International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. #, No.# (Mon-20..)

AR Xiangqi: Augmented Reality Chinese Chess with Gesture based Interaction Play with AI Opponent

Aida Ali[1*], and Ajune Wanis Ismail [1]

[1]ViCubeLab, Faculty of Computing,
Universiti Teknologi Malaysia, 81310 Johor, Malaysia

aida@utm.my, ajune@utm.my

Received ## Mon. 20##, Revised ## Mon. 20##, Accepted ## Mon. 20##, Published ## Mon. 20##

Abstract: Traditional chess games have been left behind, and people are missing out on the satisfaction of this intellectually challenging and strategic game. The era has embraced digitalized board games which nowadays are considered a way for attracting people to pick up the board game again, and Augmented Reality (AR) is one of the advantageous technologies to be utilized. AR provides irreplaceable digital and physical blending experiences to society. This work presents AR Xiangqi, an augmented reality of the traditional Chinese chess board game that implements gesture-based interaction for gameplay and artificial intelligence search algorithms as the AI opponent. The AI engine can suggest the best next move to players as well as immediate feedback to enrich the UX gaming experience as it enhances interactivity and creates an immersive gameplay experience. Evaluations carried out to assess AR Xiangqi were based on the usability and user experience of the players. The findings indicate that the majority of evaluators conveyed contentment with the user interface of the game, recognized the effectiveness of hand gesture interaction, and valued elements like prompt feedback, destruction effects, and suggestions for the next move. These results confirm that the game has effectively provided an immersive and pleasing AR gaming experience, as respondents expressed satisfaction with different facets of the design and features of AR Xiangqi.

Keywords: Augmented Reality, Xiangqi, Artificial Intelligence, Hand Gesture, and Human-Computer Interaction.

1. Introduction

Augmented Reality (AR) enriches the real-world environment by integrating virtual content into it. This is achieved through the use of technology which delivers sensory stimuli involving sound, digital visual elements, or other means [1]. In contrast to virtual reality, which creates a wholly separate virtual environment, augmented reality incorporates virtual content into the actual world. Essentially, AR blends elements of the real world with virtual environments, allowing users to witness virtual objects seamlessly integrated into the real world through the cameras of devices equipped with AR applications.

A board game is a tangible form of entertainment where participants manipulate or position pieces on a specially designed surface featuring a pattern. Board games can be classified into three distinct types: the first type demands tactics and strategy to outmanoeuvre opponents, the second relies solely on luck for victory, and the third integrates aspects of both strategy and chance. Typically, board games have specific objectives to accomplish, such as defeating the opponent's king or eliminating all pieces belonging to the adversaries from the game.

Chinese chess, also known as Xiangqi, stands as the most widely embraced board game in China. Originating around 200 B.C., the game is credited to the military commander Han Xin. Like its Western counterpart, Chinese chess involves two players, each commanding a set of 16 pieces. The red side moves first, followed by the black side. Victory is achieved by capturing the opponent's king. To garner increased interest in this traditional chess game, an AR approach featuring gesture-based interaction is proposed. Currently, AR can be achieved through various technological innovations, encompassing general hardware components, displays, software, and more. Over time, there has been significant progress in AR technology. It is perceived that AR is not confined to specific technologies and does not impose restrictions on human vision. As a result, there is potential

E-mail:ajune@utm.my

for its application to other senses, such as smell and touch [2], promising a substantial improvement in the immersive user experience.

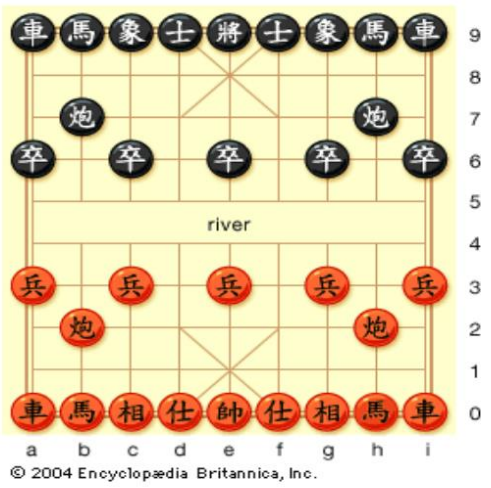
There are two varieties of AR: marker-based AR and markerless AR. A marker-based AR relies on a static and distinct image that acts as a marker, which users can scan using an AR application on smartphones. Upon scanning, virtual content such as videos, 3D objects, or animations is superimposed on the image. Marker-based AR offers the advantage of delivering excellent user experiences alongside its structured and stationary nature. It is user-friendly, requiring no complex instructions to access the AR content [3]. In contrast, a markerless AR doesn't necessitate a specific marker to retrieve virtual content; instead, it operates through scanning the local environment. Virtual content can be showcased using any element which acts as a marker from the physical world. Generally, markerless AR requires users to find a level and textured surface to showcase elements of AR. The registration and object tracking techniques in markerless AR are more complex as opposed to marker-based AR. However, when its content is positioned on a surface, markerless AR proves to be more adaptable than marker-based AR [4].

This work focuses on a Chinese chess board game with markerless AR implementation and an Artificial Intelligence (AI) engine as the opponent. The AR Chinese chess game has been developed to enrich the Chinese chess-playing experience, aiming to create an immersive atmosphere during gameplay and attract a broader audience to engage in Chinese chess. By merging the traditional Chinese chess board game with AR, a novel gaming experience is crafted, complete with exciting enhancements. The incorporation of AR serves to visualize the game, introducing additional effects to enhance the overall project. AR technology contributes to a delightful gaming experience, featuring pleasing sound effects for the Chinese chess player. Moreover, AR offers valuable hints to players before they make their moves, empowering them to choose the optimal move and avoid violating the game's rules.

