Gesture Remote: An interactive multiplayer gaming toolkit using smartphone as a controller

Aadesh Bagmar, Ameya Patil, Deeksha Dixit, Preston Tong*
aadesh,ameyap,deeksha,ptong2@umd.edu
University of Maryland
College Park, Maryland

ABSTRACT

Interactive multiplayer gaming has always been an active area of development. However, most games rely on external devices for input, such as keyboard, mouse, PS4 controllers etc. These devices support a limited set of actions including button presses or joystick movements, and are often lacking motion sensing capabilities. In addition to this, most of these devices are expensive ($50-$60). Smartphones on the other hand are ubiquitous and consist of a rich set of sensors that let users control and interact with the environment. In our research, we show how a smartphone can be used as a low-latency input device for playing real-time multiplayer games. We show that information from smartphone sensors can be used to do basic motion sensing. Our method allows users to perform natural actions, such as steering a wheel, as opposed to key presses while playing racing games in order to provide a more immersive gaming experience.

KEYWORDS

Smartphones, Ubiquitous Computing, Multiplayer Gaming, Sensors for Motion Detection

ACM Reference Format:
Aadesh Bagmar, Ameya Patil, Deeksha Dixit, Preston Tong. 2020. Gesture Remote: An interactive multiplayer gaming toolkit using smartphone as a controller. In *Proceedings of SIGCHI (MD '20).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/n

1 INTRODUCTION

The Coronavirus Pandemic has brought life to a standstill. Most of the world has asked citizens to self-isolate in hopes of combating the pandemic. Bored and restless, people are taking solace in online games. Game sales are breaking franchise records [15][16] while Twitch has exceeded 3 billion hours of watch time for the first time [17]. The gaming industry is thus growing at a much faster pace than before.

Most games are played on the desktop and use keyboard and mouse as input devices where users often need to perform unnatural actions while playing games, e.g. arrow keys to drive a car. While VR has increased game satisfaction [19], adding motion sensing capabilities has made gaming more immersive [18]. However, such controllers are often extremely expensive.

Smartphones have become ubiquitous and are packed with rich sensors. It is possible to exploit data from these sensors to convert it into a low-latency motion sensing game controller. Research in the past [1, 12, 13], has also shown that technology which does not alter human habits / actions and embeds into daily routines is more immersive by nature. The challenge we gave ourselves was: Can we use our mobile phone as an input device for playing real-time multiplayer games? Can we detect a user's motion accurately and with low latency so that we allow them to play games with natural actions?

In this paper, we report several design iterations that we went through to develop a general purpose system. We also demonstrate several real-time multiplayer games where our system can be used for an immersive experience. Lastly, we provide details from a user survey that we conducted which discusses the usability, immersiveness, and engagement of our apparatus and how the feedback changed our system design.

The paper is organized as follows. We start with related work Section 2 where we discuss existing research ideas and fully developed applications and products similar to our tool. Section 3 discusses the overall system design, implementation details and explains the working of our tool. In Section 4, we discuss the pros and cons of our idea. Finally, we conclude with probable future directions in Section 5. Our survey results are attached in Appendix A.

*All authors contributed equally to this research.

MD '20, May 16, 2018, College Park, MD

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/n

2 RELATED WORK

In 2008, a group of researchers demonstrated the use of Nokia phones which can be used as wii-like controllers [20] to play games on large screens. They used accelerometer data to track motion and used Bluetooth to connect to the server. Their experiments were done on low quality games that they built in house. They also suggest that novel interaction mechanism coupled with a fun group activity can provide an enjoyable social experience, with high levels of user interaction, hence, laying down the primary motivation for our work.

A lot of work has been done in understanding the benefits of full body motion based game control, especially for elder people who need to get some physical activity for their well-being. The studies performed in [5] and [6] show how playing full body based motion games results in a positive effect on the mood of the people while allowing them to exercise. Given the current scenario of the pandemic, where everyone is trying their best to stay safe by staying home, the condition of people of all ages is not much different from that of the elder people. It would thus be of great help to have a full body motion based game control to cheer people up in this tough time while allowing them to have physical activity. Another study performed by Norton et.al. [12] discusses guidelines for mapping human actions or gestures to events in the game. This study consisted of participants who were asked to play video games by choosing actions/gestures which they thought were more intuitive to them, to cause the desired event in the game. We tried to build the actions/gestures supported in our app, as close to the real life experience of playing the game, while keeping in mind the findings from this study.

