Smartphone as a Paired Game Input Device: An Application on HoloLens Head Mounted Augmented Reality System

Mehmet Sonat Karan, Mehmet İlker Berkman, and Güven Çatak

Abstract From social media filters to medical applications, AR (augmented reality) is a rapidly growing technology. On the other hand, AR gaming is a popular application of this field of study. Microsoft's see-through augmented reality headset HoloLens is one of the first and powerful headsets in the market today but it has limited interaction methods especially for gaming. This study deals with enhancing gaming experience in augmented reality devices by creating new interaction methods through a smartphone used as a game-controller an input device. We developed the Remote Input SDK (software development kit) to allow users to share input data between Unity applications to use a smartphone as remote controller while playing a game through HoloLens. This study explains the design and implementation of our SDK, which is available as an open-source component for Unity game development platform.

Keywords Augmented reality · Game development · HoloLens · Game controller

1 Introduction

Smartphones are the most common technical gadget and a large amount of the population own a smartphone in advanced and developing countries [1]. Today, sensors like accelerometers, magnetometers, light sensors and gyroscopes are standard components of smartphones along with the touchscreens. These sensors have become a rich data source [2]. By using these sensors, it is possible to create a virtual 6-DoF (6 degree of freedom) controller to interact with virtual world, using the data that collected from sensors to manipulate virtual elements. The sensor data

M. S. Karan
Bahcesehir University Game Design Graduate Programme, Istanbul, Turkey

M. İ. Berkman (✉)
Department of Communication and Design, Bahcesehir University, Istanbul, Turkey
e-mail: ilker.berkman@comm.bau.edu.tr

G. Çatak
Department of Digital Game Design, Bahcesehir University, Istanbul, Turkey

Ö. Cordan et al. (eds.), *Game + Design Education*, Springer Series in Design and Innovation 13, https://doi.org/10.1007/978-3-030-65060-5_22

of mobile phones has been used by game designers to develop interesting interaction schemes for mobile games, but rarely investigated for their potential as a peripheral controller device that is to be connected to a gaming platform. Although there are many studies on possible interactions schemes for multi-display systems using several mobile phone sensor capabilities, e.g. [3–5], only a few studies have employed the cross-device approach in game design.

In this study, we explored the possibilities of enhancing gaming experience in HoloLens augmented reality device by creating new interaction methods through a smartphone. We developed a SDK (software development kit), which we call Remote Input SDK, to allow developers share sensor input data between applications to use a smartphone as remote controller in order to engage with a game-like interactive environment through HoloLens.

Current input methods do not closely map to the interactions required for an enhanced gaming experience and there is a conflict with realism of the augmented world and limitations of the available interaction methods supported by current AR HMD sets.

2 Related Studies

2.1 Previous Attempts to Employ Mobile Devices as a Game Input Device

The idea of employing a mobile phone as an input device attracted designers and developers for various types of interactions. Ballagas et al. [6] reviewed several applications that employ mobile devices connected to other systems as input devices and categorize the inputs directed through by mobile devices as position, orient, select, path, quantify and text inputs. Some of the mobile phones employed in the applications they investigated were not smartphones and users were interacting through numeric keys, trackpads or velocity controlled keys on these devices. On the other hand, some of the applications employ smartphones to use their, accelerometers, cameras, RFID readers and microphones with voice recognition capabilities. These applications were mainly designed for ubiquitous computing purposes, such as allowing interactions on public displays.

Vajk et al. [7] had an early attempt to use mobile phone as game controller. They used a Nokia 5500 as a gaming interface for a multiplayer car racing game by developing an application to run Nokia's Symbian operating system that transfers data to computer through Bluetooth connection. Their system, namely Poppet Framework, was an extension of their previous study [8]. Their gameplay allowed multi-user interactions. Malfatti et al. [9] demonstrated a similar system that they had named as Bluewawe, based on Bluetooh signal transmission for a multiplayer game. Joselli et al. [10] also developed a system to control a 2D space shooter single player game running on an Apple tablet computer running IOS via an Apple I-Phone device.

The data connection between devices was using TCP-IP protocol through wireless network.