2. CHINESE CHESS

At the start of a Chinese chess game, there are a total of 32 pieces, typically presented as disks with Chinese characters engraved or inscribed on the top surface. There are seven distinct chess pieces in Chinese chess, featuring one general, five soldiers on each opponent's side, and pairs of other pieces. The initial arrangement of Chinese chess pieces is illustrated in Figure 1. Soldiers are positioned on lines 3 and 6, while cannons are on lines 2 and 7. On lines 0 and 9, from left to right, the pieces are arranged as chariot, horse, elephant, advisor, king, advisor, elephant, horse, and chariot.

Figure 1. The initial position of Chinese chess piece [5].


車 馬 象 士 將 士 象 馬 車
炮 炮
卒 卒 卒 卒 卒
river
兵 兵 兵 兵 兵
炮 炮
車 馬 相 仕 帥 仕 相 馬 車
9 8 7 6 5 4 3 2 1 0
a b c d e f g h i
© 2004 Encyclopædia Britannica, Inc.

The game board resembles the Western chess board with an 8 x 8 square layout, but it includes an extra horizontal void representing a river. Nevertheless, Chinese chess is played at the intersections of the lines, resulting in a 9 x 10 panel rather than an 8 x 8 panel. Each player on the Xiangqi board has a palace consisting of 9 points marked by diagonal lines, within which the king and two advisors are placed and are restricted from leaving. The movement rules for each piece are detailed in Table 1 below.

TABLE I. THE CHESS PIECE MOVEMENT RULES.

Chess Piece	**Movement Rules**
General	It can move one point orthogonally, but it is restricted from moving outside its palace.
Advisor	It can move one point diagonally, but it is restricted from moving outside its palace.
Elephant	It can move two points in any diagonal direction, provided there is no piece on its path, and it is restricted from crossing the river.
Horse	It can move one point orthogonally, followed by one point diagonally, given that there is no piece on its path. It can cross the river.
Chariot	It moves akin to the rooks in Western chess, having the capability to travel any number of points orthogonally, provided there is no obstruction on the path.
Cannon	Similar to the chariot. Except that, the cannon is required to leap over precisely one of any pieces along its path.
Soldier	Advance one point forward until it crosses the river, after which it gains the ability to move one point sideways. Movement backwards is prohibited during the game, and once the soldier reaches the bottom, there is no further progression

3. Proposed Method

The AR Xiangqi system comprises three main elements: the AR user interface for the Chinese chess board game, hand gesture recognition, and the AI opponent. Within the AR segment, plane detection functions as the tracking method, and the virtual, registered objects include the 3D Chess game comprising the chess pieces and the board, accompanied by AI opponent and gesture interaction. ManoMotion [6] is employed for hand gesture recognition, facilitating interaction with the chess pieces. Figure 2 below illustrates the system component and framework of AR Xiangqi. The decision to utilize markerless AR with plane detection as the tracking method was based on its ability to prevent occlusion caused by hands covering the marker. AR Xiangqi is a single-player game, with an AI engine playing as the opponent by calculating the best move during gameplay.

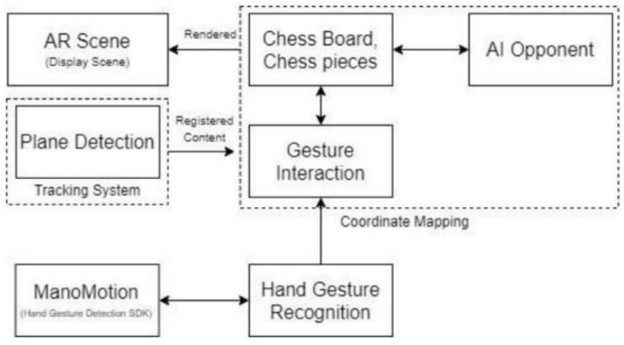


Figure 2. System framework and component flow.

A. *AR Scene and Plane Detection*

AR Xiangqi uses markerless-based tracking. The “AR Plane Manager” in AR Foundation [7] streamlines the process of plane detection by leveraging the camera of the device, creating a simple operation. Integration with AR Foundation involves downloading the package from the Unity Package Manager of version 4.2.7.

For AR plane detection, the "AR Plane Manager" is incorporated as a module in the AR Session Origin. It acts as a camera, converting attributes of tracking into the final orientation, scale and position for the scene on the device. Figure 3 depicts the setting used for the module.

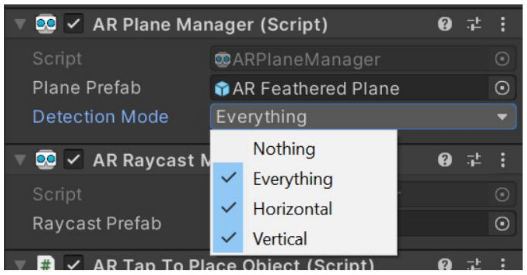

AR Plane Manager (Script)
Script: ARPlaneManager
Plane Prefab: AR Feathered Plane
Detection Mode: Everything
Nothing
✓ Everything
✓ Horizontal
✓ Vertical
AR Raycast M
Script
Raycast Prefab
AR Tap To Place Object (Script)

Figure 3. The AR Plane Manager in AR Session Origin.

In our experiment, we employed two depth sensors. Each depth sensor captures point cloud data comprising vertices, triangles, colour, and UVs (“U” and “V” denote the axes of the 2D texture), as depicted in Fig 1. Once we successfully acquire point cloud data from both depth sensors, we subject this data to a double compression process before transmitting it to the network.

A Plane Prefab is employed to represent the identified planes, offering developers the option to design a custom plane prefab or obtain one from the Unity Asset Store. The Detection Mode is configured to "Everything," enabling the identification of both horizontal and vertical surfaces in the surroundings.

