in the correct room. Figure 6 shows the pieces of clothing once the player can access them.

Fig. 6. Picture of the garments used in the clothing quiz

B. Armor Puzzle

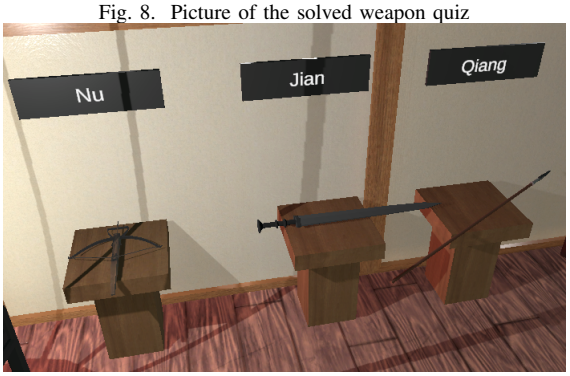
The purpose of this puzzle is to showcase how the materials used to create weapons and armor changed over time. Before the reign of the Qin dynasty, people used bronze to produce weaponry. During the Qin dynasty, the use of iron increased. Over time, the preferred material changed again to steel. Furthermore, the materials used for armor changed accordingly. During the Qin dynasty, simple leather and iron armor was used. During the Tang period, soldiers often wore scaled armor and crested helmets. To solve this puzzle, the player has to move the Tang armor from the Qin room to its correct room. Figure IV-B depicts the armors positioned when the player first encounters them.

Fig. 7. Picture of the armors as they appear in the game

C. Weapon Puzzle

The Chinese had a large amount of interesting and unique weapons. Thus, we introduce the player to some of the most infamous ancient Chinese weapons. The goal of the puzzle is for the player to memorize the types of weapons soldiers used during the Qin dynasty and their Chinese names. To solve this puzzle, the player must put the weapons on the correct pedestal. The pedestals are labeled with the names of the weapons. Figure IV-C shows the solved weapon quiz. The three introduced weapons are the Nu, an ancient Chinese crossbow; a Jian, a famous sword in Ancient China; and the Qiang, a long spear used for close combat.

Fig. 8. Picture of the solved weapon quiz



D. Chinese Character Puzzle

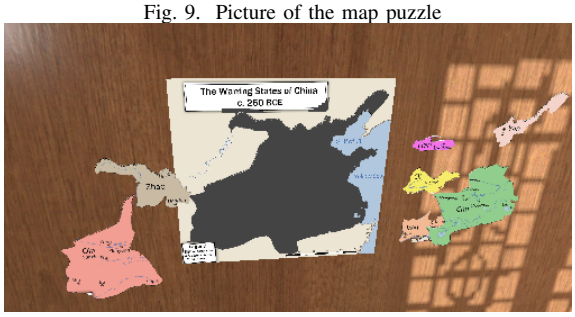
Chinese characters are the writing system used to write Chinese. Among modern writing systems, they are the oldest system that is still in use today. They are also unique among today's writing systems in that they are *logographic* – meaning that instead of phonemes, single characters represent entire words or parts of them. They have evolved considerably from their first forms, the over 3000-year-old *oracle bone script*.

This puzzle showcases and aims to teach the player about the evolution of Chinese characters over time. In the puzzle, the player must connect three variants of a Chinese character: the oracle bone script variant, the *seal script* variant from the 1st millennium BC, and the modern variant by placing tiles that depict the characters into correct slots. The twelve tiles, depicting four different characters, are initially scattered around the room, with some of the tiles hidden until half of the puzzle has been solved.

E. Map Puzzle of the Warring States Period

The Warring States period was a significant era in Chinese history. During the period, China was composed of seven kingdoms. It was the last period before the establishment of the first unified imperial dynasty in China, the Qin dynasty. Each kingdom had its characteristics, and all of them tried to unify ancient China under themselves, leading to many years of warfare. Players learn about the specific locations of each kingdom on the Chinese mainland through this puzzle. Figure 9 depicts the map puzzle as it appears in the game. Upon completing the puzzle, the map breaks into pieces. The player can use these pieces to interact with a talking portrait described in II-E.

Fig. 9. Picture of the map puzzle



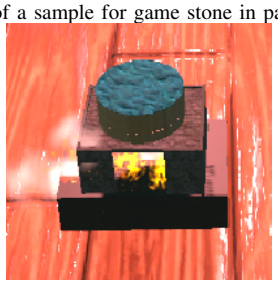
F. The paper making puzzle

Papermaking is one of the *Four Great Inventions* of China, dating back to the Han dynasty. The ancient Chinese made the earliest paper through a series of processes using raw materials such as bamboo and tree bark. The processes through which paper used to be created included soaking, steaming, and drying. The innovation of papermaking was important, as it meant people no longer needed to carve inscriptions to record text, as they could use paper instead. Given its historical significance, we have designed a puzzle for paper making.

The papermaking process, which consists of four main steps, has been transformed into an interactive puzzle. One game stone represents one step. Figure 10 shows a sample of

the game stone. Players learn how the ancient Chinese made paper by arranging these game stones in the correct order. The purpose of this puzzle is to engage players in a fun and educational activity that deepens their understanding of the papermaking process. To assist in solving the puzzle, there is a book in the room containing illustrations that show the correct steps of the papermaking process.

