Shared Gaming Object

By,
Jeffrey Bauer, Niyati Shah, Tanvi Raut, Mitayshh Dhaggai

1. Introduction

Shared gaming object works on the idea that multiple people not present at the same location can play a game together, not behind a computer screen but in a 3D world. With this concept we want to show that this idea can be extended beyond just the world of games. We can also use such a system to work around other real-world objects, like hardware and machinery. These objects can be displayed in an augmented reality environment and can then be displayed and manipulated by multiple users. This can be applied in many different industries to make collaboration easier.

As part of the project we have created an augmented reality game of chess which can be played between players playing at different locations using the Microsoft Hololens where the game state is shared between the two players. The moves played by one player are reflected immediately on the game accessed by the other player.

2. System Overview

Instead of creating our own chess pieces we have used the pieces modeled by Dmytro Fedorenko from grabCad [1] to help us with the game. We implemented our own chess board by using 64 box objects as tiles. We added materials/texture to both the pieces and the tiles. The tiles are colored black and white with a marble texture. The pieces have a wooden texture, with the shade of the wood depending on the team that the piece belongs to.

(a) Marble texture

Figure 1: (b) Wood texture

Properties

The tiles are give the following properties:

Property /Variables	Type
Tile ID	Integer
Row	Integer
Column	Integer
isEmpty	Boolean
childPiece	Game Object
OriginalColor	Color

The pieces are given the following properties

Property / Variables	Type
Piece ID	Integer
Row	Integer
Column	Integer
Type	Enum type
Team	Integer
OriginalColor	Color
ParentTile	Game Object

Major components:

The major classes and their methods are given as below.

Classes:

Classes
ChessBoardManager
ChessRules
PieceProperties
TileProperties
TileIAction
ChessPieceIAction

ChessBoardManager:

This script manages the chess game. It initializes the game and contains all the major methods that are used to manipulate the pieces on the board. The script contains the given below methods.

ChessBoardManager
Start & update
GetLocalUser
CreateBoard
FindPiece & GetPiece
FindTile & GetTile
SelectPosition & SelectPiece
getAvailableMove & MovePiece
ManipulateBoardStarted, ManipulateBoardCompleted & SetTransparency

PieceProperties and TileProperties

These two scripts do not contain any methods; only properties that define the particular piece or tile.

ChessRules

This class contains the rules that are applied to the game. It looks for all valid moves for any given piece. It is a very basic implementation, supporting only the basic piece moves. It does not support castling, En Passant, check, or checkmate. The methods in the class are as follows:

ChessRules
GetAvailableMoves
FindAndGlow
FindAndGlowAll
ClearGlow
AddOrthogonal
AddDiagonal
AddL
IsInCheck

ChessPieceIAction

This class contains all the actions that are required to manipulate the chess pieces in the game. The methods in the class are as follows:

ChessPieceIAction
Start & update
ActionGazeEntered & ActionGazeExited
ActionOnSelect
SetManager
SetGlow

TileIAction:

This class contains all the actions that are required to manipulate the tiles that make the chessboard of the game. The methods in the class are as follows

TileIAction
Start
ActionGazeEntered & ActionGazeExited
ActionOnSelect
IsGlowing
SetGlow

Inputs / Outputs

The input to the system are gaze and tap.

Gaze:

Gaze works by staring at an object(piece/tile). The gaze gesture is implemented by using the gaze cursor and the methods called by ActionGazeEntered and ActionGazeExited.
We have implemented a cursor that tracks the user's head movement, when this object collides with an piece or a tile, it gets selected by the ActionGazeEntered method. When you move away from the piece or the tile, it is deselected by the ActionGazeExited method.

Tap

The tap gesture works by tapping your index finger on your thumb in front of the hololens camera's view. It uses the ActionOnSelect method to select the pieces or tiles. To move the board, we have added a delay this method which selects the board and once we hold the tap, we can than drag the selected board.