Upon plane detection and the visibility of the plane prefab, the chess board can be placed onto the plane. To achieve this, a specialized class is generated to implement the required functionality, and the incorporation of the AR Raycast Manager is crucial as an element for this class. The function in the class conducts ray casting to ascertain whether the ray intersects with the plane. Only after the ray successfully intersects with the plane, the chess board is instantiated and placed accordingly

B. *Chess Board Game Design*

The chessboard is crafted as a three-dimensional model with 3D chess pieces. An abstract class named Chessman was formulated, and distinct subclasses were generated for each category of chess piece. This method facilitated the development of dedicated classes for General, Advisor, Horse, Chariot, Soldier, Cannon and Elephant, all inheriting attributes from the Chessman class. Central to this project is a crucial class known as "Chessboard," which manages the majority of game mechanics. It supervises vital functions like chess piece selection, chess piece movement, and interactions with the AI opponent.

This work incorporates two varieties of Chinese chess pieces. The first set features traditional Chinese chess pieces labelled in Mandarin characters. Recognizing that some players may not comprehend Chinese characters, an additional set of chess pieces has been designed, utilizing 3D models to visually represent the meaning of each character. Figure 4 shows the two types of chess piece models on the chess board.

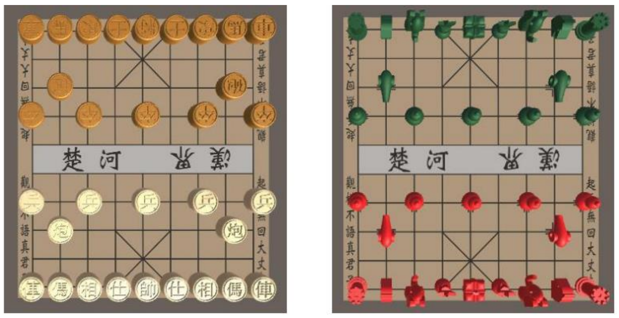


Figure 4. The chess pieces as 3D models and the chess board.

The AR Chinese chess board game operates on a turn-based system for two players. Once the first player completes their turn, the player's round concludes, and the second player assumes control. Players are obligated to make a move and cannot skip their turns. The objective is to capture the opponent's general piece. During a player's turn, they must choose one of their pieces and position it at a right and available point. Subsequently, the AI opponent calculates the optimal move for the chosen piece and then executes it. The game continues until one of the general pieces has exhausted all viable escape routes, at which point the opponent is declared the winner. Figure 5 below illustrates the flow chart of game logic for the AR Xiangqi.

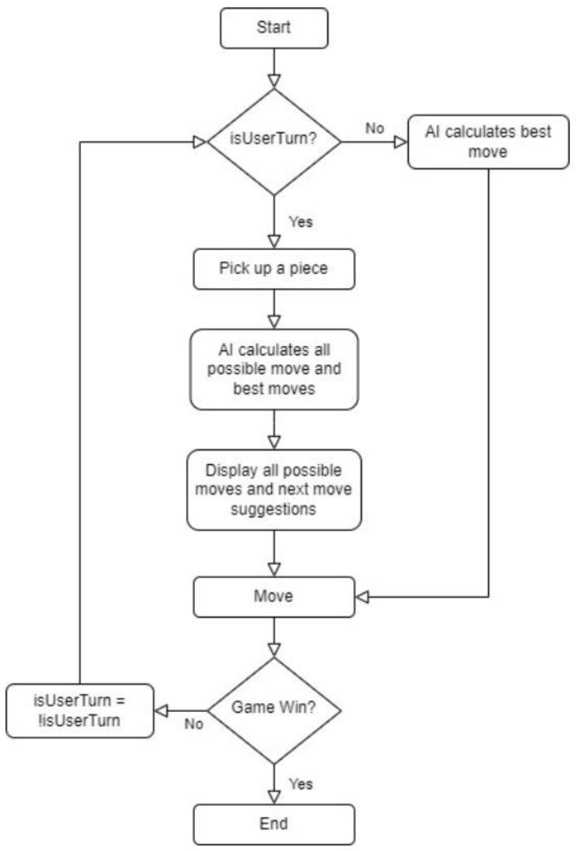


Figure 5. Game logic for AR Xiangqi.

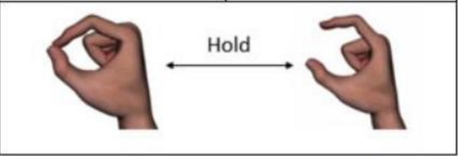
C. *Hand Gesture Design*

This work utilizes the ManoMotion SDK, which comes equipped with pre-built functions for recognizing hand gestures like clicking, picking, dropping and more. Nevertheless, accurately determining the coordinates of the hand presents a challenge. ManoMotion offers hand positions in viewport coordinates, ranging from 0 to 1. Yet, here, the coordinates extend beyond this range due to two factors.

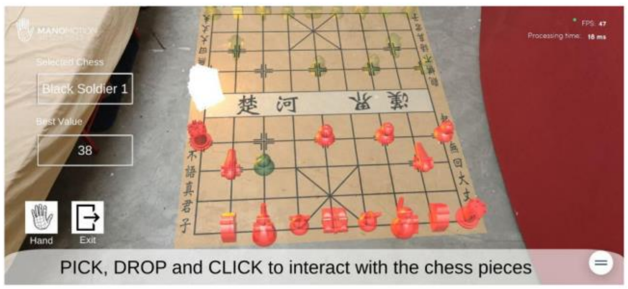
Moreover, when the hand approaches the screen's edge, its detection accuracy may diminish, leading to coordinates falling within the range of 0.1 to 0.9. To address this limitation, the "ViewportToScreenPoint" function is utilized to transform the viewport coordinates into screen coordinates. Subsequently, the transformed screen coordinates are employed for ray casting, mimicking the tapping process on the screen.

The "Pinch" gesture involves the movement of the thumb and index finger, allowing actions like "Click," "Pick," or "Drop" by opening and closing this pinch gesture. The integration of the "Pinch" gesture in this project seeks to enhance the user experience in playing the Chinese chess board game, making it more intuitive and natural. Additionally, the inclusion of the screen tap gesture may also be considered, providing users with an alternative to compare it with bare-hand gesture interaction. Table II illustrates the "Pinch" gesture interaction for playing AR Xiangqi.