Fig. 10. Picture of a sample for game stone in paper making puzzle



G. Six Ministries System Puzzle of the Tang Dynasty

Ancient China had a variety of administrative systems. The Tang Dynasty utilized the so-called *Three Departments and Six Ministries* system. This system was proposed during the preceding Sui Dynasty and further refined during the Tang Dynasty. It significantly influenced the official administrative systems of subsequent Chinese dynasties. The Six Ministries were divided as follows: the Ministry of Personnel (responsible for the selection, examination, and management of officials), the Ministry of Rites (responsible for all ceremonial rituals), the Ministry of War (responsible for the military), the Ministry of Justice (responsible for law enforcement), the Ministry of Revenue (responsible for financial management), and the Ministry of Works (responsible for construction and engineering). The clear division of labor among the Six Ministries created a robust administrative system.

The player learns about the responsibilities of each ministry by matching the name of each ministry with its corresponding duty. In the puzzle, game stones (Figure 11) represent each duty of a ministry. Meanwhile, since the names of the ministries is labeled in Chinese characters, a Chinese-English reference sheet is also placed in the room.

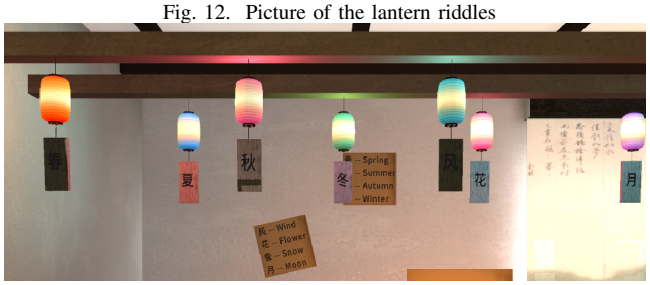
H. Lantern Riddles of the Tang Dynasty

China has many traditional festivals, one of the most important being the Lantern Festival. Celebrated on the fifteenth day of the first lunar month, the Lantern Festival was a significant event during the Tang Dynasty. Families would hang beautiful paper lanterns, and people would host lantern riddle activities in the streets. Riddles were attached to the lanterns, and those who could guess the answers would receive prizes. This engaging and immersive activity is recreated in the Tang Dynasty room, allowing players to experience it first-hand.

Fig. 11. Picture of the game stones in Six Ministries system puzzle

Additionally, Tang poetry is incorporated into this puzzle. Poetry was an essential part of Tang culture, and countless famous poets emerged during this period, making it the most flourishing era for poetry in ancient Chinese history. In the room, there is a book with translations of poems in both Chinese and English. The player will read the poems and guess the items described in each poem. All the available answers are hanging under the lanterns. Figure 12 shows the lantern riddles. Players will experience the traditional Chinese Lantern Festival riddles through this puzzle, gaining a deeper appreciation for the cultural richness of the Tang Dynasty.

Fig. 12. Picture of the lantern riddles



V. Input Devices & Control

The game uses the Varjo XR-3 [3] headset. It is a VR headset capable of mixed reality. The HTC Vive Controller is used for the controller. The trackpad of the controller is used for movement. The sensor in the trackpad recognizes the relative location of the player's finger when touching the trackpad. This location then translates into a 2D interpretation of the direction and velocity with which the player wants to move. Furthermore, the player triggers the teleport functionality by pressing the upper region of the trackpad, as described in VI-B. By pressing the left or right region of the trackpad, the player can turn around 90°. The trigger button at the back of the controller is used for all interactions with objects defined in this report. When pushing the trigger on the back of the controller, the game interprets it as trying to interact with the closest Interactable Object as mentioned in section VI-E.

Furthermore, during the production phase, the keyboard can simulate any of the inputs described above. However, the player can only use the VR setup to play the game in the finished product.

VI. Implementation of Gameplay

This section describes the implementation of all the main scripts in the game in detail. Each section either describes an important class or gives a detailed description of the implementation of a certain game mechanic. VI-A to VI-J describe how the system handles the player input. Sections VI-K to VI-O explain the implementations of the various puzzles described in IV. Starting from section VI-P, we talk about the implementation of various other game mechanics that are part of the main gameplay.

A. PlayerMovement Class

The `PlayerMovement` class implements the basic functionality for moving the player. The class receives the player's input and moves the player within the scene based on this input. The `Update()` function updates the player position and camera rotation. Unity calls this function once per frame.

B. VRInput Class

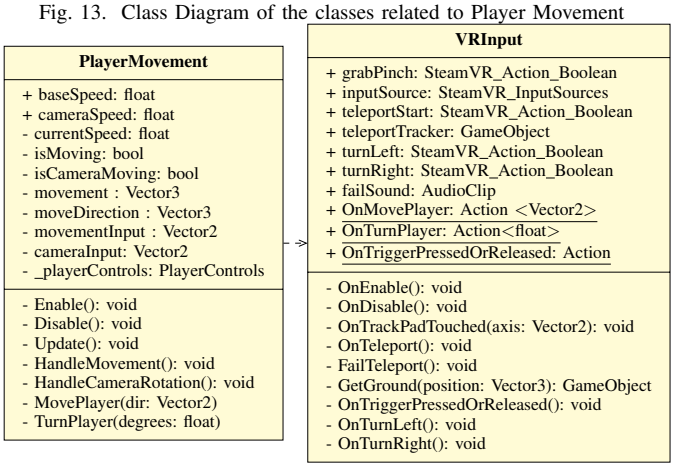
`VRInput` is an additional class that implements the support of VR Controller input. The class listens to the input of the VR Controller and then calls corresponding functions in the `PlayerMovement` and `PlayerInteraction` classes. Moreover, it implements the "teleport" functionality. Teleport means that rather than gradually moving position, the player can choose to select a specific position to teleport to instantly. For a more detailed explanation of the teleport functionality, please view VI-D.

