**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Immersive Media Programming**

**AR Team Project 『AR Indian Poker Game』**

**Technical Document**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Group 5

Song Jihye, Jang Yoona, Kim Yejin, Kim Gyeongjin

**System Architecture**

We used three different scenes for our AR Indian Poker game, which are the Main scene, GamePlay scene, and GamePlay2 scene. And the program is run in the order of the Main scene, GamePlay scene and GamePlay2 scene. Below are descriptions of each scene and important components of each scene.

1. **Main scene**

: Scene that the player can see when they first connect to the game.

* Main components of the scene

1. Start button

: Button to start the game

: Player can enter the game by pressing the button. In detail, we move on to the gameplay scene where we can set the order for the game to proceed and check the opponent’s card.

1. Option button

: Button to adjust additional elements needed for the game.

: When the player clicks the corresponding button, the player will see a manipulator that can adjust the sound of the game.

1. Quit button

: Button to exit the game

1. **GamePlay1 scene**

: A scene where the player can decide who will start betting first between the computer and the player, and check the opponent’s card.

* Main components of the scene

1. Proceeding remarks

: Comment explaining to the player how to proceed and what to do

1. Decide the turn by Arrow

: Arrow is the yardstick for deciding who will bet first, the player or the computer

: Touching the arrow object stops the arrow object that was rotating, pointing in either computer or player direction. The player in the direction of the arrowhead points to bet first.

* Detail for implementation

: We already randomly extract either 0 or 1 through random variables before the player touches the arrow. If the value of the variable is 0, the player takes precedence, and if it is 1, the computer takes precedence. Thus, if the value of the variable is 0, the arrowhead points in the player direction when touched, and if the value of the variable is 1, the arrowhead points in the computer direction (opposite of the player).

1. Check the counterpart(computer)’s card by Marker and Card object

: We use marker-based image tracking for checking counterpart(computer)’s card. Once the marker is detected, a random card appears on the marker. Card the randomly selected from a total of 10 cards ranging from diamond ace to 10.

* Detail for implementation

: Card objects initially exist without material. We implemented that one of the materials from Diamond ace through 10 is randomly applied to card objects and displayed when a marker is detected.

1. **GamePlay2 scene**

: A scene where the player can create a board to place the chip on, play a game, and check the results at the end of the game.

* Main components of the scene

1. Set the gameboard

: We use plane tracking for generating game boards to bet chips. If the plane is detected through plane tracking, the indicator is activated, and when the screen is touched, the game board is created at the location where the indicator points to.

1. Manipulation of Player’s turn

: At the start of the game, each player has 20 chips so they cannot bet more than 20 in total. In the player’s turn, one chip bets on each touch of the game board. After betting as many chips as the player wants, press the ok button to finish betting. Pressing the die button will terminate the game because it means that the player will give up the game.

* Detail for implementation

: The chip is created in a player-touch position on the game board. If a player touches the outside of the game board, the chip won’t create and bet.

: Players must bet the same or more chips than the opponent bet just before. Therefore, if the player bets fewer chips than the opponent bets just before, the ok button is not activated. In other words, the ok button is activated and can be pressed when the player bets the same number of chips than the number of chips the opponent bets just before.

: If the player has to make an all-in because the number of chips left is less than the number of chips previously betted by the opponent, the ok button will be activated when all remaining chips are betted.

1. Betting method for computer’s turn

: Betting on a computer is performed differently depending on the betting trend selected at random. Computer betting trends are divided into timid/stable/aggressive, with one trend selected at random for each round. In addition, among the three trends, we set the tendency to bet reliably to be chosen with high probability.

1. UI for checking player and computer’s chips

: At the bottom of the scene, the remaining chips of the opponent(computer) and the player are displayed. The UI is updated at the end of each bet. A Player can also check their opponent’s cards through the UI.

1. Show the final result

: After the game, the player’s card and opponent(computer)’s card are created on the game board along with the game exit announcement, allowing the player to check his or her own card.

1. Continue button / Quit button

: At the end of the game, the player can decide whether to exit the game and return to the main scene or proceed to the next round as the number of chips each currently has.

**Key AR features used within the project**

1. Image Tracking

: We used image tracking techniques to recognize marker images and generate card objects on top of them in the process of players checking card objects on a computer. Unlike just checking the opponent's card through the ui image, the marker is used to create a 3d card object, which induces the user to feel a little more imperative, similar to actually checking the opponent's card in the real Indian poker game.

1. Plane Tracking

: We used plane tracking to create a game board where players and computers could bet their chips. Just like betting chips on the floor or on the gameboard when playing a real Indian poker game, the project recognizes the floor and allows it to create a board for the game. In addition, we also made it impossible to bet chips outside of the game board. In other words, it has detailed settings so that chips can be placed only on the game board.