TABLE II. MANOMOTION GESTURE INPUTS.

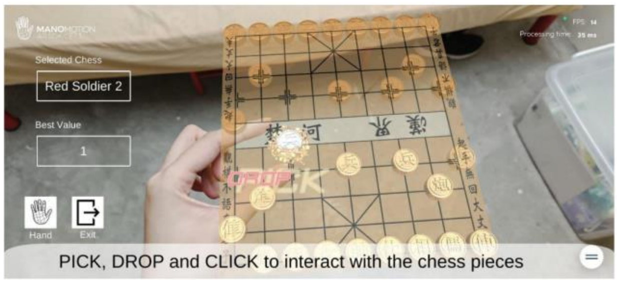
ManoMotion Gestures		
Continuous	**Trigger**	
Open Pinch	Click	
	Immediately	
Closed Pinch	Pick	Drop
		

During gameplay, the open pinch gesture represents the click action, whereas the closed pinch gesture encompasses both the pick and drop actions. Figure 7a illustrates the AR Xiangqi chess board with chess pieces whereas the pinch gesture interaction during gameplay is shown in Figure 7b.



PICK, DROP and CLICK to interact with the chess pieces

(a) AR Chess board with destruction effect to show the piece has lost



(b) Pinch gesture interaction during gameplay

Figure 6. AR Xiangqi interface for (a) shape pieces and (b) character pieces.

D. *Minimax algorithm for turn-taking strategy*

The first step entails assigning both the position value and the piece value to every chess piece. The piece value signifies the relative importance of each chess piece, whereas the position values represent their effectiveness in various positions. These values draw inspiration from the concepts presented in [8], with minor adjustments to accommodate the algorithm and search depth. To prevent prolonged computation times, the AI opponent algorithm is limited to a search depth of three.

In this context, "depth" pertains to the count of moves or steps ahead that the AI search algorithm takes into account while assessing possible moves and reaching decisions. It signifies the extent of prediction or forecasting moves that the AI utilizes to predict the outcomes of its engagements. The minimax search tree is constructed by utilizing the piece value, position value, and threats from the other chess pieces, employing the equation depicted in the equation below.

$$\sum_{p\in black}^{chessboard\ array\ size} p\ (strength(p) * position(p)) - \sum_{q\in red}^{chessboard\ array\ size} q\ (strength(q) * position(q))$$

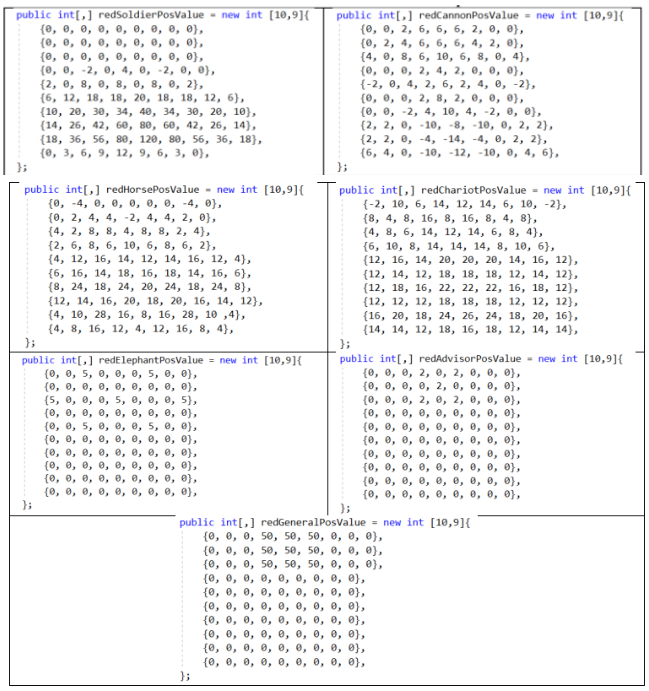
The minimax algorithm calculates the chess board value by considering both the piece value and position values. The values for both the red and black are computed independently with the equation above. To determine the value of a chess piece in a particular position, its piece value is multiplied by the corresponding position value. The chessboard value is then obtained by subtracting the red value from the black value using this approach. Referring to the formula given the black side denotes as $\epsilon\ black$, by taking the summation of all the product of the black piece value $strength(p)$ and its corresponding position value $position(p)$, its difference together with the corresponding summation of the product of the red side defines the value of the board

The General is assigned the largest value, specifically 1000, among all the chess pieces, reflecting its paramount significance in the game. The values of other chess pieces are defined by considering their strength and flexibility. Also, position values for the pieces have a crucial part, with higher values assigned to positions that give a risk to the General. An estimated value for each chess piece at specific positions is stored in a 2D array. Given the distinct movement patterns of various pieces, their position value matrices also differ. Table III illustrates the piece values for each chess piece, while Table IV shows the position values for red chess pieces. The position values for black pieces are computed, mirroring the values derived from the red chess pieces' position values.

TABLE III. INITIAL VALUES FOR CHESS PIECES.