C. Interaction of Player Movement Related Classes

The `PlayerMovement` class handles all player movement except for the teleportation functionality. Because it is written generically, it supports multiple types of inputs, including the keyboard and the HTC VIVE Controller. The `VRInput` class listens to the input of the HTC VIVE Controller and delegates the updated user input to the `PlayerMovement` class. Figure 13 shows the class diagram of the two classes.

The `PlayerMovement` class saves the current user input for movement and camera rotation in `movementInput` and `cameraInput`. While rotating the camera via script is unnecessary in VR, it adds valuable functionality to testing the game with the keyboard. The `Update()` function calls `HandleMovement()` and `HandleRotation()` to update the player's position and camera rotation based on the current input. The `PlayerMovement` class also subscribes its function `MovePlayer` to the `VRInput` action `OnMovePlayer` and `TurnPlayer` to `VRInput`'s `OnTurnPlayer` so that both keyboard and trackpad input yield the desired result.

Fig. 13. Class Diagram of the classes related to Player Movement



D. Teleport Implementation

When the player wants to teleport, they need to press the upper region of the trackpad. Once the player presses it, a curved ray is shot into the scene. The player can adjust the path of the ray by moving the game controller. There is a `TeleportTracker` object at the end of the ray. This object marks the position the player would teleport to if they were to release the trackpad now. This `TeleportTracker` only collides with objects of the layer "Ground". When the player tries to teleport, the game checks whether this location is legal. If the player tries to teleport through walls or areas not yet accessible, the teleport fails. This test uses the `GetGround()` function to determine whether the player tries to teleport onto legal ground. If the teleport fails, a fail sound plays, so the player knows they cannot teleport to this position.

E. Interactable Object Class

`InteractableObject` is implemented as an abstract class. Interactable objects are the most essential objects in the game. All objects the player can use inherit from `Interactable Object`, thus making them crucial to the gameplay. An interactable object becomes highlighted if the player stands close enough to it and meets the necessary conditions, if any. Once the object is highlighted, the player can interact with it by pressing the trigger button of the controller.

F. PickUp Object Class

`PickUpObjects` are `Interactable Objects` that attach themselves to the position of the HTC Vive Controller within the game. The player can trigger this attaching of the object by pressing the trigger button on the controller. This functionality allows the player to carry an object around. They can release the object by pressing the trigger button a second time. `PickUpObjects` are a fundamental aspect of the gameplay, as most puzzles are solved by picking up and moving objects.

G. Place Object Class

PlaceObjects are objects that activates once the player places to correct PickUpObject onto it. Once this object is placed, the game notifies the player by playing a sound. In addition, it may trigger some event to happen, provided the player has already solved the other parts of the puzzle.

H. PlayerInteraction Class

Similar to PlayerMovement, the PlayerInteraction class processes the player's input. It implements the player's different interactions with the game world. The class keeps track of whether the player is within reach of an interactable object; if there are multiple interactable objects, it keeps track of which one is closest to the player. Furthermore, it remembers if the player is currently holding an object. All changes of any of these states are propagated via Unity actions to all objects in the scene so they can adjust their state accordingly.

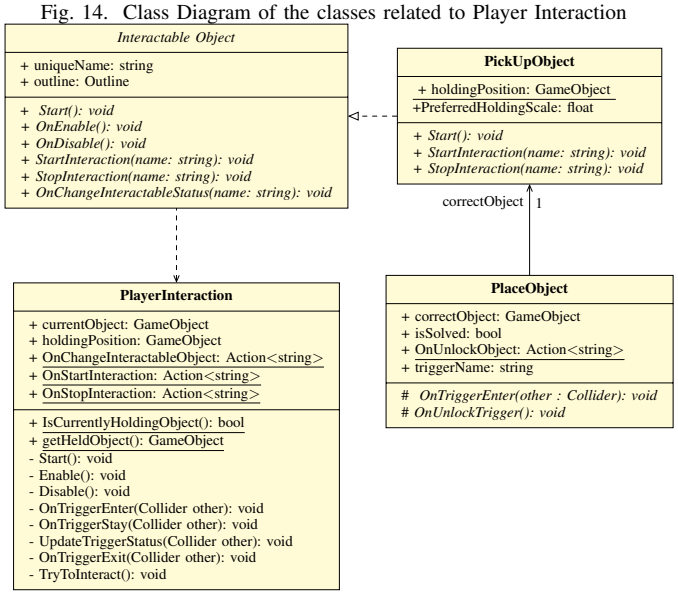
I. Player Interaction Flow Description

PlayerInteraction is a script attached to the player. When a player enters or stays within the trigger area of an object with the Interactable Object script attached to it, it checks whether this is closer to any other Interactable Objects currently in range. The object closest to the player is stored in the currentObject variable. All the objects are notified by OnChangeInteractableObject() that the object that is currently stored in the currentObject variable has changed. This notification is essential because the game visually highlights the object closest to the player. Once the player tries to interact with an object, the TryToInteract function is called. If currentObject is not null, the game triggers its OnStartInteraction function. Figure 14 depicts the classes mentioned above and their associations with each other.