Design of tangible game controllers with haptic feedback is also an active area of research. Foottit et. al. used an Arduino controlled haptic glove [3] as an interaction device for a virtual flight controller. They discuss how natural movements for controlling virtual environments are more immersive and suggested that users adapt promptly to intuitive gestures. Consequently, use of a tangible controller enabled the users to master virtual interactions as a significantly faster rate.

Leap Motion is a free-hand gesture-based controller introduced in 2013 [13]. Leap Motion was primarily designed for productivity, however, was later redesigned into Orion, which is an offshoot meant for VR [13]. Pirker et al designed a user study to compare Leap Motion with the standard keyboard and mouse [13]. They discovered that users valued the accuracy of keyboard and mouse over Leap Motion, but overall found Leap Motion to be a more fun user experience [13]. Leap Motion is a big source of inspiration for our gesture remote. We wanted to design a more interesting and innovative experience that could replace the mundane use of keyboard and mouse. Although the gesture remote may not be as accurate as keyboard and mouse, we wished to inject a novel twist to game play.

Makey Makey [2] is a tool for improvising tangible and nature-based user interfaces. It allows any physical object to be used as an input device for the keyboard by just connecting wires and making use of electrical conductivity of human body. Our idea similarly draws inspiration for replacing the keyboard input with our gestures. Like Makey Makey, we avoid making changes in the game code itself and instead, build a side channel to use tangible devices to play games.

The Wifi Mouse is an Android application developed to simulate a phone as a wireless keyboard, mouse, and/or trackpad for the computer [11]. Our gesture remote is similarly designed to act as wireless input. However, instead of the standard keyboard and mouse, our software maps gestures to keybindings in addition to supporting shaking. The Wifi Mouse has been found to contain "spam and bloatware" and requires unnecessary permissions like "storage access & file modification" [11]. On the other hand, our app contains only essential code and requires only necessary permissions, such as camera permission to scan barcodes. No data is collected and stored on our end.

Remote Mouse [10] is another app which also serves as a remote control for operating a computer. While the app does support remote keyboard and mouse, it does not facilitate custom keybindings. It has the controls for most common computer apps VLC, Power Point and Windows Photo Viewer hard coded as presets. This is where our app provides more flexibility to the user in terms of mapping any gesture to any action or keyboard or mouse input. These mappings can be changed in just 3-4 steps.

Mobile Gamepad [4] is an android app that provides a multitouch gmepad for PC games. It is primarily designed for taking gesture inputs for computer games. This app provides custom keypad for supported games. However, Gesture Remote is different than Gamepad as it allow users to play any game or do other generic keyboard controlled tasks with custom input mapping which is not supported in Gamepad.

3 SYSTEM / PROJECT

System Overview

Our system consists of two major components. Both of these components are software based and they communicate over WiFi using User Datagram Protocol (UDP).

Software Design

Our system is designed as a client server model where the laptop where the game runs hosts a server to which the phone connects to. The connection uses UDP to reduce latency. We use information from a smartphone's IMU sensors.