1. Interaction with virtual 3D objects

: We thought that just showing virtual objects on the screen would reduce immersion. Therefore, we implemented within the project that allows players to interact with virtual 3d objects through certain behaviors (like tabs on the screen). As an example, it allows users to interact by touching arrow objects to sequence bets between the player and the computer. Also, when creating a game board or betting a chip, it also allows users to tap the screen to create objects in the desired location. We added appropriate sound effects when performing actions such as betting chips, making the player feel more realistic.

**Description of scripts and methods**

1. MainUI Script

: Script for setting the behavior of buttons such as Start, Options, Quit in the main scene.

* StartGame(): Methods for navigating to a gameplay scene
* NextScene(): Methods for navigating to a gameplay2 scene
* QuitGame(): Methods for shutting down an application

1. OptionUI Script

: Script for controlling option part in the main scene

* Awake(): Methods for receiving and setting animation components to set animation effects when opening and closing Option windows.
* Start(): Method for initial setting to adjust the volume for background music and sound effects. The default values for the background music and sound effects are set to 1.
* Update(): Methods that call the BackgroundMusic() method and the EffectSound() method every frame to update the setting values immediately.
* BackgroundMusic(): Methods that allow player to control the volume of background music using the slider
* EffectSound(): Methods that allow player to adjust the volume of sound effects using the slider
* Close(): Method for turning off the Options window. Call the CloseAfterDelay() method through the Startcoroutine function.
* CloseAfterDelay(): Methods for giving a delay for a period of time and turning off the Options window

1. ProcessTextChange Script

: Script for set up text UI to explain game progress in the gamePlay scene.

* Start(): Welcome to the game with a text UI.
* Update(): Explaining the sequencing process, and inform who will bet first based on the result via text UI.
* DividingText(): Display a request to recognize markers to generate cards by text UI.
* NextScene(): After card verification, the text UI displays a request to click the button to move on to the scene for the game's start.

1. WhoFirst Script

: A script that determines who is the line between the player and the computer, and thus determines the direction in which the arrow stops.

* Start(): It receives a component for raycasting, and determines who will bet first, the computer or the player, through random variables.
* Update(): Continue to check if arrow objects are touched, and if not, continue to rotate. If touched, the arrowhead points to the player who will bet first, computer or player, and the arrow disappears four seconds later.
* TryGetTouchPosition(out Vector2 touchPosition): After confirming that the touch has been made, it returns true when it detects the touch.

1. MultipleTracking Script

: Script for recognizing markers and creating a computer’s card for them.

* Awake(): Methods for setting marker image information so that markers can be recognized
* OnEnable(): Method for subscribing to the trackedImageChanged event
* OnDisable(): Method for not subscribing to the trackedImageChanged event
* OnImageChanged(ARTrackedImagesChangedEventArgs args): Methods that control the situation in which the marker image was first recognized, when it was maintained after it was first recognized, and when it disappeared.
* UpdateImage(ARTrackedImage img): A method that shows the card objects on your computer on top of the marker image in situations where it is okay to show the cards after the image is first recognized (after a player has finished sequencing).

1. RandomMaterial Script

: Script for randomly applying material to a card object when showing a computer’s card.

* Start(): It calls the render component to apply material and calls the ChangeMaterial method.
* ChangeMaterial(): The material selected randomly through the SelectRandomMaterial method is applied to the material of the current card object.
* SelectRandomMaterial(): Select one of the materials (diamond ace to 10) stored in the material list randomly and return it. The material selected now becomes a computer card.

1. GameManager Script

: Scripts that store overall variables and functions for gameplay

* Awake():Enable other scripts to use variables and methods in this script through GameManager.manager by making the manager variable always points to this script.
* Start(): Invoke the StartGame function
* StartGame(): Make the game state “inGame”
* makeDefault(): Initialize all variables in the script to their initial state.
* SetGameState(GameState state): Method for determining the state of the game. It specifies what activities to do as they switch to each state. For example, in the gameOver state, the FinalMotion function is performed.
* GetGameState(): A method that retrieves the current state of the game and returns it.
* GameOver(): This method is invoked at the end of the game, which switches the game state to gameOver and disables settings for game progress.
* FinalMotion():It is a method that executes the action when the game is over. It executes the finalResult() function, which shows the cards of the player and the computer.

gameSetting():It is a method that changes a computer's card into an int form based on material information. Players can also randomly assign cards. At this time, the player's card was set to not overlap with the computer's card. We set the betting propensity of the computer to be determined randomly among stability, timidity, and sensitivity. Among them, stability was more likely to be chosen.

* PlayerAct(): Each time a player makes a turn, it calculates how many turns he has made and sets the player to a betting state. It also activates the die button to be pressed.
* TurnEnds(): A method that updates information about the number of bets when the player or computer finishes betting, and then passes the turn to the next player. Set the game to end if the computer or player has called without raising, or to hand over the turn to the next player.
* wasItCall():A method to tell if the player or computer called when betting. Return true if you called, false if you didn't.
* ComAI(): A method that performs betting on a computer in turn according to the previously selected computer betting propensity. In the case of aggressive betting, the computer never dies first, but bets with the feeling that it will make the opponent die. If a computer's betting propensity is stable, the number of bets is adjusted according to the number of computer cards, and if the player's card count is small, a trick is used to prevent the player from dying. If a computer makes a timid bet, it relies the most on the player's card numbers, makes the player mistake that his number is high, and implements it so that com is more likely to declare a die.