In the case of PickUpObjects, they attach themselves to the holdingPosition GameObject in the OnStartInteraction function. This way, they are now attached to the player, and the player can carry them around. Once the player triggers OnStopInteraction, the object unattaches itself again. Since some objects are big and take up a lot of screen space, a variable called PreferredHoldingScale exists in PickUpObject. If it is set to a value smaller than 1, it shrinks while being carried around.

If the player places a PickUpObject onto a PlaceObject, the PlaceObject checks whether the player placed the correct object on it and reacts accordingly. If the player solves a puzzle, the PlaceObject triggers an event.

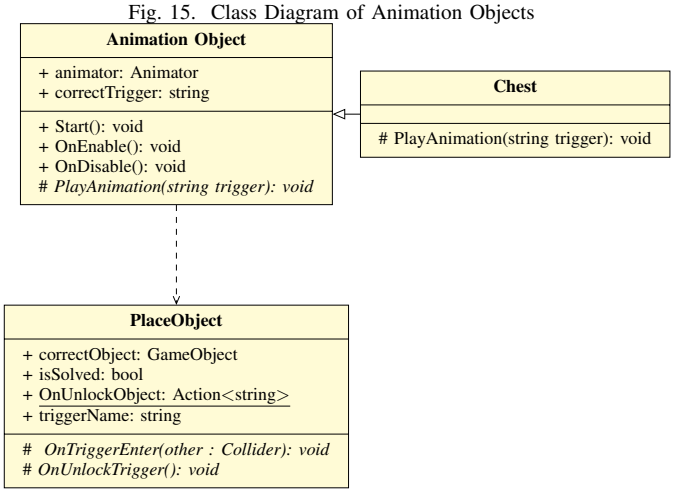
Fig. 14. Class Diagram of the classes related to Player Interaction



J. Animation Object class

Animation Objects subscribe to the PlaceObject OnUnlockObject action. As a reward for completing a puzzle, they play an animation that either opens up a new area or makes new objects accessible (e.g., a closed chest opens). Figure 15 depicts this relation between the PlaceObject and the AnimationObject class. We also distinguish between two kinds of animation objects: "Regular" animation objects and chests. An object that only uses the regular AnimationObject class triggers some animation that makes an area accessible. An object of the Chest class, while not necessarily having to represent a chest visually, is an object that has valuable objects inside. These objects are set as inactive until the animation of the chest starts. Then, the chest activates the objects inside. The activation prevents the player from being able to interact with the objects located inside before unlocking them.

Fig. 15. Class Diagram of Animation Objects



K. The Chinese Character Puzzle

The Chinese character puzzle uses three custom classes in total: `CharacterTile`, `CharacterSlot`, and `CharacterAnswer`. It consists of twelve wooden tiles, each depicting a Chinese character and twelve slots in four groups of three where the character tiles need to be placed correctly.

1) `CharacterTile`: `CharacterTile`, as the name suggests, is used for the wooden tiles depicting Chinese characters. It inherits the class `PickUpObject` and thus also `InteractableObject` and their functionalities. In addition, it keeps track of which Chinese character it represents, irrespective of the historical variant, and if the tile is next to a `CharacterSlot` so that it can be placed into one via its override of `InteractableObject.StopInteraction`. In order to simplify the puzzle, the tile cannot be removed from a character slot if placed into the correct one.

2) `CharacterSlot`: `CharacterSlot` represents the singular slots where the character tiles can be placed. They always appear in the game in groups of three within one `CharacterAnswer`, and the three different `CharacterTiles` depicting the historical variants of the same character need to be placed in the slots. The three tiles representing the different historical variants of the same character can be placed in the slots in any order to make the puzzle more user-friendly.

A `CharacterSlot` tracks if a correct tile is inserted within it. `CharacterSlot` is also an `InteractableObject`, but this interaction only works for taking incorrectly placed character tiles out of the slot.

Placing a tile in the slot works via `CharacterSlot`'s `Interact` function, which is called from `CharacterTile.StopInteraction`. If the slot is empty, the tile is placed into it by changing the `Transform` of the tile with the slot's `PlaceTileInside` function. Otherwise, if there is already a correctly placed tile inside, the slot will attempt to insert the tile into one of the slots next to it via its "parent" `CharacterAnswer`. Finally, if there is a wrong `CharacterTile` inside and the player is holding a tile, `CharacterSlot.Interact` will swap the places of these two tiles. The latter two "special" interactions exist to make the puzzle more playable in VR, as it is often difficult to target one of the three `CharacterSlots` specifically.

3) `CharacterAnswer`: Finally, the class `CharacterAnswer` represents the objects found in the game that contain three `CharacterSlots` and depiction of the character that is supposed to be in the `CharacterTiles` that need to be placed in the slots. Unlike the other classes, a `CharacterAnswer` cannot be interacted with directly. A `CharacterAnswer` keeps track of the correct character and whether the answer is completed – i.e. its slots are filled with correct tiles.

The "completeness" value is updated with the function `checkifSolved()` every time a `CharacterTile` is correctly inserted into one of its `CharacterSlots`. The answer is considered solved if all three slots contain a correctly inserted tile. The class also contains static variables to track whether the entire puzzle is solved. When two of the `CharacterAnswers` are solved, a cabinet door opens, revealing more character tiles, and when all of them are solved, the player receives energy.