The output of the system:

On gazing at a piece or a tile the piece or the tile will glow using the setGlow method which is set to true when the ActionGazeEntered is true. And it stops glowing when the ActionGazeExited becomes false turning the setGlow to false.

On tapping the piece, the piece is selected and it will rise above the board on its place. On tapping the tile, if a piece is already selected than the piece is moved to that location, if no piece is selected then nothing happens. On using the tap and hold gesture the entire board is selected and can be dragged to any other position by keeping the hold on.

Prefabs:

Prefabs are asset type that let you store game objects in them with all their components and properties. They are like template of objects that are stored in them so you do not need to write their basic functions in all the objects of that type individually. We have made the cursor, hand detection, sharing object and select objects structures as prefabs. The game objects like the chessboard and chess pieces are also made as prefabs to help us work efficiently. If we have many of the same objects like the rook, creating a prefab of type pawn makes changing it very easy. As each of the pawns have the same kind of property, if we need to change certain property we can easily do so by just changing that property in the prefab and it would immediately be reflected in all the pawns.[2]

3. Implementation

Language: C sharp
Libraries: We uses the built in networking and collection libraries and the Holo tool kit.

System Requirements:

The basic system system requirements to run this software are
Microsoft Hololens
64-bit Windows 10 Pro
64-bit CPU
CPU with 4 cores (or multiple CPU's with a total of 4 cores)
8 GB of RAM or more
The following features on your system must be supported and enabled:
- Hardware-assisted virtualization
- Second Level Address Translation (SLAT)
- Hardware-based Data Execution Prevention (DEP)

The software needed are
- Visual Studio 2015 Update 3
- HoloLens Emulator (build 10.0.14393.0)
- Unity for hololens 5.4.3

To run the emulator fast a GPU is required without it the emulator is very slow.
- DirectX 11.0 or later
- WDDM 1.2 driver or later

Implementation:

When the game starts, the whole chessboard object is created and initialized. All the tiles who have pieces on them update their childPiece property as the piece on it and change the isEmpty property to false.
The pieces update their property of team, row, column, parentTile and OriginalColor based on the team they are on. This is updated in the createBoard method of ChessManager.

On selecting a piece it extracts all the properties of that piece and marks its available moves by glowing the tiles it can move to in red. This is done in the chessRules. Next when we select an available tile, It extracts all the properties of that tile. It will clear the available moves glow on the board. Then it moves the piece to that tile. If there is a piece belonging to the opposite team already present on that tile then that piece will be removed. Now for the selected piece, the original parentTile is selected to NULL, the properties of the piece, row, column and parentTile, are then updated according to the new position. And then the new tile where the piece has moved will change its childPiece to the selected piece.

4. Challenges

Most of us did not have experience with the Unity software, so our first challenge was to understand how it worked and experiment with it. Once we played around with a few objects and could manipulate them on the HoloLens, we were much more comfortable using it.

The next challenge we faced was getting the networking between the Hololens' working. We had to first decide how we would share the information about the game, how can we would control the objects in consideration and how the sharing work (should it be synchronous or asynchronous). We wanted to make sure that two people cannot play at the same time and how the moves are reflected; is only the move reflected or can the other user see when the player selects a piece and what are his options for valid moves.

We decided on the basic idea of how our networking would work and then, with the aid of the Microsoft Holographic Academy, we were able to implementation our ideas. The basic idea is that we have the move made by a player sent to all other players in the game. Each player would maintain their own board and update it with the new move made.

The last challenge we faced was implementing the rules of the game. As chess is a very complex board game where different pieces have their own moves on the board and there are conditions are so many other conditions in which one can end the game do other things like promote pieces. And implementing those rules for a virtual board has its own challenges as in reality we just know the rules, but we need to put in those rules for each piece. Also as it was a virtual game we needed to make sure that two players don't play at the same time, they cannot play each others players and they have their own turns.

5. Results

The final output for the project is a chess game which can be played between two people not in the same location. Given below are the steps and screenshots of the game being played.