General	Advisor	Elephant	Horse	Chariot	Cannon	Soldier
1000	2	2	4	5	5	1

TABLE IV. POSITION VALUE FOR THE RED CHESS PIECE.



```
public int[,] redSoldierPosValue = new int [10,9]{
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, -2, 0, 4, 0, -2, 0, 0},
    {2, 0, 8, 0, 8, 0, 8, 0, 2},
    {6, 12, 18, 18, 20, 18, 18, 12, 6},
    {10, 20, 30, 34, 40, 34, 30, 20, 10},
    {14, 26, 42, 60, 80, 60, 42, 26, 14},
    {18, 36, 56, 80, 120, 80, 56, 36, 18},
    {0, 3, 6, 9, 12, 9, 6, 3, 0},
};
```

```
public int[,] redCannonPosValue = new int [10,9]{
    {0, 0, 2, 6, 6, 6, 2, 0, 0},
    {0, 2, 4, 6, 6, 6, 4, 2, 0},
    {4, 0, 8, 6, 10, 6, 8, 0, 4},
    {0, 0, 0, 2, 4, 2, 0, 0, 0},
    {-2, 0, 4, 2, 6, 2, 4, 0, -2},
    {0, 0, 0, 2, 8, 2, 0, 0, 0},
    {0, 0, -2, 4, 10, 4, -2, 0, 0},
    {2, 2, 0, -10, -8, -10, 0, 2, 2},
    {2, 2, 0, -4, -14, -4, 0, 2, 2},
    {6, 4, 0, -10, -12, -10, 0, 4, 6},
};
```

```
public int[,] redHorsePosValue = new int [10,9]{
    {0, -4, 0, 0, 0, 0, 0, -4, 0},
    {0, 2, 4, 4, -2, 4, 4, 2, 0},
    {4, 2, 8, 8, 4, 8, 8, 2, 4},
    {2, 6, 8, 6, 10, 6, 8, 6, 2},
    {4, 12, 16, 14, 12, 14, 16, 12, 4},
    {6, 16, 14, 18, 16, 18, 14, 16, 6},
    {8, 24, 18, 24, 20, 24, 18, 24, 8},
    {12, 14, 16, 20, 18, 20, 16, 14, 12},
    {4, 10, 28, 16, 8, 16, 28, 10 ,4},
    {4, 8, 16, 12, 4, 12, 16, 8, 4},
};
```

```
public int[,] redChariotPosValue = new int [10,9]{
    {-2, 10, 6, 14, 12, 14, 6, 10, -2},
    {8, 4, 8, 16, 8, 16, 8, 4, 8},
    {4, 8, 6, 14, 12, 14, 6, 8, 4},
    {6, 10, 8, 14, 14, 14, 8, 10, 6},
    {12, 16, 14, 20, 20, 20, 14, 16, 12},
    {12, 14, 12, 18, 18, 18, 12, 14, 12},
    {12, 18, 16, 22, 22, 22, 16, 18, 12},
    {12, 12, 12, 18, 18, 18, 12, 12, 12},
    {16, 20, 18, 24, 26, 24, 18, 20, 16},
    {14, 14, 12, 18, 16, 18, 12, 14, 14},
};
```

```
public int[,] redElephantPosValue = new int [10,9]{
    {0, 0, 5, 0, 0, 0, 5, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {5, 0, 0, 0, 5, 0, 0, 0, 5},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 5, 0, 0, 0, 5, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
};
```

```
public int[,] redAdvisorPosValue = new int [10,9]{
    {0, 0, 0, 2, 0, 2, 0, 0, 0},
    {0, 0, 0, 0, 2, 0, 0, 0, 0},
    {0, 0, 0, 2, 0, 2, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
};
```

```
public int[,] redGeneralPosValue = new int [10,9]{
    {0, 0, 0, 50, 50, 50, 0, 0, 0},
    {0, 0, 0, 50, 50, 50, 0, 0, 0},
    {0, 0, 0, 50, 50, 50, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0},
};
```

Once the board value is calculated for both the black and red sides, the minimax search tree is generated by adding the array of every possible move for each chess piece into the children's list. For example, when it is the red side's turn to move, it first takes all the red chess pieces that are possible to move into a list. The minimax searches the possible moves for each of the stored red chess pieces in the list similar to a player selecting a red chess piece and all the possible moves for the piece are shown.

When calculating the best move for the AI opponent, alpha-beta pruning is implemented on the minimax search tree. This is due to the limitation of minimax exhaustive searching strategy particularly when the tree is too large to traverse down. The Alpha Beta method is a recursive function that conducts a depth-limited search in the game tree, yielding the best evaluation value for the recent position. As shown in Figure 7, this pseudocode takes into account the current status of the chess board, the current depth for the search tree, alpha and beta values, and a Boolean value signifying either the current player is a maximizer or minimizer. If the traversal has reached maximum depth, the evaluation value of the recent position is returned by the algorithm. If not, it continues with the minimax search.

```
function evaluate (board, depth, maxPlay, alpha, beta)
   if depth = 0
      return value of board
   if maxPlay   //maximizer player
      value = -1000
      for each chessman on board
         value = max(value, evaluate(newBoard, depth+1, false, alpha, beta)
         alpha = max(alpha, value)   //get maximum value
         if beta <= alpha   //prune if alpha >= beta
            break
      return value

   else   //minimizer player
      value = 1000
      for each chessman on board
         value = min(value, evaluate(newBoard, depth+1, true, alpha, beta)
         beta = min(beta, value)   //get minimum value
         if beta <= alpha   //prune if alpha >= beta
            break
      return value
```

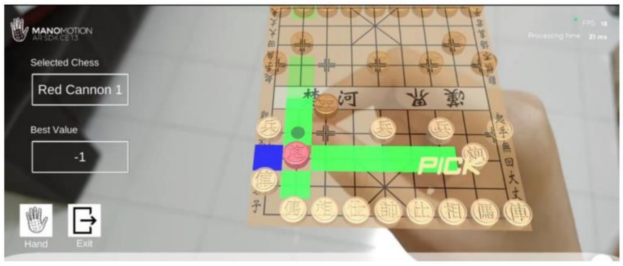
Figure 7. Pseudocode for alpha-beta pruning on minimax search tree

For the maximizer player, the algorithm starts by assigning the evaluation value the smallest possible value before searching for the king of the maximizer player on the chess board. Then, it repeats over each chess piece belonging to the maximizer player and produces potential moves. It generates a fresh board state for each move, checks for threats to the king, and recursively calls itself for the updated board state. After the function updates the evaluation value, pruning is then implemented based on the alpha and beta values.