L. The Map Puzzle

The puzzle pieces are randomly scattered in various positions on the wall. The player uses the controller to select pieces and place them in the correct positions.

To realize such a map puzzle, we built the `MovePuzzleElement` class, which inherits from the `InteractableObject` class. We add this class as a component to each map piece. In the Start() method, we set the position of the parent of the puzzle piece as the correct position. When the player interacts with a puzzle element, the `StartInteraction` method tracks the controller's position and move the puzzle element. Then, in the `Update` method, the position of the puzzle element is updated for each frame based on the position changes of the controller. When the player stops interacting, the `StopInteraction` method checks if the puzzle element has been correctly placed. If the puzzle element is correctly placed, the `OnPuzzleElementMoved` event is triggered, notifying the system that the element has been correctly placed.

M. Correct Placement of Objects

We provide two classes to determine whether players have correctly placed objects on the answer board.

1) `CheckPlace`: This class acts as the component of the game stone to determine whether the object has been placed in the correct position. We add two attributes to each game stone: `target cube` (answer position) and `object name`. There is also a collider around the game stone. In this class, we use the `OnTriggerEnter()` method to detect if the object has collided with the target cube. If the game stone detects a collision with the target cube, the game stone will be placed on the target cube. At the same time, it will become a child of the target cube and can no longer be moved. Additionally, the material of the target cube will change color to indicate to the player that they have answered correctly. Meanwhile, it will also call `PaperPuzzleComplete.UpdateCount(object name)` to notify that this game stone is correctly placed.

2) `PuzzleComplete`: This class monitors the overall completion of the puzzle. In this class, we implement two lists: `List<GameObject>` targetPositions and `List<string> placedObjects`, and set a counter `correctlyPlaced` (initial value is 0). The `targetPositions` list pre-stores all the target cubes for this puzzle. Whenever `CheckOrder` calls the `PaperPuzzleComplete.UpdateCount(object name)` method, we check if `placedObjects` contains the object name. If it does not contain the object name, we add it to `placedObjects` and increment the value of `correctlyPlaced` by 1. When the value of `correctlyPlaced` equals `targetPositions.Count`, it indicates that the puzzle is completely solved, and the corresponding action will be invoked to trigger animations or events.

N. Placement Puzzles

The papermaking Puzzle, the Six Ministries System puzzle, and the lantern riddles all function in a similar manner: players must place objects in the correct place.

In the papermaking Puzzle, players arrange the game stones correctly to solve the puzzle. When they successfully order the game stones, these stones lock into their designated bases, which then change color to indicate correct placement. In the Six Ministries System Puzzle, players use game stones representing various duties of government departments. To complete the puzzle, they must place these game stones on the corresponding department answer boards. Once all the game stones are in the correct positions, the box will open using an animation. In the Lantern Riddle, players read poems to guess the objects described in each one. The correct answers hang under lanterns. Players use a handle to take the correct answer from beneath the lantern and place it on the answer board in the correct order to complete the puzzle.

We use the methods in VI-M to implement these puzzles. For other necessary functionalities, such as picking up and placing objects, we use `PickUpObject` (VI.F).

O. Remaining Puzzles

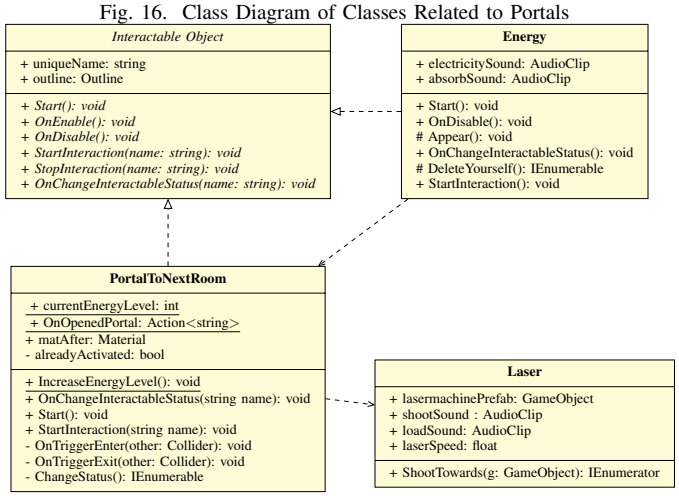
The remaining puzzles (the armor puzzle, the weapon puzzle and the clothing puzzle) only used the classes described in VI. No custom code is needed for them since they mechanically only consist of putting objects into the correct places.

P. Portals & Energy

To travel in time – i.e., between rooms, the player must create portals as described in section II-H. A portal is another interactable object with which the player can only interact, provided they have collected enough energy. Energy is another class derived from `InteractableObject`. As a GameObject, it appears after completing certain puzzles. If interacted with, the energy plays an absorbing sound. After playing the absorbing sound, the GameObject increases the counter for available energy and then destroys itself.

Once the energy counter exceeds zero, the player can interact with the portal. The interaction triggers a laser animation where a laser device appears before the player. The laser device then starts shooting a laser toward the painting, which is about to become a portal. Once the laser hits the painting, the picture gradually changes color until it looks like a portal. From now on, the player can walk through the portal. Figure 16 depicts a class diagram of the classes involved in this process.