Finger Walking In Place (FWIP) is proposed as a navigational solution for CAVE virtual environments [11]. Originally it was implemented for the Lemur touchscreen midi controller and then exported to run on Apple IOS devices [12]. In a recent study, Liang et al. [13] employed a tablet computer as a controller for a virtual reality game. Using their system, players experienced a more efficient gameplay compared to their performance with a gamepad device.

Touchscreen handheld devices and their accelerometers had also been implemented in along with AR HMDs. Ha and Woo [14] developed a mobile application to control the positions of augment 3D objects viewed through a HMD, exploiting the phone's internal sensor information.

Several interaction and object manipulation methods were evaluated using a handheld device for input on an AR HMD system [15], revealing that interactions using devices with motion sensors and touch screen capabilities are applicable to AR.

2.2 Defining Augmented Reality

The journey of augmented reality started with the Sutherland's research on head mounted displays in the early days of computer graphics [16]. His prototype still inspires the researchers and developers who are working on virtual reality and augmented reality even though it is only capable of drawing simple vectors and detecting walls of a room.

Milgram and Kishino [17] placed augmented reality between virtual environment and real environment in their mixed reality continuum where real environments are located at the left end of continuum. Azuma [18] suggested that augmented reality is a technology that has three key requirements: (1) It combines real and virtual content. (2) It is interactive in real time (3) It is registered in 3D.

These requirements also define the technical requirements of augmented reality systems. First, it has to have a visualization mechanism that can combine virtual and real content. Secondly, there must be a software which is enabling a user to interact with the virtual environment. Finally, a tracking system is a must. An AR system has to find the position of the user's viewpoint in three-dimension environment and overlap virtual image successfully [19]. Augmented reality is aiming to make user interfaces invisible and improve user interaction with the real world environments at the opposite extremum.

Wearable technology means an electronics or a computer that embedded into clothing or designed as an accessory that can be worn on the body [20]. Wearable computers enhanced the connection between user and application by providing the data that is collected from directly user's body. For this reason, Starner et al. [21] had defined wearable systems are a natural platform for augmented reality.

2.3 Design Methods for Augmented Reality

There are too many interaction methods exist in the AR applications. These are mostly forming of existing interaction methods in desktop, mobile or VR applications. According to Billinghurst et al. [22], there is a need to develop interaction metaphors specialized for AR The author also found that when a new interface environment, such as AR is developed, it typically follows these steps: (1) Prototype demonstration (2) Adoption of interaction techniques from other interface metaphors (3) Development of new interface metaphors appropriate to the medium (4) Development of formal theoretical models for modelling user interactions. In addition, they suggest that for design of the physical and virtual objects of the application, the interaction metaphor needs to be developed. In this case, there are three elements that must be designed in the AR application; physical object, virtual element to be displayed, and the interaction metaphor.

Using design patterns is another approach in designing interaction of augmented reality applications. In order to improve game design of AR games and make them more enjoyable and intuitive [23]. Xu et al. [23] present nine design patterns for Handheld AR Games: Device metaphors, control mapping, seamful design, world consistency, landmarks, personal presence, living creatures, body constraints and hidden information.

2.4 Complexities in AR Software Development

Augmented Reality technology now available at outside of the laboratories. Nevertheless, AR technology have to solve some problems like mobility, tracking, cost, power usage and ergonomics, to be accepted as everyday technology such as smartphones or smart watches [24].

Mobility and outdoor usage are one of the main problems of AR. Most of the complex AR system requires powerful hardware and this means heavy machine with cables. Large devices with limited battery and cables create bulky experience for user [25].

Billinghurst et al. [19] point out that tracking and calibration are the major problem for AR systems especially for optical see-through AR displays. Because of the calibration parameters are spatial dependent it is hard to calibrate wearable device where the device position keeps changing with user's movement. Calibration failures causes misalignment between real and virtual view images. On the other hand, tracking and calibration problems bring the non-immersive experience together. In recent years, the tracking and calibration problems are not an issue any more, as AR development frameworks offer these functions as a standardized software feature.

Since the AR overlaps to user's view, some design guidelines must be followed to preventing user to overly rely on the AR application and missing important cues from the environment [24]. Bengler and Passaro [26] suggested guidelines to design

AR for cars. They focus on driving task, no moving or obstructing imagery. They provide information that improves driving performance.