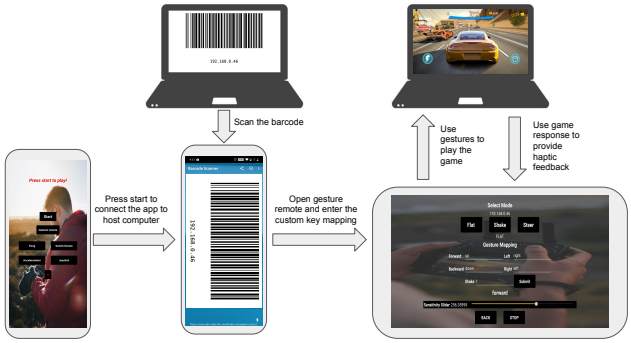

Figure 1: Flow diagram

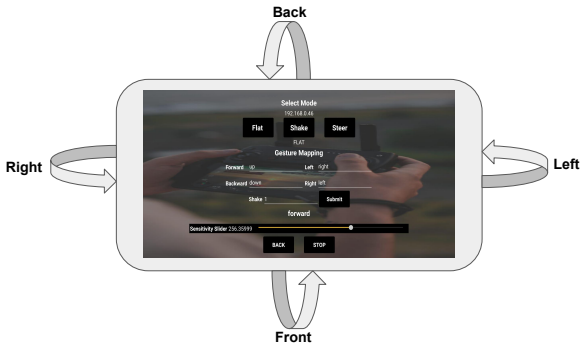

Figure 2: Gestures in Flat mode

We used the MIT App Inventor IDE [14] for designing our client side application.

Design Choice. We had to make a design choice between sending all the sensor data to the server and processing it vs identifying the action at the client side itself and sending it to the server. experimented with both and went ahead with the second option. There were three main reasons for doing this. Firstly, network transfer is the largest bottleneck for us. Transferring more data would increase the latency. Secondly, smartphones are becoming better in their processing power. This allows quick computation at the client side without affecting it's performance by a lot. Third, this distributed arrangement is much more efficient since it alleviates the server from processing data coming from multiple users at the same time. The next section explains our software setup and details out functionality of our client side and server side mobile application.

Software Setup

To connect the gesture remote app to the host computer through UDP the python server running on the host computer generates a barcode representing the IP address of the host computer. Only once the barcode is scanned and processed the app allows the user to access the Gesture Remote features. The overall flow of our system setup is shown in Figure [1].

Client-side mobile app. The users have the flexibility to hold the phones in two different modes. Flat mode in which the screen faces upwards and Steering mode in which the screen faces the user. The app provides 8 different gestures over these two modes which can be mapped to any key input. A separate shake gesture is also provided in Shake mode. A description of these gestures is provided below.

Flat mode gestures: The neutral position in Flat mode is keeping the phone flat screen facing up. All the gestures are detected based on this neutral position.

- Front : upper edge pressed down
- Back : lower edge pressed down
- Right : right edge pressed down
- Left : left edge pressed down

Steer mode gestures: The neutral position in Steer mode is when the phone is held straight parallel to the body screen facing the user. No action is detected in the neutral mode.

- Front : upper edge pushed away and lower edge pulled towards the user
- Back : upper edge pushed towards and lower edge pushed away from the user
- Right : turn phone clockwise from the neutral position
- Left : turn phone counter-clockwise from neutral position

A visualization of gestures in the Flat mode is shown in Figure [2]. As described in section 3 the processing of the gestures take places on the app itself. In steering mode the the Front and Back actions are identified based on the Z-axis tilt of the Accelerometer while the Right and Left actions are detected based on the Y-axis tilt values. Similarly, in Flat mode X-axis tilt values are used for Forward and Back actions. While, Right and Left utilizes the Y-axis tilt values. In the Shake mode a shake is detected if acceleration is changed in any of the X, Y or Z dimensions.

Based on the user feedback we got during our survey we also included a sensitivity slider in the app. This sensitivity slider can be used to adjust the frequency at which the gestures are detected. The slider positions determines the Accelerometer timeout between detection of two consecutive gestures. This, enables the users to adjust the app's response time according to the application at hand. For instance, a slower response time is needed while using the app for scrolling while web browsing, while gaming warrants a faster response time.

Once the gestures are detected by the app they are mapped to the text strings provided by the user for each action in the gesture mapping section of the app. These strings are

then sent to the IP Address retrieved from the barcode in the form of a UDP datagram. A stop button is provided to stop the transmission of any data from the app.

Party Box server-side application. Our host side application is a simple python UDP server created using Python socket library [9]. We used the pynput [8] library to simulate the key presses on the keyboard of the host computer. The server side application also contains the functionality to convert the IP Address of the server into a barcode for easy input of the host address in the app. This barcode is generated using the barcode [7] library. We also created a key mapping dictionary on the server which contains the mapping all probable key string values which the user can input to pynput key press commands. For instance string 'space' is used as the key associated with the command pynput.keboard.Key.space. Finally, we bind the TCP/IP socket to a port and then wait for messages from the app. Once, a message is read from the socket the server uses the string in the datagram as a key and executes the corresponding key press value from the key mapping dictionary.