Fig. 16. Class Diagram of Classes Related to Portals



Q. Implementation of Opened Books

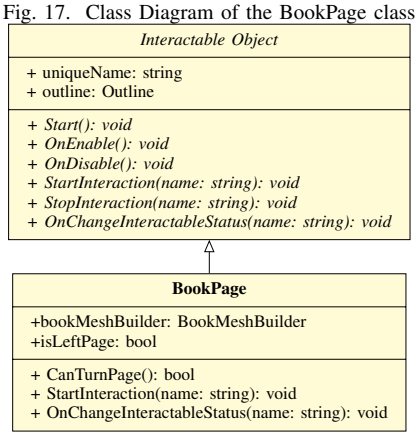
While closed books are simple objects with `PickUp Objects` scripts and no further code is required, open books require an implementation of page flipping. To create books with suitable pages, we use a plugin called Book Effects by Electric Wolf [2]. With this plugin, it is possible to create books by simply describing the individual pages as a list of images.

The class `BookPage` allows the player to interact with open books. This class inherits from `Interactable Object`. If the player interacts with it using the trigger on the HTC Vive Controller, the page turns accordingly. The player cannot turn a page that is the first or last page of the book. The game performs this check with the function `CanTurnPage()`. If the page is first or last, it does not appear as interactable in the game scene. This shows the player that they cannot interact with it. Each book has exactly two GameObjects with `BookPage` as a component: One for the left and one for the right side. Those two objects are independent of which pages are currently displayed, as they only serve as an interface for the player to interact with.

VII. AI USAGE

The game relies heavily on AI to make it more interactive and allow for more freedom for the player. With AI, the player can retrieve more information about any object they

Fig. 17. Class Diagram of the BookPage class



find in the game. The player must place an object before a talking portrait, as described in II-E. The placement triggers a process where AI generates a new text describing the object while pretending to be the person in the portrait. Once the text generation is complete, the video generation starts. In this video, the person in the portrait says the generated text out loud. After downloading the completed video, it is rendered into the portrait; It looks like the portrait itself is talking. The talking gives the illusion of the player being taught by famous people in Chinese history.

Section VII-A and VII-B describe the main classes involved in the process. The remaining sections describe how the different classes interact to create the desired result.

A. Historical Object Class

A `Historical Object` is an object that has some historical information attached to it. This information describes its name, what kind of object it is, and during which period of ancient China it emerged. Any historical object is also a `PickUp Object`; for the player to access the information in the object, the object must be placed in front of a talking portrait. If the object is placed in front of a talking portrait of an emperor from an older dynasty than the object, the person cannot provide any information. If the object is from a period at least as recent as the one of the person in the talking portrait, the person talks about it according to the knowledge of their respective period.

B. Talking Portrait Class

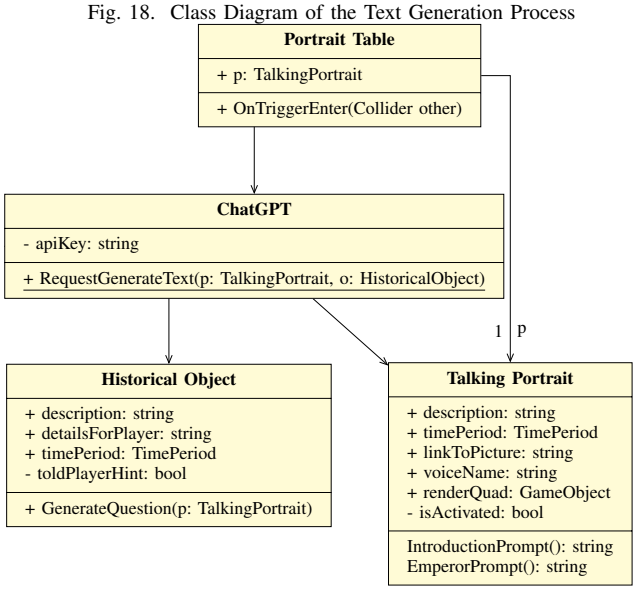
A talking portrait is a portrait that provides information to the player when triggered, which happens by placing a historical object in front of it. If the person in the portrait is from an earlier period than the object, the person informs the player that they do not know what kind of object it is. Otherwise, the player receives information about the history of the object and hints regarding the puzzle to which it belongs.

C. Text Generation Implementation

The generated text depends on various factors. First, if the object is relevant to a puzzle and a hint for the player is provided, the generated text contains this hint. If the player has completed the corresponding puzzle, the game does not use the hint for the generated text. Instead, it continues with the second option: If no hint is provided or the puzzle is solved, the generated text contains historical information about the object.

Whenever new text is generated, there is a statement in the generation prompt, saying ”Pretend to be X,” where X is a short description of the person in question. For example, in the case of Emperor Qin, this game replaces X with ”Qin Shi Huang, the first emperor of China.” Furthermore, in the prompt, we ask to consider the period during which the person lived. Figure 18 depicts the classes involved in this process.

Fig. 18. Class Diagram of the Text Generation Process



`Portrait Table` is a class that recognizes a when the player places a (historical) object on it. When this happens, it calls the function `RequestGenerateText` of the ChatGPT class using its associated `TalkingPortrait` and the information of the Historical Object. The `ChatGPT requestGenerateText()` function creates the appropriate prompt using the information handed over. This prompt is sent as a web request to the OpenAI API. The response contains the generated text. With this generated text, the game calls the video generation process explained in VII-E.