Social acceptance has also been declared as important issue for wearable technologies like augmented reality by number of researchers. According to Billinghurst et al. [19], the reasons behind this reluctance are privacy concerns, or fear of looking silly, or being a target for thieves.

In addition to technical aspects of designing augmented reality experience, there is also commercial and ethical aspects. For example, if the application can only be runnable on specific devices, users have to buy them. This concept has been exploited by gaming console manufacturers. Thus, designing game for this concept is attractive under commercial perspective [27]. However, it is disadvantage under ethical perspective due to economical differences of players.

3 Development Environment

One of the first head mounted augmented reality headsets available on the market today is Microsoft's HoloLens. It comes with Universal Windows Platform (UWP) which offers powerful development environment. So, it is easy to develop games for HoloLens since developers can export their game via Unity3D to UWP.

HoloLens comes with 32-bit Intel processors, custom-built Microsoft Holographic Processing Unit (HPU 1.0) a coprocessor manufactured exclusively for HoloLens by Microsoft, 64 GB Flash, 2 GB RAM and built-in rechargeable battery. All of these processing units provide 2–3 h of active standalone usage (headset fully functional while using with power cable plugged in). It also has Inertial Measurement Unit (IMU) (including accelerometer, gyroscope and magnetometer), four environment-processing cameras (two on each side), a depth camera to map its surroundings and allow interaction between the real and virtual world while tracking the device's position, a 2.4-megapixel camera, microphone, and an ambient light sensor. HoloLens features Wi-Fi and Bluetooth for wireless connectivity. The headset uses Bluetooth to pair with the Clicker, a thumb-sized finger-operated input device that can be used for interface scrolling and selecting.

While using HoloLens, users need to be at least 1.25 m away from holograms. If users get closer the holograms for more than one meter, virtual objects rendered by the HoloLens will begin to fade out, and they will actually disappear as they close to 0.85 m away. The device has a very narrow field of view of just 35°. Although augmented reality and mixed reality has to fill user's view to provide truly immersive experience, in HoloLens users can only see holograms at the center of their vision in a small letterbox.

There is certain memory limitation for HoloLens app. If an application exceeds the memory limit, resource manager of HoloLens will terminate the application. Also, device has passive cooling. That means if its internal temperature spikes too high, the system will shut down the offending applications.

Developing application for HoloLens requires high-end system configuration. Although it is not necessary to own a HoloLens headset in order to develop applications for it, a special system configuration with a specific operating system version is required to run an emulator [28].

Unity is one of the most powerful game engines available in the market and it is possible to develop games with free plan of Unity. During the Build 2015 keynote, it was revealed that Unity is partnering with HoloLens. After that partnership they worked together to strengthen the integration between Unity and Visual Studio which results more efficient development environment for creating non-gaming 3D experiences and that's make the Unity is the best choice for developing applications on HoloLens. In addition, that partnership, Unity has large community, hundreds of tutorials online and Microsoft has released tutorials so it is easy to learn how to develop an application in Unity for HoloLens.

For the reasons mentioned above, we decided to use Unity versions released after version 2018.2 running on a computer with the Windows 64-bit Pro operating system with an eight-core AMD Ryzen 7 1700 3.0Ghz CPU, NVIDIA GTX 1060 6 GB GPU, 16 GB of RAM. We used both the HoloLens emulator and the headset itself during the development process.

4 Design and Development

Our Remote Input SDK allows users to share predefined input data such as sensor data and virtual gamepad button states over Wi-Fi between applications which were built in Unity3D game engine. SDK was written in C# and it uses Transmission Control Protocol (TCP) for wireless communication. It can be added to any Unity Project by just extracting the unity package file which is available to download at GitHub page of SDK or it can be also downloadable from GitHub as Unity Project, via the URLs given in supplementary data.

4.1 The Structure and Flow in Remote Input SDK

Developing a Universal Windows Platform game with Unity using.NET may require extra coding because some of the.NET libraries are not available. When building a game with Unity for multiple platform including UWP, Microsoft suggest that using platform-dependent compilation to make sure that UWP is only run when the game build for UWP.