We decided to use UDP as our communication protocol between the app and the host computer because it is a message oriented protocol which sends the messages in a continuous stream without waiting for a confirmation from the receiver. This makes it faster than other network protocols such as TCP. UDP is also simple to implement as it does not require any long lived connection between the server and client.

Party Box Games

In-house games. Given our newly made gesture remote, we decided to create our own custom games to demonstrate the compatibility of the gesture remote with multiplayer games. We have designed two in-house games: a two player pong game shown in Figure 3 and a four player midi party rhythm game shown in Figure 4. Both are hosted on Heroku and can be found at https://pongparty.herokuapp.com/ and https://midiparty.herokuapp.com/. To play pong, a phone can be set up to move the bar left and right while following the players actions with the phone. For midi party, a midi track is taken as input and the song is split into tracks that are streamed across the screen. Each bar represents a note and each color is for a different player. The goal is to have players perform an action on beat for the notes in order to hear it. So if all players perform an action correctly, then the full song can be gradually heard. Different actions can be designated per player. So when a player performs that action, such as shaking the phone or tilting it forward, then the designated note will be played. Right now we have not integrated haptic feedback for these games, but we plan to. For example, for Pong Party if the ball bypasses the player bar, then the phone

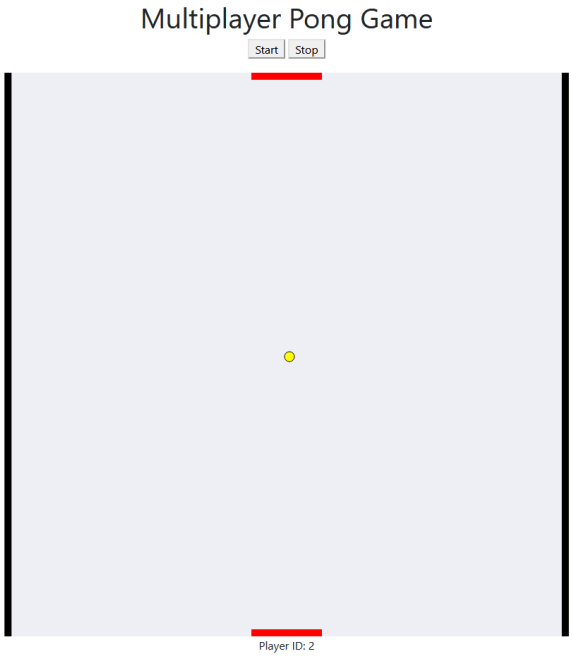

Figure 3: Pong Party

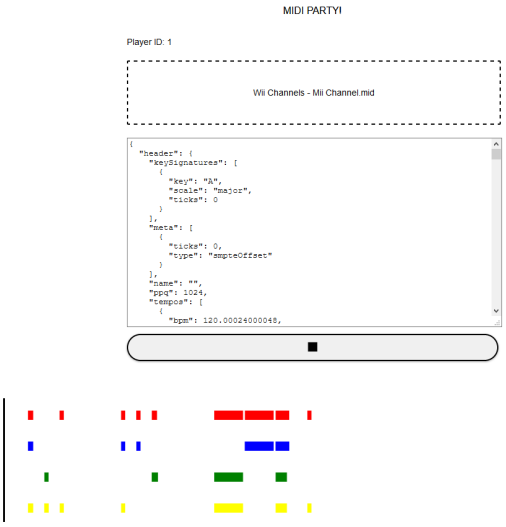

Figure 4: Midi Party

can vibrate signaling the opponent scored a point. For Midi Party, the phone can vibrate if a note has been correctly hit.