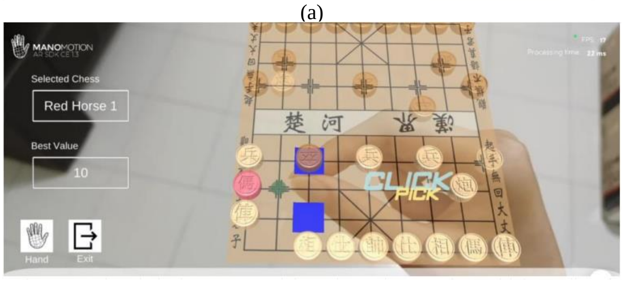
The function trails the same procedure for the minimizer player, but the evaluation value is the opposite where it starts as the largest possible value. It first looks for the king of the minimizer player on the chess board, then iterates over each chess piece that belongs to the minimizer player, and produces potential moves. In each move, it evaluates whether it poses a threat to the king, then recursively calls itself for the updated board state while the maximizer flag is set as true, it updates the evaluation value. Based on alpha and beta values, the function then implements pruning.

In the end, the Alpha Beta function passed the best evaluation value discovered during the search. The parameter "maximize" is configured as false for the AI opponent to determine the best move, and conversely, it is set to true to propose movements to the player. Figure 8 illustrates how the AI algorithms calculate the next best move and suggest the player during gameplay.

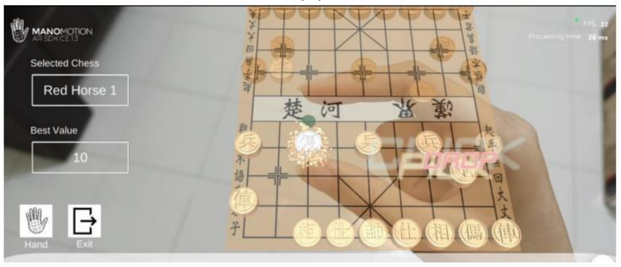
During this time of play, the player is picking up the Red Chariot piece and we can see that there are green squares and blue squares overlaid upon the AR chess board. In Figure 8 (a), the green squares represent all the available moves for the chess piece picked up by the player whereas the blue squares represent the next best move suggested by the AI for the current player in turn. The value of the best move calculated for the next move is displayed in the "Best Value" box on the mobile screen.



(a)



(b)



(c)

Figure 8. AR Xiangqi suggests the next best move to the player with immediate feedback

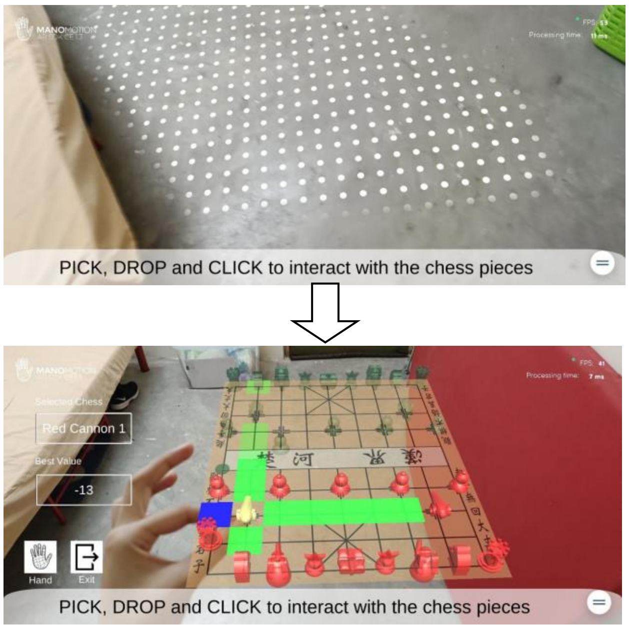
As in Figure 8 (b), the player uses the closed pinch gesture to pick up the Red Horse piece with immediate feedback displayed on the UI chess board. The player uses an open pinch to drop the piece on one of the available next moves as in Figure 8 (c). The player uses the open pinch gesture to drop the Red Horse piece followed by the destruction effect when the opponent's piece is lost We can see the immediate feedback from the chess board by displaying Pick and Drop moving text display on a mobile screen simultaneously when the player is using the closed pinch and open pinch gestures during real-time gameplay interaction.

4. GESTURE-BASED WITH AI OPPONENT

Implementing hand gesture interaction can be as challenging as implementing chess mechanics. In this project, the ManoMotion SDK is utilized, which offers built-in functions for detecting hand gestures such as picking, dropping, clicking, and more. However, obtaining the accurate position of the hand coordinates is a challenge.

This work utilizes two primary technologies, AR Foundation and ManoMotion. The prototype enables users to engage with virtual game objects using their hands, providing an immersive experience. Additionally, the prototype includes an AI opponent that implements the Alpha Beta Pruning Algorithm and provides suggested moves to the user, aiming to suggest the best possible move.

When a player enters the game, they hold the handheld device's camera and search for a suitable plane to display the chessboard. The detected plane is then visualized, as depicted in Figure 9 before the gesture has been overlaid on the plane. To position the chessboard, the user can tap on the screen. Subsequently, they can manipulate the chess pieces by using their fingers to pick and drop them. It is important to note that the entire hand must remain fully inside the handheld device's screen; otherwise, ManoMotion will not be able to detect the player's hand.


PICK, DROP and CLICK to interact with the chess pieces
Red Cannon 1
Best Value
-13
Hand
Exit
PICK, DROP and CLICK to interact with the chess pieces
Figure 9. The user interacts with chess pieces using hand gesture

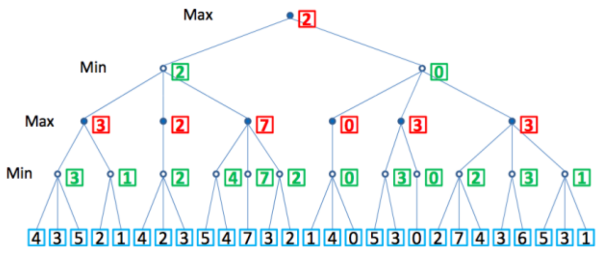


Figure 10. The minimax search tree for AI opponent

Figure 10 illustrates a minimax search tree, the minimax tree represents a chess game, with each level or depth corresponding to a player's turn. The value is propagated from the leaf nodes to the parent nodes based on the min/max levels, aiding in the determination of the player's optimal move. Although this is a simplified tree, in actual Chinese chess, there would be additional branches, each representing a potential move by a chess piece.

