Connect Four: An Augmented Reality Game

Markus Seiberl*
University of Applied Sciences Upper Austria

ABSTRACT

Trends show that augmented reality is becoming increasingly popular. Lumus [1] estimates that by 2025 the total revenue is exceeding $25B with the gaming industry leading at $11.6B. With more than 1.05 billion AR-enabled smartphones on the market [3], augmented reality is more accessible than ever before. Therefore this paper describes a possible way for implementing the connection board game Connect Four as an augmented reality game on a smartphone.

1 INTRODUCTION

The monthly active users of smartphones with a mobile AR installed base grew from 47 million to 150 million users from December 2017 to May 2019 [3]. As future trends tend towards augmented reality shopping or navigation, the yearly revenue is lead by the video games section.

Besides the native approach with ARCore and ARKit, Unity 3D in conjunction with Vuforia offer a way to create an augmented reality application for both platforms, Android and iOS. As one common code base for both platforms is preferable for developers, this paper researches the augmented reality game development with Unity 3D and Vuforia. As an example the Connect Four game is taken.

Connect Four is a connected board game, in which 2 players play against each other. Every player chooses a colour and let alternately a coin of their chosen colour drop in one of 7 possible columns of a 6 by 7 suspended grid. The first player who reaches four coins in a row either horizontally, vertically or diagonally wins the game.

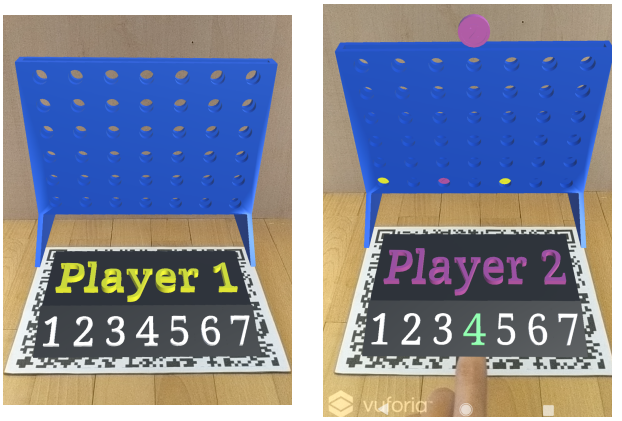
2 GAMEPLAY

We intended this game to be played either at a table or on the floor. The optimal distance between marker and smartphone for a Google Pixel 2 is about half a meter. As new smartphones get equipped with better cameras like wide-angle cameras, the optimal distance is likely to be reduced.

After launching the game the camera view is shown. As soon as the Vuforia library detects a specific marker the gaming field is rendered on top of it, as seen in figure 1a. Player 1 and 2 can be differentiated in their colours. Yellow for player 1 and purple for player 2.

Player 1 makes the first move. The player can select the column in which the coin should drop by hovering an object like a finger over the number of the specific column. Subsequently, a coin for indication appears on top of the grid and the number's colour turns into a light green. At this point, the player can still move the finger over another number to select a different column. This state is depicted in figure 1b. If more than one number is selected, the colour of the selected numbers turns red indicating an error.

*e-mail: markus.seiberl@fh-hagenberg.at



(a) Gaming field is placed on top of the recognised marker.

(b) Player 2 selected column 4 but can still change it.

Figure 1: Screenshots of the Connect Four game.

To let the coin drop, the player needs to hover the finger over a number for one second. After the one second, visual and auditory feedback is given to the user. The auditory feedback consists of a short bell ring [2]. The selected column number now appears in a strong dark green. We introduced a cool-down of half a second to counter-fighting an accidental selection immediately after dropping the coin. The text between the column numbers and the grid indicates which player's turn it is and changes as soon as the coin gets dropped.

As soon as one player archives 4 coins in a row either horizontally, vertically or diagonally, the player wins. Thus the game ends. As visual indication on the left and right side of the grid appear confetti spray cannons celebrating the win of the player. Additionally, the text "*Player <player number> WINS!*" pops up above the grid. The acoustical feedback includes a 4 seconds long fanfare [1]. As players possibly want a rematch, a Snackbar shows up at the bottom of the screen, asking the user to play again. A conceivable end of the game is depicted in figure 2.

3 IMPLEMENTATION

For developing the Connect Four game many components have to play together. As game engine, we have chosen Unity. The interactions and game logic is handled with C# scripts via the C# scripting API. We used the Vuforia SDK to enable the creation of augmented reality applications in Unity. The tracking of the marker is also handled via the Vuforia SDK.