Third Party Games. In addition to the custom games that we built, we also tried to use our app to control third party games to gauge the efficacy of our app in terms of universality of use. For this, we randomly chose games over the internet which had any scope of mapping game inputs or events to intuitive

natural actions or gestures. We then mapped appropriate gestures to those inputs and tried playing the game. Some examples of games played are shown in Figure 5 and the respective gesture mappings are described below:

- **Google Doodle Game - Pacman** The aim in this game was to drive Pacman in the maze in 4 directions - north, east, west and south using the 4 arrow keys. Accordingly, the most intuitive gesture mapping was to map the arrow keys to phone tilt in the respective direction using the **flat** mode
- **Google Doodle Game - Snake** The aim in this game was to drive the snake in the field in 4 directions just like the Pacman game. Once again, we chose to map phone tilt to the arrow keys to move the snake using the **flat** mode
- **Google Doodle Game - Cricket** The aim in this game was only to swing the bat at the right time so as to hit the ball as far as possible, or to hit it away from the fielders. The game control consisted of only the space bar to swing the bat. We mapped the **shake** gesture on the phone with the space bar key. Since timing the swing was crucial in this game, we increased the sensitivity of the phone controller so the user could see the ball carefully before deciding when to swing the bat and still make proper contact of the bat with the ball
- **Car Racing** This was a simple car driving game, where the car was to be operated by pressing the arrow keys. We mapped the arrow keys to the gestures in the **steering** mode as that was the most intuitive mapping for this game
- **Tetris** The aim in this game was to arrange the falling blocks of tetris so as to fit maximum number of blocks in the frame, with minimum holes or vacant spaces. The *left* and *right* arrow keys were used to move the blocks left or right, and the *down* arrow key was used to drop the block down. To change the orientation of the block, the game used the *up* arrow key. We mapped these movement keys to the left, right and down gestures in **flat** mode, and we mapped the *up* arrow key to change orientation to the up gesture in **flat** mode
- **Multiplayer Virtual Drumming** This was a fun interactive virtual drumming application. Each of the instrument - drums, cymbals and toms were operated by individual keys on the keyboard. We leveraged the multiple instruments by connecting more than one phone to our UDP server where each phone was operated by an individual. This allowed us to assign control of different instruments to different people. We were also able to replicate the damping effects on the cymbals by quickly changing the gesture on our phone, which corresponds to quickly pressing and releasing a key on the keyboard. A long key press, simulated by holding the gesture for long time resulted in trailing sounds of the cymbals

This also shows that our app could achieve gesture based control of third party games without requiring us to interfere with the game code, thus proving its universality of use. Below, we list a few more use cases outside of gaming where the Gesture Remote can be used!

Other use cases

Since our tool acts as a universal gesture Remote, users can use it for doing other use cases as well. We have listed a few of them below.

- **Website navigation** We can use the application to scroll web sites, select links and navigate as well. To do this, users can map the "tilt" actions of the phone to up and down arrow keys and we can map the "shake" action to "ctrl + shift + 1" to click on the first link.
- **Remote Control for Netflix** Another interesting and useful application could be to use our Gesture Remote as a remote to control Netflix. We can map actions to play/pause/rewind/forward the video by tilting the phone. We could also map actions to increase / decrease the volume.
- **Save and Screen Lock** A lot of times we end up leaving our laptops unlocked. However, we rarely forget carrying our phones with us. A simple application of the gesture remote could be by mapping our "shake" action to "Ctrl + S" followed "Windows + L" so that. Any time we grab our phone, it would automatically save all the work and lock our screen. A developer friendly version of this could be doing a "git commit / git push" when a developer starts to leave.

4 DISCUSSION AND LIMITATION

Even with our fixes and changes to a UDP server, users will still experience some lag. This may affect game play and should not be used for competitive or professional settings. In addition, right now users are restricted to only four directions when mapping movements to keybindings. Gestures are limited to these four directions in addition to shaking. Vigorous gestures may result in broken objects as there is no safety wrist strap. The movements with the phone is restricted to gentle actions as rough ones may cause injury. We plan on implementing safety features in order to prevent damage. Finally, we have no security features for our UDP server and app. Anyone with access to the same Wifi network will be able to issue commands to the UDP server.