To enhance the performance of the minimax tree and reduce the processing time, alpha-beta pruning is used when there are numerous branches at each level. This technique effectively avoids the evaluation of states that are unlikely to provide a better solution to the current situation. By keeping track of alpha, which represents the maximum value, and beta, which represents the minimum value, unnecessary evaluations can be pruned, leading to faster and more efficient searches in the game tree. Referring to Figure 11, the white boxes indicate that unnecessary evaluations are pruned when the alpha value becomes greater than the beta value

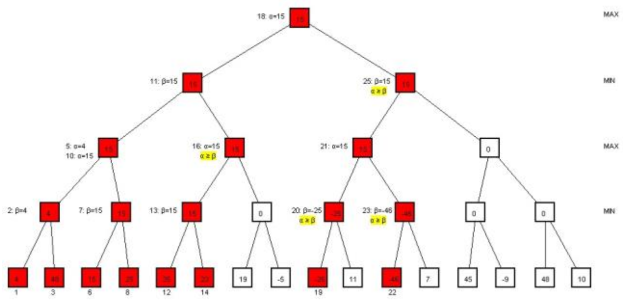


Figure 11. The alpha-beta pruning tree

In this algorithm, the calculateBoardValue(board) function calculates the value associated with the current board position. The FindKing(board, isRed) function locates the positions of the red and black kings on the board. By using the theBeforePosition(board, depth) function, a copy of the board state at a specific depth is created. The chess.PossibleMove(after) function generates the possible moves for a given chess piece in the current board state. The GenerateMoves(after, chess, x, y) function generates all feasible moves for a particular chess piece located at position (x, y). Lastly, the IsKingChecked(board, isRed) function checks whether the king of the specified color is currently in a check position.

5. EXPERIMENT

This section presents the usability testing, which was to evaluate the user satisfaction and effectiveness of AR Xiangqi. The testing and evaluation process included twelve (12) respondents, a sample size deemed sufficient given the straightforward nature of the application. There is no definitive optimal sample size for evaluating AR applications has been established [9], and prior research

[10] suggested a range of 10±2 respondents in usability testing.

Each respondent was given around 15 minutes to explore and engage with the game. Before the testing, all the respondent filled out a pre-experiment questionnaire, collecting details about their personal and demographic information and prior knowledge about AR. After the usability testing concluded, respondents received a post-experiment questionnaire. This part of the questionnaire concentrated on evaluating factors like efficiency and user interface, and comments or feedback about AR Xiangqi. The questions are outlined in Table V below. These questions mainly utilize a linear scale format, with the range starting from 1 (strongly disagree) to 5 (strongly agree).

TABLE V. MANOMOTION GESTURE INPUTS.

The game UI is clear and easy to interact with.
The plane detection before placing the chessboard is effective.
During gameplay, the chessboard placement can be seen.
The chess pieces can be differentiated easily by their colour and model.
The 'Click' gesture can be detected effectively.
The 'Pick' gesture can be detected effectively.
The 'Drop' gesture can be detected effectively.
The hand gesture can interact with the chess pieces effectively.
The chess pieces can be interacted with tapping on a smartphone screen effectively.
How much do you like the effect of the chess piece being destroyed in the game?
Do you think the opponent is hard to defeat?
Do you like the next mode suggestion?
Does the next move suggestion help you to play the game?
Which kind of interaction method would you choose after playing the game?

6. RESULTS AND DISCUSSION

Derived from the information gathered in the pre-experiment set of questionnaires, the respondent group consisted of 8 females and 4 males, with the majority of testers being 24 years old. Significantly, a substantial majority of participants reported having previous experience with AR, indicating their understanding of the concepts and technologies in AR. Most respondents who had experience with augmented reality described themselves as novices, implying a relatively recent initiation into the technology.

Regarding interaction by hand gesture, 83% of respondents mentioned having familiarity with interacting with objects in the game by hand gestures. Of all these respondents, 58% characterized themselves as novices in interaction by hand gesture. Only 8% of respondents said they had no previous involvement with either AR or interaction by gesture. In summary, the feedback indicates that most of the respondents had some degree of prior experience in AR, primarily as novices in both AR experience and interaction by hand gesture. These insights offer an important and valuable perspective in understanding their responses besides assessing the effectiveness of hand gesture interaction in AR Xiangqi.

The analysis of responses from the post-experiment survey yields imperative insights into the perception and experience of the respondents with AR Xiangqi. In general, all participants (100%) indicated agreement and contentment with the game UI, highlighting its user-friendly and entertaining nature. This validates the effectiveness of plane detection using handheld device cameras, as most of the respondents attested that the AR Xiangqi adeptly recognized the playing surface, enhancing the overall immersive experience.

Notably, 72% of respondents demonstrated a partiality for the original Chinese chess pieces model as indicated by their choices. This inclination suggests that the testers found the traditional chess pieces more appealing, and the inclusion of character assets enhanced overall enjoyment and participant engagement. Respondents' ratings indicated a generally satisfactory visibility of the chessboard, with 92% of testers rating it as 4 or above on a scale ranging from 3 to 5. This implies that the majority of participants found the visual depiction of the chessboard to be lucid and easily discernible. Moreover, 83% of respondents successfully distinguished chess pieces by colour, character or shape, emphasizing the effectiveness of the design and visual hints in assisting gameplay and distinguishing between the many chess pieces.