Remote Input SDK is requiring Networking namespaces such as System.Net, System.Net.Sockets and these namespaces are not available when building a Unity game for UWP with.Net scripting backend. While developing network part of the Remote Input SDK, the strategy software design pattern has been used to overcome this problem. The strategy software design pattern is a behavioral pattern that allows

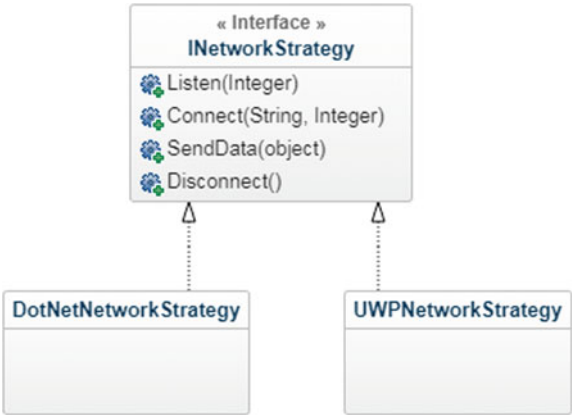

Fig. 1 Usage of strategy software design pattern in remote input SDK. *Source* Mehmet Sonat Karan

selecting algorithm at runtime. Remote Input SDK has two different network strategy one for specially UWP, UWPNetworkStrategy class and the other one for all other platforms DotNetNetworkStrategy class, which can be inspected on Fig. 1.

At the application startup, NetworkStrategyFactory class decide which network strategy class should be instantiated with help of platform-dependent compiling. See the following code on Fig. 2.

Once the network strategy instance is created, all methods of the INetworkStrategy interface will be implemented with the correct algorithm based on the runtime platform. Since the Unity support IL2CPP scripting backend for UWP, DotNetNetwork-Strategy implementation works for all platforms when building applications with IL2CPP. So this approach became unnecessary.

SDK can be used as Host or Controller. A state machine is used to handle this states and transitions between states.

There are two different state in Remote Input SDK; MainMenu and GamePad states. MainMenu states can act differently according to SDK's working mode which

Fig. 2 Sample for strategy software design pattern. *Source* Mehmet Sonat Karan

```
#if NETFX_CORE

        _networkStrategy =
NetworkStrategyFactory.Instance.CreateUWPworkStrategy();

#else

        _networkStrategy =
NetworkStrategyFactory.Instance.CreateDotNetworkStrategy();

#endif
```

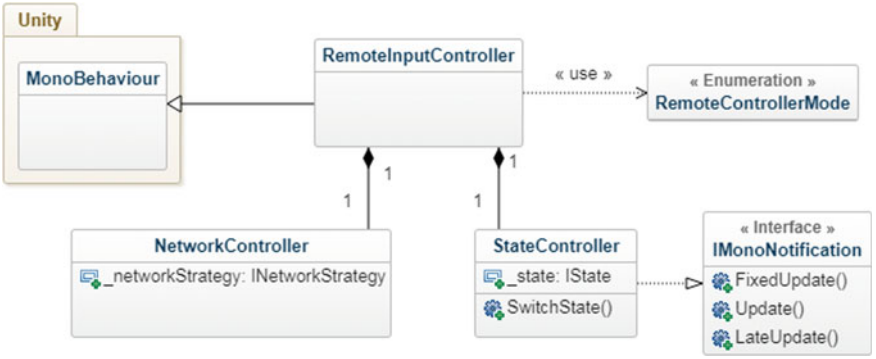

Fig. 3 Remote Input SDK class diagram. *Source* Mehmet Sonat Karan

can be change between two modes; Host or Controller. If Host mode selected, Main-Menu state create a view for host application which allows users to Listen specific port to any possible controller connection. If controller mode is selected state will create a view that allows users to enter host's IP address and let them connect to host. GamePad state is only available if application is running on controller mode. When SDK is connected to host application, state controller will automatically switch state to GamePad state and a gamepad view will be created.

As on overview of Remote Input SDK, there is one MonoBehaviour which is main controller of the system (See Fig. 3). It is a component added to a GameObject in initial scene and it does not affect from scene loads. Lifetime of this controller starts when the application start, and end when user terminate the app. RemoteInputController has two sub-controllers; NetworkController and State Controller. NetworkController is responsible for wireless communication and StateController is controlling the SDK's states. StateController uses IMonoNotification interface to get update calls from MonoBehaviour.