Figure 5: First row from left to right, using Gesture Remote to play pacman, snake, cricket. Second row from left to right, using Gesture Remote to play car racing, tetris and virtual drums

Overall, we were able to successfully complete our initial challenges. With our gesture remote, we can play real-time multiplayer games with both accuracy and latency. The generic key mapping provided by the Gesture remote was greatly appreciated by the users. The gesture remote enables natural actions over mundane keyboard presses.

5 FUTURE WORK AND CONCLUSION

The Gesture Remote in its current state makes use of most commonly found sensors available on a mobile phone which includes gyro sensor and accelerometer. However, there are many more sensors and feedback devices on the phone which can be exploited to make the phone even better as a game controller. One idea would be to use vibration mechanism on the phone as a haptic feedback for game events. Since one important feature of our app is that it does not need to interfere with the source code of the game, we would like to stick to this approach of understanding game events without explicitly talking with the game. We could do so by monitoring the game audio for special highlight sounds which mark some event in game, and on detecting the sound, we send a message to the respective phone controller to vibrate, thus providing haptic feedback.

Another sensor that can be used is the light sensor or the proximity sensor. We could provide users, the option of executing binary actions like start or stop shooting a gun, start or stop applying nitro boosters in a car racing game, by moving our thumb over the light sensor thereby blocking out the light or allowing it. The phone speaker can also be used, by diverting certain sounds from the computer speaker to the phone speaker, thus creating spatial effects of audio.

Mobile devices are improving day after day with new technologies being added. The array of sensors and output devices on the phone and the customised mapping of gestures to game events makes it possible to think of intuitive gestures to perform actions in the game with appropriate feedback. It thus remains to be seen as to how much we can go ahead with utilising every single piece of hardware on our everyday device - the mobile phone.

6 RESOURCES

Demo video: https://www.youtube.com/watch?v=0rkeWaG9XeM
Github: https://github.com/cardwizard/CoronaPartyBox

7 ACKNOWLEDGEMENTS

We would like to thank our friends Shlok, Aman, Rajath, Naman, Siddharth, Abhishek, Pranav, Asmita, Aditya, Kushagra, Bharat, Akash, Maanav, Varun and Diksha for playing games using our Gesture remote and taking time to fill the survey. They did not force us to write this.

REFERENCES

[1] Nadia Bianchi-Berthouze, Whan Woong Kim, and Darshak Patel. 2007. Does Body Movement Engage You More in Digital Game Play? And Why?. In *Proceedings of the 2nd International Conference on Affective Computing and Intelligent Interaction (ACII '07)*. Springer-Verlag, Berlin, Heidelberg, 102–113. https://doi.org/10.1007/978-3-540-74889-2_10

[2] Beginner's Mind Collective and David Shaw. 2012. Makey Makey: improvising tangible and nature-based user interfaces. In *Proceedings*

of the sixth international conference on tangible, embedded and embodied interaction. 367–370.

[3] Jacques Foottit, Dave Brown, Stefan Marks, and Andy M. Connor. 2014. An Intuitive Tangible Game Controller. In *Proceedings of the 2014 Conference on Interactive Entertainment (IE2014)*. Association for Computing Machinery, New York, NY, USA, 1–7. https://doi.org/10.1145/2677758.2677774

[4] Mobile Gamepad. 2020. Mobile Gamepad. https://play.google.com/store/apps/details?id=com.mmh.mobilegamepad&hl=en_US.

[5] Kathrin Maria Gerling, Ian J. Livingston, Lennart E. Nacke, and Regan L. Mandryk. 2012. Full-body motion-based game interaction for older adults. In *CHI Conference on Human Factors in Computing Systems, CHI '12, Austin, TX, USA - May 05 - 10, 2012*, Joseph A. Konstan, Ed H. Chi, and Kristina Höök (Eds.). ACM, 1873–1882. https://doi.org/10.1145/2207676.2208324

[6] Younbo Jung, Koay Jing Li, Ng Sihui Janissa, Wong Li Chieh Gladys, and Kwan Min Lee. 2009. Games for a Better Life: Effects of Playing Wii Games on the Well-Being of Seniors in a Long-Term Care Facility. In *Proceedings of the Sixth Australasian Conference on Interactive Entertainment (IE '09)*. Association for Computing Machinery, New York, NY, USA, Article 5, 6 pages. https://doi.org/10.1145/1746050.1746055

[7] Python Library. 2020. pyBarcode 0.8b1. https://pypi.org/project/pyBarcode/.