The assessment of interaction by hand gesture yielded optimistic responses, with 75%, 67%, and 66% of participants finding gestures such as "Click," "Pick," and "Drop" effective, respectively. This suggests that hand gesture interactions successfully executed game actions. Regarding tapping interaction, an overwhelming majority (approximately 91.7%) of respondents strongly agreed on its effectiveness. Additionally, 97% of participants expressed appreciation for the destruction effects accompanying the elimination of a chess piece. In terms of the perceived difficulty of the game, responses were evenly distributed, though a slightly higher percentage (66%) of participants regarded the game as either difficult or very difficult. This signifies that AR Xiangqi delivers not only engaging gameplay but also offers a challenging experience for the game players.

The feature suggesting the next best move garnered positive responses from 83% of participants, and a significant 92% of testers found it extremely beneficial for their gameplay. This implies that the suggestion feature contributed value and provided assistance to the overall gaming experience. In terms of preferred interaction methods, just over half, specifically 58% of respondents, preferred hand gestures, whereas the remaining participants favoured the conventional method of screen tapping. This suggests a leaning towards the immersive and intuitive qualities associated with interaction by a hand gesture in gameplay.

7. CONCLUSION

This work presents the AR Xiangqi, a traditional Chinese chess game as a markerless AR platform with AI searching using a minimax algorithm with alpha-beta pruning that gives suggestions to players of the next best move. Together with gesture-based play and immediate feedback for action confirmation, AR Xiangqi promotes an engaging UX gaming experience as it enhances gaming interactivity and creates an immersive gameplay experience. The usability testing reveals that testers are positively satisfied with the game UI, hand gesture interaction, next move suggestion and gameplay UX. Overall, the questionnaire results indicate that participants positively embraced the AR Chinese Chess game. Most respondents expressed contentment with the game UI, acknowledged the effectiveness of interaction by hand gesture, and valued features such as graphic effects for chess piece destruction and next-move cues. These results confirm that the game has effectively provided an engaging and pleasurable AR gaming experience, as respondents expressed contentment with many game elements of AR Xiangqi. For future works, employing AR Xiangqi on other platforms such as virtual reality like the work in [8] and implementing multiplayer functionality within the existing prototype would be an effective enhancement. This addition would significantly enhance player engagement and introduce a competitive element to the game. [9] has shown the engagement and usability of the chessboard which can be referred to as further improvement on Xiangqi Chess. For the interface, we can display AR Xiangqi in the form of a holographic display [10] and a more advanced technique, real hand gestures can be implemented in hologram [11]. In conclusion by incorporating gesture-based interaction and leveraging AI search algorithms for the game's opponent, AR Xiangqi enriches the user experience by offering immediate feedback, suggestions for the next move, and a truly immersive gameplay environment.

ACKNOWLEDGEMENT

We would like to extend our gratitude to Universiti Teknologi Malaysia for the facilities and experiments space.

REFERENCES

[1] Nguyen, R., Gouin-Vallerand, C., Amiri, M. (2023). Hand interaction designs in mixed and augmented reality head-mounted display: a scoping review and classification. Frontiers in Virtual Reality, Vol 4

[2] Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S., & MacIntyre, B. (2001). Recent advances in augmented reality. IEEE computer graphics and applications, 21(6), 34-47.

[3] Boonbrahm, P., Kaewrat, C., & Boonbrahm, S. (2019). Interactive marker-based augmented reality for CPR training. International Journal of Technology, 10(7), 1326-1334

[4] Teichrieb, V., Lima, J. P. S. M., Apolinário, E., Farias, T. S. M. C., Bueno, M., Kelner, J., & Santos, I. (2007). A survey of online monocular markerless augmented reality. International Journal of Modeling and Simulation for the Petroleum Industry, 1(1), 1-7.

[5] Ma, L. (2020). Xiangqi vs chess—The cultural differences reflected in Chinese and western games. Open Journal of Social Sciences, 8(03), 52.

[6] Manomotion, accessed 13th June 2023 https://www.manomotion.com/

[7] Mark Minasi. 1989. The minimax algorithm. AI Expert 4, 4 (April 1989), 15–22.

[8] Li W. SurfChessVR: Deploying chess game n parametric surface in virtual reality. (2023) 9th International Conference on Virtual Reality (pp. 171-178). IEEE.

[9] Yusof, C. S., Low, T. S., Ismail, A. W., & Sunar, M. S. (2019, November). Collaborative augmented reality for chess games in handheld devices. In 2019 IEEE Conference on Graphics and Media (GAME) (pp. 32-37). IEEE.

[10] Fadzli, F. E., Ismail, A. W., Rosman, M. F. A., Suaib, N. M., Rahim, M. S. M., & Ismail, I. (2020, November). Augmented reality battleship board game with holographic display. In IOP Conference Series: Materials Science and Engineering (Vol. 979, No. 1, p. 012013). IOP Publishing.

[11] Ismail, A. W., & Iman, M. A. (2023). Implementation of natural hand gestures in holograms for 3D object manipulation. Virtual Reality & Intelligent Hardware, 5(5), 439-450

Aida Ali is a senior lecturer at Universiti Teknologi Malaysia (UTM). Her primary research focus. She is a researcher in the ViCubeLab research group at UTM. She obtained her PhD from UTM and her research interests began by exploring hybrid fuzzy classifiers for class imbalance, and particle swarm optimization. Her recent research findings are in a 3D virtual park for learning, virtual reality hand gesture interaction, also location-based augmented reality (AR) for virtual event experience.

Ajune Wanis Ismail is a senior lecturer at Johor Universiti Teknologi Malaysia (UTM). She earned her B.Sc degree in computer graphics and computer vision and her M.Sc. and Ph.D. degrees from UTM, Johore, Malaysia. Her current research interest includes mixed/augmented reality. She is currently the Head of the Mixed and Virtual Environment Research Laboratory (Mivielab), UTM, where she has also been appointed as the head of the ViCubeLab research group. Her research interests include hand gesture interaction and cross-reality environments.