D. Video Generation Flow

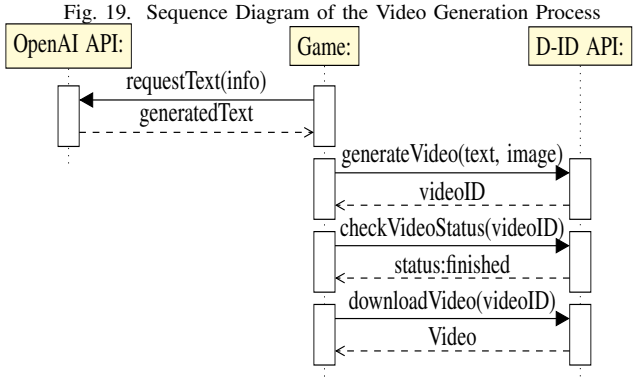
When not interacted with, the talking portraits are static game objects with an image of their respective emperor as a material attached to them. However, if the player chooses to interact with them, a video plays right in front of the game object. The video is rendered onto a separate quad right in front of the portrait. If not used, it is invisible.

A typical flow of such video creation is as follows:

1) The player places a historical object in front of the portrait
2) The placement triggers the start of the video creation
3) Based on the information provided of the object and the portrait, send a request to OpenAI to generate a new text
4) After receiving an answer from OpenAI API, send a request to generate a new video of the portrait talking with the generated text
5) Check if the video was created successfully
6) Download the video
7) Play the video on the quad in front of the talking portrait

Figure 19 depicts this process as a sequence diagram.

Fig. 19. Sequence Diagram of the Video Generation Process



E. Video Generation Implementation

D-iD provides a convenient interface for generating ”talking avatars”. When a character's avatar is provided, and the text or audio is used as input, D-iD will automatically generate the corresponding lip movements on the character's avatar based on the text and audio and create a video. In the game, when the player asks questions from the emperors' portraits in the rooms, it will seem like they are speaking and answering the player's questions.

Requests to D-iD (`https://api.d-id.com/talks`) are sent using simple JSON statements by `UnityWecRequest` to generate the corresponding videos. The JSON Post request used to generate speech is below:

```
{
    ”source_url”: ”image_path”,
    ”script”: {
        ”type”: ”text”,
        ”input”: ”e.g. Hello world!”
        ”provider”: {
            ”type”: ”microsoft”,
            ”voice_id”: ”en-US-JennyNeural”
        }
    }
}
```

After making the request, we receive a response containing the `talkId`. After that, we send a GET request to `https://api.d-id.com/talks/<talkId>` to obtain a specific ”talk.” The response to this GET request will include the target video's URL (result URL), which can be downloaded to get the desired video.

The video can be generated by simply providing the image URL (`source_url`), text (`input`) and selecting an appropriate voice (`voice_id`) from the Microsoft Azure Voices library. These videos are bound to the corresponding grid in Unity using the video player. When people ask questions about the portrait, the lip-sync video, created based on the text generated by OpenAI, will automatically play to introduce or answer the player's questions, similar to the off-site assistance in real-life escape rooms.

VIII. Used Tools

This section introduces the primary tools used to implement the game. As it is a serious game in virtual reality, Unity was used as the game engine, and SteamVR and the ViveInput Utility were used for the VR functionality. Furthermore, various APIs were used to implement certain game mechanics.

A. Unity

Unity [5] is one of the most-used engines for video games. Unity is a game engine that offers various useful utilities, such as handling the physics and sound of the game. The different types of classes are implemented as Unity scripts using C#.

B. SteamVR

SteamVR allows us to connect the VR hardware to the game. It allows us to pair the devices to the system as well as define the bindings of our inputs [4].

C. Vive Input Utility

Vive Input Utility is a unity plugin that tracks the status of Vive input devices [6]. With this plugin, the game implements the player's movement and interaction in VR.

D. Book Effect by Electric Wolf

Book Effect by Electric Wolf is a plugin for Unity [2]. It allows for the creation of books that a user can open, close, or turn its pages. The front and back cover, as well as the spine, can have custom textures. Pages are stored as a simple list of textures that users can edit in the Unity Editor.

E. OpenAI API

OpenAI API is an API that allows for text generation based on simple prompts in natural languages [7]. It also offers the ability to convert text to a sound file via text-to-speech.

F. D-ID API

D-ID is an AI API that generates a video out of a static image [8]. The image is a portrait of a person, and the resulting video shows the person talking. The content of the person's speech can either be defined by some text or an audio file. In case of only providing a text, the API generates the audio in addition to the video.

IX. Outlook

The following sections describe further improvements and limitations of the project. When developing the game, we encountered some issues that could not be solved within the course of that semester.

A. Limitations of the D-iD API

When using the D-iD interface, the source URL of the image must be provided. However, the D-iD interface only accepts links to a file server. Furthermore, in our use case, most images are inaccessible by the D-iD interface due to data protection and copyright issues, resulting in the error *HTTP/1.1 451 Unavailable For Legal Reasons*. Although D-iD offers temporary storage before creating an animation and provides a corresponding endpoint, most images still cannot be uploaded for copyright reasons.

This temporary storage of the D-iD API also allows for uploading an image via the API. However, even when the same JSON file and C# code provided in the documentation example were used, and the JSON POST request was provided on the official web interface, the API consistently returned the error *HTTP/1.1 400 Bad Request*.