[8] Python Library. 2020. pynput 1.6.8. https://pypi.org/project/pynput/.

[9] Python Library. 2020. socket — Low-level networking interface. https://docs.python.org/3/library/socket.html.

[10] Remote Mouse. 2020. Remote Mouse. https://www.remotemouse.net/.

[11] Wifi Mouse. 2020. WiFi Mouse(Android remote control PC/Mac). https://play.google.com/store/apps/details?id=com.necta.wifimousefree&hl=en_US.

[12] Juliet Norton, Chadwick A. Wingrave, and Joseph J. LaViola Jr. 2010. Exploring strategies and guidelines for developing full body video game interfaces. In *International Conference on the Foundations of Digital Games, FDG '10, Pacific Grove, CA, USA, June 19-21, 2010*, Ian Horswill and Yusuf Pisan (Eds.). ACM, 155–162. https://doi.org/10.1145/1822348.1822369

[13] Johanna Pirker, Mathias Pojer, Andreas Holzinger, and Christian Guetl. 2017. Gesture-Based Interactions in Video Games with the Leap Motion Controller. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 620–633. https://doi.org/10.1007/978-3-319-58071-5_47

[14] Shaileen Crawford Pokress and José Juan Dominguez Veiga. 2013. MIT App Inventor: Enabling personal mobile computing. *arXiv preprint arXiv:1310.2830* (2013).

[15] News Report. 2020. Animal Crossing powers Nintendo to record Switch sales in Japan. The biggest week for switch yet. https://www.theverge.com/2020/3/26/21195022/animal-crossing-switch-sales-japan-famitsus. Accessed: 2020-02-05.

[16] News Report. 2020. Doom Eternal had the series' best opening sales weekend. https://www.eurogamer.net/articles/2020-03-25-doom-eternal-had-the-series-best-opening-sales-weekend. Accessed: 2020-02-05.

[17] News Report. 2020. Live Streaming report. https://blog.streamlabs.com/streamlabs-stream-hatchet-q1-2020-live-streaming-industry-report-9630bc3e0e1e. Accessed: 2020-02-05.

[18] News Report. 2020. Using Oculus with motion-sensing controllers: The most fun video game experience I've ever had. https://www.geekwire.com/2015/using-oculus-with-motion-sensing-controllers-the-most-fun-video-game-experience-ive-ever-had/. Accessed: 2020-02-05.

[19] William Shelstad, Dustin Smith, and Barbara Chaparro. 2017. Gaming on the Rift: How Virtual Reality Affects Game User Satisfaction. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 61 (09 2017), 2072–2076. https://doi.org/10.1177/1541931213602001

[20] Tamas Vajk, Paul Coulton, Will Bamford, and Reuben Edwards. 2008. Using a mobile phone as a "wii-like" controller for playing games on a large public display. *International Journal of Computer Games Technology* 2008 (2008).

A SURVEY QUESTIONS AND RESULTS

Figure 6: Different users trying out the Gesture Remote.

We surveyed 12 participants who played our games with and without the gesture remote. The survey results can be accessed here. Below, we provide a summary of the feedback.

Profiling users based on game time

Majority of the users were people who were gaming just once a week but their game time has **increased significantly** during the COVID19 pandemic. Figure 7 shows all the responses that we received. It is important to understand this distribution since this is our perfect target audience. People who game a lot are likely to already own a Play Station controller or an XBox. Thus, we wish to target people who have play games once in a while but would be open to enhancing their experience.

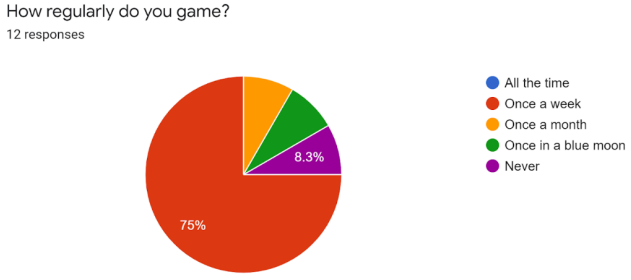

Figure 7: Response to the question: "How regularly do you game?"