1. EnterSound Script

: Script to give sound effects when pressing ‘Let's bet’ UI to move on to the next scene (gamePlay2 scene)

* SoundEffect(): Method to set the sound to sound when a button with the text Let's bet!

1. DontDestroy Script

: Scripts that are required for the object to be retained when a scene is switched. In this project, when moving from the GamePlay scene to the GamePlay2 scene, it was used to convey the variables stored in the Game Manager script.

* Awake(): Set all components of the game object containing the script to remain unchanged as the game scene changes.

1. S2ProcessText Script

: Script to set up text UI that explains the progress of GamePlay2 scene

* Start(): Set text UI to provide guidelines for creating game boards
* ForPlayerText(): Sets text UI to indicate that it is a player's turn and calls the playeract method for player’s betting.
* ForComText(): A method for setting textui to remind the player that it is the computer's turn. It bets according to the comAI function, and if the computer dies, it calls the Whodied method to deal with the die situation.
* WhoDied(): Find who declared die and let us know who did die through text UI.
* NextTurn(): A method that tells you who bet and how much the player bets when the player hits the ok button or when the computer finishes betting through textui. It also calls the invokeTurnEnds method to help deal with situations where the turn is over.
* Finish(): The method called when the game is over, updates the number of chip panels and notifies textui that the round has ended. It also calls the GameOver function in the gamemanager script to help perform actions at the end of the game.
* invokeTurnEnds(): It calls the TurnEnds method in the Gamemanager script to help deal with situations where the turn is over.

1. PlacementIndicator Script

: Script to create an indicator for plane tracking

* Start(): Methods that provide initial settings for generating indicators
* Update(): A method that sets the indicator to be visible when the plane is recognized.

1. GameProcessor Script

: Script to manage the overall progress of the game

* Awake(): The method set up to use a script that contains the methods needed to run the game.
* Start():A method that performs the initial setup required at the start of the game, such as card distribution and chip count setting for each player as the game proceeds in the previous round.
* Update(): A method that directs the game system to perform actions accordingly depending on the variable values for the order, whether the manager is enabled, and the current betting situation on the player or computer.
* TryGetTouchPosition(out Vector2 touchPosition): Methods for determining the position of player’s touch
* IsPointerOverUIObject(Vector2 touchPos): Method to check if the player touched the UI

1. Play2UI Script

: Script responsible for updating UI for button, chip panel in GamePlay2 scene

* DieButton(): Script that controls the situation when a player presses a die button.
* OKButton(): Script that controls when a player presses the ok button. If a player all-in his or her chips, finish the game, or hand over the next turn to the computer.
* UpdateComText(int update): A method that updates the remaining number of computer’s chips to text UI according to the number of bets when the computer finishes betting.
* UpdatePText(int update): When a player bets, the player updates the remaining number of chips to textui according to the number of bets.
* ComCardImageUpdate(int cardNum): A method that sets the card of the computer to be checked through the UI in the gamePlay2 scene.
* continueButton(): When the game is over, press the continue button to set up the next round of the game while keeping the number of chips that each player currently owns.
* backButton(): When the game is over, press the back button to make all variables default and set them to return to the main scene.
* ResultUI(): A method that allows UI to tell you who won the game and how many chips each had at the end of the game.

1. FallingSound Script

: Script for sound effects when a chip tagged object falls on the board

* Start(): Method for getting audio components to give sound effects
* OnCollisionEnter(Collision collision): Every time an object with a hip tag and a game board collide, it is set to make a sound of chips falling.

1. RoundSavor Script

: Script to store the number of chips each has when the player tries to play the next game as it is by pressing the continue button.

1. PlayerCard Script

: Script for applying material to player cards

* Start(): Methods for applying material to player cards
* Update(): When a player's card object is created after the end of the game, this method allows the player to easily check the card by setting the card in the direction of the player (AR camera).
* ChangeMaterial(): Method for applying the material of an object based on the material returned from the SelectRandomMateiral method.
* SelectRandomMaterial(): Function that returns the corresponding material based on the card information assigned to the player (playerCardNum).

1. ComCard Script

: Script for applying material to cards on a computer

* Start(): Methods for applying material to computer cards
* Update(): When a card object for a computer is created after the end of the game, this method allows the player to easily check the card by setting the card in the direction of the player (AR camera).
* ChangeMaterial(): Method for applying the material of an object based on the material returned from the SelectRandomMateiral method
* SelectRandomMaterial(): Method to return the corresponding material based on the card information (comCardNum) assigned to the computer.

1. LookCam Script

: Script to set the text specifying each card to look at the AR camera when showing the card as the final result.

* Update():A method that sets the text object containing the script (text that specifies whose card each card is) to look in the direction of the player (AR camera).