Since the API call does not work, we rely on uploading the images through the D-iD official web interface beforehand and obtaining the corresponding endpoint to ensure the talk creation function of D-iD is correctly used. However, this temporary storage only provides 24-48 hours of storage.

To address this limitation in the future, creating a dedicated host server, uploading the images there, and then granting D-iD access permissions could be considered. This way, D-iD may successfully read the information needed in the image file.

B. Checking for Misinformation

Currently, no checks are in place to ensure that the text generated by the OpenAI API is factually correct. This lack of checks can lead to teaching the player wrong information. While the player does not need to use the information generated by the text to play the game, this is nevertheless a problem that can occur.

C. Inclusion of NPCs and Human Models

There are brief sections where the player can hear someone talk in the game. However, the player can never see anyone—they have their eyes closed in the game or are in a different room. Using animated models of people could increase the player's immersion.

X. Conclusion

Escaping Ancient China VR is an immersive, interactive VR game that teaches the player about ancient Chinese history. The game displays various cultural and societal achievements during the Qin, Han, and Tang dynasties. The player learns about these aspects by solving multiple puzzles related to Chinese history. If the player wants to learn more about a specific topic related to any object in the game, they can retrieve more information by interacting with AI. The AI is disguised in the game as a series of portraits of important figures in Chinese history that start talking once activated. Even though the API used to generate these portraits currently has some limitations, the game achieves its goal of teaching the player the desired topics in an entertaining manner.

XI. Work Load Distribution

The following tables describe the assigned tasks of the different project members for both the project itself and this report, which is relevant for the grading of the project.

Olli Ullgren checked the report for grammar mistakes.

TABLE I
Workload of the project work

Description of task	Responsible person
Creation of rooms & outside environment (layout, game asset acquisition, sound, lighting)	Marina Weber
Armor puzzle	Marina Weber
Clothing puzzle	Marina Weber
Weapon puzzle	Marina Weber
Chinese character puzzle	Olli Ullgren
Map puzzle of the Warring States Period	Chennan Zhou Marina Weber
Paper making puzzle	Chennan Zhou
Six Ministries system puzzle	Chennan Zhou
Lantern Riddles	Chennan Zhou
Connecting all the puzzles and other game objects to one logical game flow	Marina Weber
Transition Animations (Occur when the player finishes a puzzle and unlocks something)	Marina Weber
Integrating Usage of OpenAI API	Marina Weber
Integrating Usage of D-iD API	Chennan Zhou
Implementing basic interaction & movement mechanics	Marina Weber
Tutorial	Marina Weber
Ending scene	Olli Ullgren
Research about historic facts	Olli Ullgren Chennan Zhou Marina Weber
Portal design	Olli Ullgren
Laser & Portal transition animation	Marina Weber
Book implementation	Marina Weber

TABLE II
WORKLOAD OF THE REPORT

Section	Written by
I	Olli Ullgren
II	Marina Weber
III	Olli Ullgren
IV-B, IV-C, IV-A	Marina Weber
IV-D	Olli Ullgren
IV-E, IV-F, IV-G,IV-H	Chennan Zhou
V	Marina Weber
VI, VI-A, VI-B, VI-C, VI-D, VI-E, VI-F, VI-G, VI-H, VI-I, VI-J, VI-O, VI-P, VI-Q	Marina Weber
VI-K	Olli Ullgren
VI-L, VI-M, VI-N	Chennan Zhou
VII-E	Chennan Zhou
VIII, VIII-B, VIII-C, VIII-D, VIII-E, VIII-F	Marina Weber
VIII-A	Olli Ullgren
IX , IX-B, IX-C	Marina Weber
IX-A	Chennan Zhou
X	Marina Weber

REFERENCES

[1] E. Bozkir. (2024) Introduction to Serious Games in Extended Reality. Lecture slides.

[2] Book Effect Documentation. http://electricwolf.co.uk/Unity/BookEffect/BookEffect.html. Accessed on 10.07.2024.

[3] Varjo XR-3 official website. https://varjo.com/products/varjo-xr-3/ Accessed on 13.07.2024.

[4] SteamVR Official website. https://www.steamvr.com/de/. Accessed on 13.07.2024

[5] Unity Official website. https://unity.com/de. Accessed on 13.07.2024

[6] Vive Input Utility official documentation. https://developer.vive.com/resources/vive-sense/tool/vive-input-utility/ Accessed on 13.07.2024.

[7] OpenAI API official documentation. https://platform.openai.com/docs/overview Accessed on 13.07.2024

[8] D-ID official website. https://www.d-id.com/ Accessed on 13.07.2024

[9] Wikipedia: Qin dynasty. https://en.wikipedia.org/wiki/Qin_dynasty

[10] Wikipedia: Han dynasty. https://en.wikipedia.org/wiki/Han_dynasty

[11] Wikipedia: Tang dynasty. https://en.wikipedia.org/wiki/Tang_dynasty

[12] Wikipedia: Emperor Wu of Han. https://en.wikipedia.org/wiki/Emperor_Wu_of_Han

[13] Wikipedia: Wu Zetian. https://en.wikipedia.org/wiki/Wu_Zetian

[14] Wikipedia: Confucius. https://en.wikipedia.org/wiki/Confucius

[15] Wikipedia: Qin Shi Huang. https://en.wikipedia.org/wiki/Qin_Shi_Huang