On engagement with the tool

Most users found our game to be highly enjoyable and immersive. Figure 8 shows the responses we received.

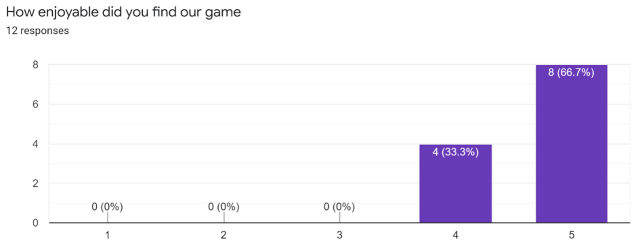

Figure 8: Response to the question: "How enjoyable did you find our game?"

A whopping **91.7%** users found our game to be more intuitive and **83.3%** users found it to be highly immersive as shown in Figure 9.

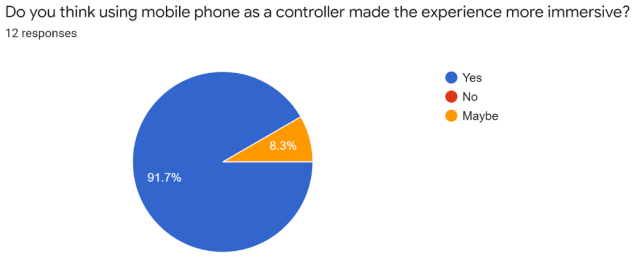

Figure 9: Responses to the question: "How immersive did you find the application?"

The users experienced some issues with latency. We also observed that given some time users adopted to playing the game with latency.

On Strengths, Areas of improvement and Features

When we asked users about what they **liked about our system**, here are some of the responses: "Good useful system with very grounded and fundamental applications", "The motion gestures were not too complicated. Very easy to pick and start.", "The sensitivity of the controller and by such a little effort playing a game was possible.", "The immersive experience of using your hands fully instead of using the keypad"

Users also suggested some key **areas for improvement**. We list them here:"It lagged a little bit, so maybe that.", "The dependency of phone being unlocked at all times should not be there. The app could run in the background, even when the phone is locked.", "Response time too high, rest position margin too low"

When asked about what **features** did they like the most, the users said, "Multiple people could connect for multiplayer games", "Moving the phone to swing the bat", "Ease of use, basically a free motion controller :)", "The feature to scroll and select alongwith it". The experience of playing the cricket game with the app was praised by multiple users.

Users were also proactive in **suggesting new applications for our tool**. This is important for us since our app acts as a Universal Gesture remote which allows users to find creative ways of using it. Here are some of their suggestions: "For controlling slideshows in presentations." "Or use the controller as a Netflix remote on your laptop.", "As soon as you leave your computer it automatically put your laptop to sleep and saving all the content in the process.", "This can be integrated with voice to create a complete PC controller".

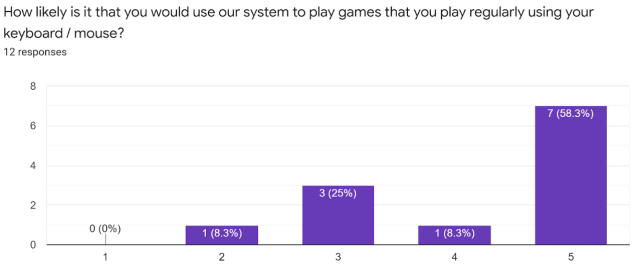

Figure 10: Responses to "How likely is it that you would use our system to play games that you play regularly using your keyboard / mouse?"

Lastly, we asked them how likely they were to use our application for playing games. We got mixed reviews from the users. Our survey tells us that **66.6%** users are willing to make the switch. We hope this number would increase after we fix the known issues and make the deployment and distribution of the tool easier. Figure 10 shows the entire distribution. This tells us that the app may have good market potential.