The following is a video of the game near its completion:
https://www.youtube.com/watch?v=ZMNfGAd-KR8

The code for the game can be found at:
https://github.com/tanvi2504/AugmentedReality_Chess

Using the bloom gesture, start up the main menu.

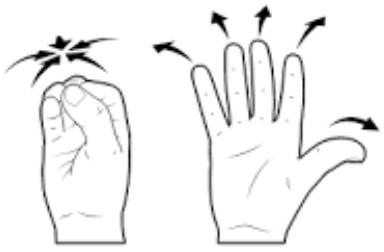
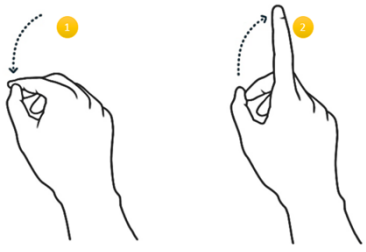
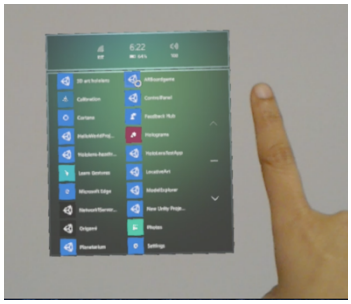


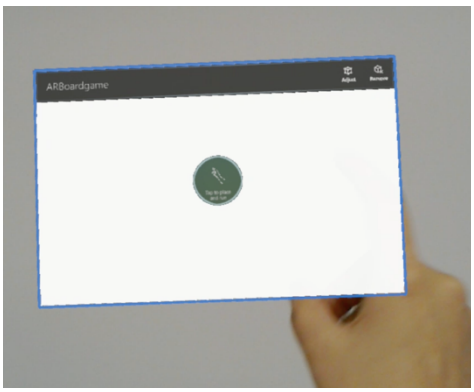
Figure: Bloom Gesture

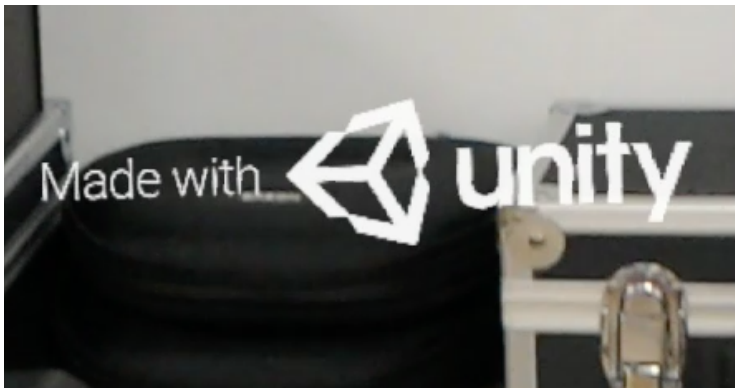
Select the '+' sign in the menu and click on the ARBoardgame from the given list.



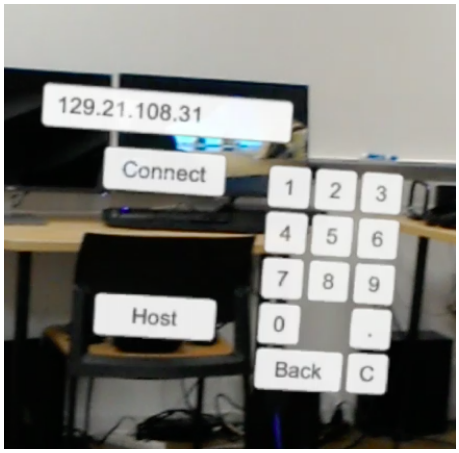


Tap on the window to start game.