3.1 3D Objects

Before the actual development, the 3D game objects were created. For designing the objects we used the 3D modelling program Tinkercad. It runs in the browser and is fairly simple to use. Despite its simplicity, it was sufficient as only cubes, cylinders and wedges were combined with add and cut operations. The export as *.obj* files

[1] Sound effect obtained from `https://www.zapsplat.com`

Figure 2: Player 1 wins as he got 4 coins in a row.

made it easy to import the created 3D models into the Unity project.

As the Instantiating of game objects at runtime is very costly in terms of CPU time, we decided to instantiate all coins at startup. This ensures a smooth experience for the user and won't lead to fps drops while creating a coin. The grid has 6 rows and 7 columns and therefore the coin count for each colour is 21.

3.2 Interaction

The interaction in the Connect Four game is state-dependent. During gameplay, the AR game makes use of Virtual Buttons. Virtual Buttons are placed directly on a Vuforia Target to make them interactive. Connect Four implements seven of these Virtual Buttons for the player to select the column. All the player has to do, is to move an object like a finger to the number. As soon as enough of the target where Virtual Button is placed is obscured the "*OnButtonPressed*" event is fired.

As the build-in threshold for firing the events "*OnButtonPressed*" and "*OnButtonReleased*" can be crossed fairly often an own algorithm was additionally used to determine when to let the coin drop down into the grid. The user needs to select a Virtual Button for a certain time to let the coin drop. It turned out that one second is the optimal duration.

The second interaction element is a UI Button "Reset" on the right side of the Snackbar which pops up after one player won. By clicking this button, the game resets to the initial state. All coins, the confetti canons and the "*Player <player number> WINS!*" text get disabled. Also, the next turn text resets to "Player 1".

3.3 Game Logic

Every time a player drops a coin into the grid, an algorithm has to check whether or not the player got four of his coins in a row. To simplify this algorithm, a multidimensional array with the size of the grid (6x7) is initialised at startup. Depending on which player is dropping the coin, either the number "1" for Player 1 or the number "2" for Player 2 is written into the array. The rows are logically inverted so that the 6^{th} row represents the lowest row of the grid. Listing 1 shows a code snippet of the algorithm determining whether or not the current player archived 4 coins in a row. The algorithm determines if a player won, by first checking the horizontal axis, the vertical axis and then diagonally.

Listing 1: Algorithm to check if four coins of the same player are aligned horizontally.

```
public bool ConnectedFour(int row,
                          int column,
                          int playerNr) {
    if (CheckHorizonal(row, column, playerNr) ||
        CheckVertical(row, column, playerNr) ||
        CheckDiagonal(row, column, playerNr)) {

        return true;
    }
    return false;
}

private bool CheckHorizonal(int row,
                            int column,
                            int playerNr) {
    var left =
        CountLeftSide(row, column, playerNr);
    var right =
        CountRightSide(row, column, playerNr);
    if ((left + 1 + right) >= 4) {
        return true;
    }
    return false;
}

private int CountLeftSide(int row,
                          int column,
                          int playerNr) {
    var count = -1;
    for (int col = column; col >= 0; col--) {
        if (gameField[row, col] != playerNr) {
            return count;
        }
        count += 1;
    }
    return count;
}

private int CountRightSide(int row,
                           int column,
                           int playerNr) {
    var count = -1;
    for (int col = column; col < 7; col++) {
        if (gameField[row, col] != playerNr) {
            return count;
        }
        count += 1;
    }
    return count;
}
```

3.4 Rigidbody

To let the coin drop into the grid, the property ”Kinematic” of the Rigidbody component is set to *false*. The coin then collides with the grid or a coin if already coins are in the column. Problems arise when Vuforia loses track on the marker. Because gravity still affects the coins and the grid is not anymore rendered, the coins simply fall through. By registering an EventHandler on the TrackableBehaviour of the image target, it is possible to receive events when the state of the TackableBehaviour changes. As soon as the state changes to ”NO_POSE” the property ”Kinematic” gets enabled for every coin. The side effect is, that whenever Vuforia loses track of the marker while a coin is falling down, the coin stops at that specific position and only starts falling down again if Vuforia is tracking the marker again.

4 Future Work

A settings page where the user can set tune different parameters like the sensitivity of the virtual buttons or the colour of the players would enhance the user experience. Additionally, a start screen can be added. Performance-wise, the coins have room for improvement. Currently, they make use of the ”Mesh Collider” component for detecting collisions.

References

[1] Augmented reality trends, 2018. Accessed at 01.01.2021.

[2] InspectorJ. Bike, bell ding, single, 01-01.wav, Sept. 2019.

[3] A. Makarov. 9 augmented reality trends to watch in 2020, 2020. Accessed on 01.01.2021.