While developing Remote Input SDK, UnityMainThreadDispatcher, VirtualJoystick, FollowCamera and Unka the Dragon asset [29] external components are used. While working with Unity, if an operation modifies engine's properties it has to be done in main thread of Unity. To do this we use UnityMainThreadDispatcher. It is a thread-safe way of dispatching coroutines to the main thread in Unity [30]. VirtualJoystick is small component which is developed by researcher that allows user to interact with unity application by using virtual joystick. Finally, Unka The Dragon is an asset which is available in the Unity Asset Store [29]. The asset contains high quality 3D model and predefined animation sets.

The interface for our Remote Input SDK is prototyped as depicted in Fig. 4. Prototype was tested on Android and iOS devices as Controller and a PC used as host. Prototype can act as both Host or Controller. If user pressed Listen button application simply listen to predefined port and when a controller enters the host's IP to address bar and simply pressed to Connect button data transfer will start from Controller to Host.

Fig. 4 Remote Input SDK running on Windows PC (Host) and Android device (Controller). *Source* Mehmet Sonat Karan

4.2 Game Development and Use of Remote Input SDK in the Environment

Developing a Holographic User Interface was the first challenge we had to overcome while developing an application for HoloLens with Unity. It should be work on both Unity Editor and the HoloLens itself. So we follow the guideline that is described in Unity Forums [31]. First of all, we add HoloLens Input Module component to EventSystem object and leave Standalone Input Module attached so our UI still work with editor. After that, we changed canvas render mode to World Space so it can be registered in 3D space. Finally, we create virtual cursor to more easily tell to user where their gaze is pointed.

Microsoft developed a toolkit which is aiming rapid prototyping, accelerating content production and removing complexities in multi-platform XR solutions [32]. It is available at GitHub and also downloadable as a Unity package file. We use this toolkit while prototyping our HoloLens app. Toolkit contains examples that showing the best practices for mixed reality development specially with Unity. It also has ready to use prefabs which is really helpful while creating simple holographic user interfaces.

Unka the Dragon [29] asset provides high quality 3D animal models with fully functional animation setup. We have chosen to use this asset because their highly detailed and animated dragons can fly in the air and walk in the ground so it perfectly matches with our control-mapping method where the user's actions are registered in 3D space and mapped with avatar's actions.

We use accelerometer sensor to detect tilting up and down states of controller. Unity give access to last measured linear acceleration data of the device in three-dimensional space. We collect that data every frame and transform it to tilting states according the data change in last 10 frames.

The life cycle of Remote Input game object starts when the scene that contains it is load and it ends when the application is closed. So before trying to connect a controller, a scene with the Remote Input game object must be loaded.

Listen method of RemoteInputController class has to be called before trying to connect with controller application. After calling Listen method host will be ready for connection and ReadyForConnection event will be fired. Now controller application can connect to host. When the controller connected, ControllerConnected event will be fired and the controller will begin to start sending input data over WiFi. Every time an input data received an event called InputDataReceived will be fired. This event contains input data in it, and it can apply directly to the environment.

When a scene with Controller mode and RemoteInput game object is selected, the controller UI is shown. The user must type the IP address of the host in the address field and press the connect button to connect a host. The GamePad view is automatically displayed when the controller is connected to the host computer. GamePad view can be customized in consonance with the game requirements. The GamePad class that contains the input data must be modified according to the GamePad view.

As it can be seen in videos at supplementary material, when the users run the game that contains Host application on their HoloLens devices, HoloLens device prompts the IP address of itself and tells them to run the Controller application on their mobile device. After doing this, users can control the Unka the Dragon [29] around the room they are in, by tilting up the mobile phone upwards to start its flight, which is an accelerometer based input to the gameplay. They can land the dragon by tilting their phone downwards. A joystick-like interface on the touchscreen of the mobile phone lets the users to control the dragon's direction in 3D space and a button on the phone touchscreen triggers flame throws, as depicted on Fig. 5.