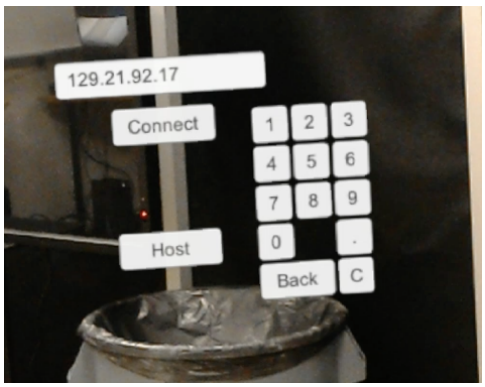




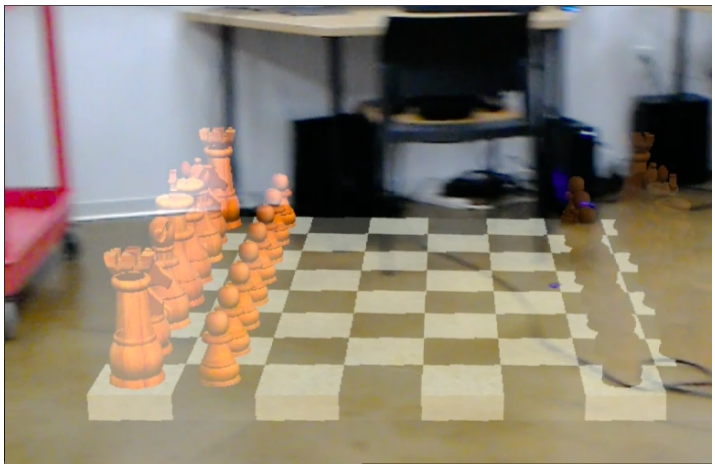
Once the game starts you see a menu.
If you are going to be the host, click the ‘host’ button.



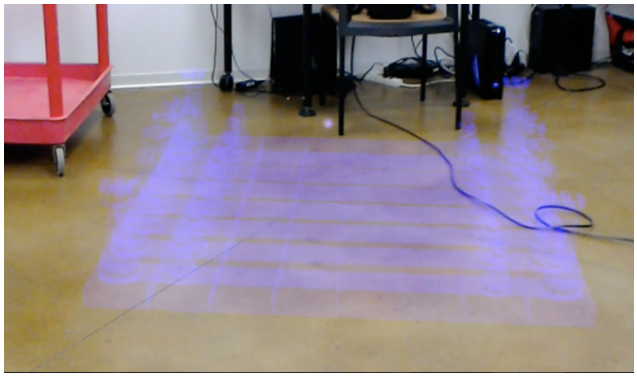
If you are the client/non host player, type in the IP address of the host and then click on the ‘connect’ button.



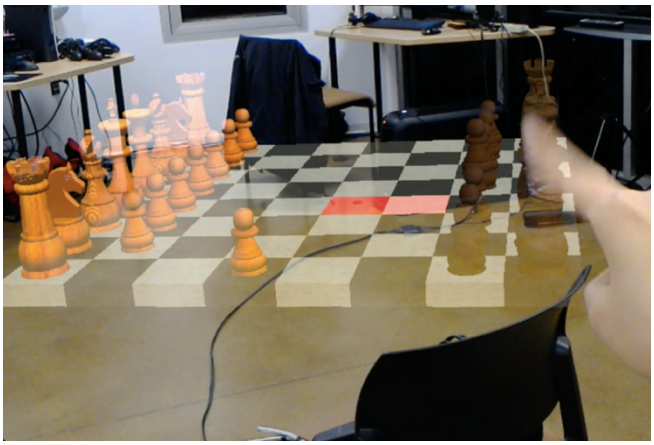
Once connected, both the players will be able to see the chessboard in front of them.
Note that there are no re-sync capabilities, so if a move is made before all of the clients connect or if a player loses connection then the game must be reset for everyone.



To move the position of the chess board, use the tap and hold gesture to select the board.
Once the board highlights in a blue color the board can be moved anywhere by holding the tap gesture and dragging it to the desired place.



To move a piece, gaze at the desired piece until it glows and then use the tap gesture to select it. Once selected, it will rise over the board and other pieces. Then gaze at the desired tile among the available moves tile till it glows. The available moves will glow in red. Use the tap gesture on the tile to select the tile and the piece will be shifted to that tile.