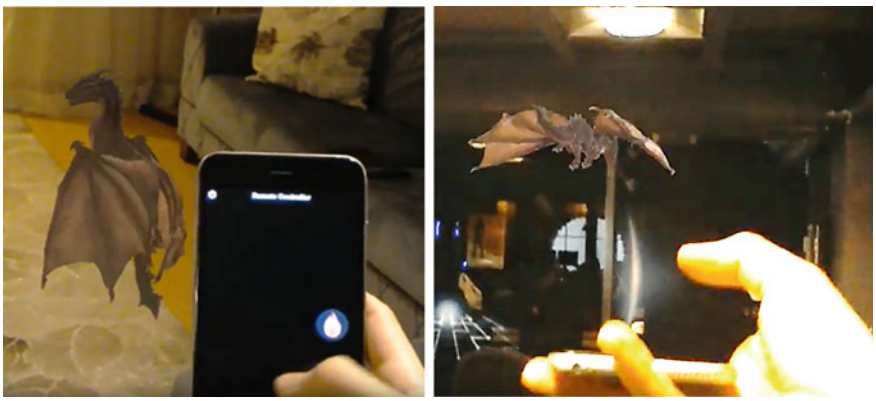

Fig. 5 Screenshots from the user's HMD view. *Source* Mehmet Sonat Karan

5 Conclusion

In this study, we used smartphone as an input device and use the sensor data as an input data to enhance gaming experience on Microsoft's augmented reality headset, HoloLens. Although the idea of employing a mobile device as a game controller had been implemented for computer games and virtual reality applications previously, The SDK we developed is not only for connecting a smartphone to Microsoft's HoloLens AR device, but also it can be used to connect a mobile phone to any gaming platform such as desktop PC's and secondary mobile devices, which support applications created with Unity development platform.

The Remote Input SDK can create connection between Unity applications so an application developed with Unity for XR devices can be controlled by a smartphone. Users can add input methods to their application like gyroscope or accelerometer by using smartphone or they can use smartphone as an information browser. Since our SDK uses wireless network technology to connect other applications, users can also interact with their application out of the headset camera's boundaries.

The SDK has a restriction on connectivity. SDK can lose connection or transfer data with delay when network signal is weak. To overcome this problem other connection methods like Bluetooth can be added to system in the future. Second limitation is about HoloLens' development environment. HoloLens is an expensive device, if developer's budget is not enough to use a HoloLens, he/she needs to work with emulator. HoloLens emulator required more powerful hardware and experiences in the virtual environment may not coincide with the actual environment for some cases.

Considering the popularity of smartphones and their rich interaction capabilities based on their sensors and cameras, we believe that Our SDK could be an inspiration and a starting point for game development platform vendors such as Unity should include built-in functions to let game developers employ a mobile phone as controller in their games, either their game is developed to primarily run on a platform other than a mobile device, and even to control a game running on another mobile device such as a tablet computer. However, further research is required via involving the players, to explore the user performance and game user experience in order to develop rich and powerful interactions with smartphone based paired input devices.