This move is reflected on the boards for both players immediately. The other player can than follow the same step and move a desired piece from the opposite colored piece.

To capture a piece, use the tap gesture to select the piece or the tile under that piece.

Note: The HoloLens displays the color black as transparent. The images above were taken using the HoloLens HUD, which can be accessed by navigating to the HoloLens' IP Address in an internet browser. The HUD increases the transparency of all objects when it is recording what is being seen on the HoloLens, so the images are a little hard to see. When wearing the HoloLens however, the images appear much more opaque.

6. Conclusion

We have completed the game enough so that two players can easily play a game of chess with the basic rules applied. We still need to complete the rules that would make it a perfect game of chess, including the rules for ending the game. That means we need to add conditions of checkmate, resigning and draws. We can also add a time control to have a proper competition between people. Rules like En passant and pawn promotion are also required to be added in the game. Just like in competitions we can add move recording to make the game just like in competitions.

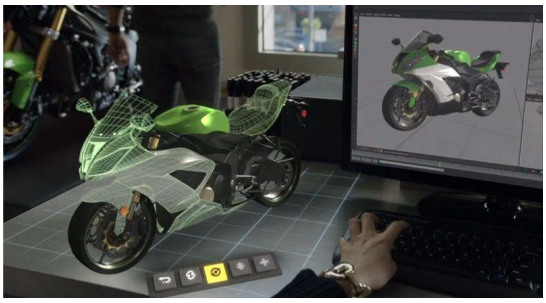
Apart from rules, we can also add new functionalities in the game that can make it more like the one in real world. We can include things like an undo and redo move, we can have the games state saved so that it can be played later. Right now we need to restart the whole game to play a new game, but this can be done by just clicking a button, which would then make it reset the game. We have the functionality to move the board from one place to another, but we cannot resize it, so we could also add that functionality.

To make the game more interesting we can add sound effects and animation to the pieces (like seen in Harry Potter).
By adding another row and column with numbers (1-8) and letters (a-h) to the board, it would help identify the row and column positions in the board. Then we could use this information to apply voice commands build in hololens to play game instead of selecting tiles.

As this game is being played by two people not in the same location, it becomes difficult to play the game without any communication as we would not know if the other player has left or the connection is bad, or who starts etc. So adding some kind of communication channel between the players would make it more efficient to play the game. We can also improve the menu of the game and new features in it so that it is more intractable. As part of stretch goals we can add more games that can be played by more than two players using the hololens.

7. Future work

As part of future work, we can implement many other games that can be played by not just two but multiple players around the world. This concept can also be extended to include not just games, but hardware or other devices. Industries that work on hardware or machinery development can collaborate together to make them. Also this concept can also be used in the health/medicine industry to work on to look at the human structure or a protein structure. It could also be used by the architecture industry to create models of buildings and structures. Below are some of the usage of the shared objects on hololens. [4]




Examples of automobile and machinery industry [7][6][5]

Examples of architecture industry [8]



Examples of health and medicine industry.[9][10]

8. References

[1] https://grabcad.com/library/chess-74
[2] https://docs.unity3d.com/Manual/
[3] https://www.wired.com/2015/12/hands-on-with-hololens-groundbreaking-potential-for-ar/
[4] https://www.cnet.com/news/microsoft-jumps-in-augmented-reality-with-hololens/
[5]http://www.geekwire.com/2016/nfl-meets-hololens-heres-microsofts-vision-watching-sports-augmented-reality/
[6] http://www.digitalcare.org/what-is-augmented-reality-hololens/
[7] http://www.popsci.com/microsoft-and-volvo-bring-augmented-reality-to-car-shopping
[8] http://www.gamezone.com/news/microsoft-reveals-augmented-reality-headset-hololens-at-windows-10-event-3412301
[9] http://www.ibtimes.co.uk/microsoft-hololens-company-reveals-what-it-feels-like-use-augmented-reality-helmet-1510256
[10] https://developer.microsoft.com/en-us/windows/holographic/academy