Supplementary Material

User video captures of the application in action

https://www.youtube.com/watch?v=EIX19dNL65U

https://www.youtube.com/watch?v=_xVtDZa8d68

Source code of Remote Input SDK

https://github.com/karansonat/RemoteInput

References

1. Silver L (2019) Smartphone ownership is growing rapidly around the world, but Not always equally. Pew Research Center, [Online]. Available: https://www.pewresearch.org/global/2019/02/05/smartphone-ownership-is-growing-rapidly-around-the-world-but-not-always-equally/
2. Shoaib M, Bosch S, Incel OD, Scholten H, Havinga PJM (2015) A survey of online activity recognition using mobile phones. Sensors (Switzerland)
3. Paay J et al (2017) A comparison of techniques for cross-device interaction from mobile devices to large displays. J Mob Multimed
4. Ramos G, Hinckley K, Wilson A, Sarin R (2009) Synchronous gestures in multi-display environments. Hum-Comput Interact
5. Chong MK, Mayrhofer R, Gellersen H (2014) A survey of user interaction for spontaneous device association. ACM Comput Surv
6. Ballagas R, Borchers J, Rohs M, Sheridan JG (2006) The smart phone: a ubiquitous input device. IEEE Pervasive Comput
7. Vajk T, Coulton P, Bamford W, Edwards R (2008) Using a mobile phone as a 'wii-like' controller for playing games on a large public display. Int J Comput Games Technol
8. Vajk T, Bamford W, Coulton P, Edwards R (2007) Using a mobile phone as a 'Wii like' controller. In: Proceedings of the 3rd international conference on games research and development
9. Malfatti SM, Santos Dos FF, Santos Dos FF (2011) Using mobile phones to control desktop multiplayer games. In: Proceedings 2010 Brazilian symposium on games and digital entertainment SBGames
10. Joselli M et al (2012) An architecture for game interaction using mobile. In: 4th international IEEE consumer electronic society—games innovation conference, IGiC 2012
11. Kim JS, Gračanin D, Matković K, Quek F (2008) Finger walking in place (FWIP): a traveling technique in virtual environments. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)
12. Kim JS, Gračanin D, Matković K, Quek F (2009) iPhone/iPod touch as input devices for navigation in immersive virtual environments. In: Proceedings—IEEE virtual reality
13. Liang HN, Shi Y, Lu F, Yang J, Papangelis K (2016) VRM controller: an input device for navigation activities in virtual reality environments. In: Proceedings—VRCAI 2016: 15th ACM SIGGRAPH conference on virtual-reality continuum and its applications in industry
14. Ha T, Woo W (2011) ARWand: phone-based 3D object manipulation in augmented reality environment. In: Proceedings 2011 international symposium on ubiquitous virtual reality, ISUVR 2011
15. Budhiraja R, Lee GA, Billinghurst M (2013) Using a HHD with a HMD for mobile AR interaction. In: 2013 IEEE international symposium on mixed and augmented reality ISMAR, 2013
16. Sutherland IE (1968) A head-mounted three dimensional display. In: Proceedings of the December 9–11, 1968, fall joint computer conference, part I on—AFIPS '68 (Fall, part I), p 757
17. Milgram P., Kishino F (1994) Taxonomy of mixed reality visual displays. IEICE Trans Inf Syst
18. Azuma RT (1997) A survey of augmented reality. Presence Teleoperators Virtual Environ 6(4):355–385
19. Billinghurst M, Clark A, Lee G (2014) A survey of augmented reality. In: Foundations and trends in human-computer interaction
20. Wright R, Keith L (2014) Wearable technology: if the tech fits, wear it. J Electron Resour Med Libr
21. Starner T et al (1997) Augmented reality through wearable computing. Presence Teleoperators Virtual Environ
22. Billinghurst M, Grasset R, Looser J (2005) Designing augmented reality interfaces. Comput Graphics (ACM)

23. Xu Y (2011) "Pre-patterns for designing embodied interactions in handheld augmented reality games. In: 2011 IEEE international symposium on mixed and augmented reality—arts media, and humanities, 19–28
24. van Krevelen DEF, Poelman R (2010) A survey of augmented reality technologies, applications and limittions. Int J Virtual Real
25. Khor WS, Baker B, Amin K, Chan A, Patel K, Wong J (2016) Augmented and virtual reality in surgery-the digital surgical environment: applications, limitations and legal pitfalls. Ann Trans Med
26. Bengler K, Passaro R (2004) Augmented reality in cars requirements and constraints. BMW Gr. Forsch. und Tech
27. Lindt I, Ohlenburg J, Pankoke-Babatz U, Ghellal S, Oppermann L, Adams M (2005) Designing cross media games. In: Proceedings PerGames 2005—3rd International Conference on Pervasive Computing—PERVASIVE 2005. Munich, Germany
28. Community MD (2018) Using the HoloLens emulator. Windows Dev Center, [Online]. Available: https://docs.microsoft.com/en-us/windows/mixed-reality/using-the-hololens-emulator Accessed 18 Oct 2018
29. Malbers Animations, Unka the Dragon. Unity Asset Store [Online]. Available: https://assetstore.unity.com/packages/3d/characters/creatures/unka-the-dragon-84283
30. De Witte P UnityMainThreadDispatcher [Online]. Available: https://github.com/PimDeWitte/UnityMainThreadDispatcher.
31. Unity UI on the HoloLens Unity Forums, [Online]. Available: https://forum.unity.com/threads/unity-ui-on-the-hololens.394629/. Accessed 16 Nov 2018
32. D. (Microsoft) Kline Mixedrealitytoolkit-unity, GitHub, [Online]. Available: https://github.com/microsoft/MixedRealityToolkit-Unity. Accessed 10 Oct 2018