






VAMR Basketball on Head-Mounted and Hand-Held Devices with Hand-Gesture-Based Interactions

Eric Cesar E. Vidal Jr.   and Ma. Mercedes T. Rodrigo 

Ateneo de Manila University, Katipunan Avenue, 1108 Quezon City, Philippines

Abstract. Virtual, Augmented, and Mixed Reality (VAMR)-based sports simulators may help facilitate learning and training for students confined within closed/cramped spaces (e.g., during a pandemic), and allow disabled or disadvantaged players to participate in the said sports alongside non-disabled peers. This work explores the development of a VAMR basketball simulator that uses hand gestures to accurately mimic the real-world performance of basketball moves such as throwing and shooting. Furthermore, the simulator is developed simultaneously for both head-mounted (VR/AR headsets) and hand-held (smartphone/tablet) form factors, facilitating deployment of the same simulator over multiple hardware configurations. This eliminates the need to force students to use one platform for online learning, and would also allow for future real-time networked multi-user support across different form-factor devices. This model also paves the way for future user studies comparing the efficacy of head-mounted and hand-held modes for simulating basketball and other sports in VAMR.

Keywords: CAVE and multi-participant environments · Education and training · Gaming · Human factors · Mobile systems · Presence in VAMR

1 Introduction

Traditional sports games, such as basketball, have been a popular subject of computer-based simulations for entertainment, teaching and training. Computer-generated virtual environments provide users the opportunity to participate in such sports while in remote places [1, 2]. Such virtual environments are realized using Virtual, Augmented, and/or Mixed Reality technologies, collectively referred to as VAMR.

The increased reliance on distance-based learning due to the COVID-19 pandemic means that the task of keeping students motivated and engaged in learning activities has become more challenging than ever. Hergüner et al.'s study found that positive attitudes towards online learning amongst students of sports sciences directly contribute to an improvement in said students' readiness to learn online [3]. An online learning environment that is not easily accessible to students for reasons relating to instructional delivery, usability, or cost/availability, will likely impair students' learning attitudes, or worse, prevent participation in the learning activity altogether. Therefore, creating a

sports learning environment that is easily accessible to a wide range of students is an important key towards successful online learning of sports.

This ease-of-access problem initially appears to be adequately addressed via the adoption of VAMR-based sports simulations in the curriculum. In contrast to more traditional methods such as lecturing and passive multimedia, VAMR deals with the problem of instructional delivery by putting users in direct exposure to sports scenarios, with an emphasis on learning-by-doing, e.g., players would physically learn how to shoot a free throw, pass the ball to a teammate, and execute offensive or defensive plays step-by-step. In addition, the novelty effect of using a VAMR-based learning tool for the first time may significantly increase student motivation [4]. However, studies of VAMR applications reveal that the novelty effect may wear out over prolonged use, and any underlying usability issues, such as repetitiveness of bodily motion, discomfort from equipping/holding the device, user weariness, and frustration, become increasingly distracting over time [5].

Alongside these usability issues, cost and availability issues are also important factors for students. VAMR devices can be very expensive, with high-end devices such as Microsoft HoloLens selling for thousands of dollars and is outside of the typical student's budget. More recently, Augmented Reality on mobile phones and tablets are increasingly used as cheaper alternatives to full-fledged Virtual Reality headsets. However, this market fragmentation causes further availability issues for educators: an educator may not be able to mandate the use of the exact same hardware platform for a given class, which would force additional costs on students who already own some other VAMR-capable device. Standardization efforts of VAMR application programming interfaces, such as OpenXR [6], would allow applications to be written once for multiple platforms, solving this fragmentation problem. However, OpenXR is currently not available on all platforms (notably on iOS and Android handheld devices), and even then, developers still need to contend with per-platform variations, such as widely divergent field-of-view (FOV) display angles and sensor configurations across device models. Furthermore, mobile phones typically do not ship with OpenXR-ready input controllers, forcing developers to implement non-OpenXR alternatives, such as hand gesture tracking via the phone's front-facing camera [7].

This paper specifies and addresses these usability, cost, and availability issues, by documenting the design of a proof-of-concept, cross-platform VAMR basketball simulator. Basketball was chosen due to the popularity of the sport in the authors' home country, Philippines, as well as the fact that basketball's basic moves of ball passing and ball shooting are intuitively translated into hand gestures (whether tracked via controller input or front-facing camera) across all VAMR-capable devices, making the said sport a practicable starting point towards a later, more comprehensive study of VAMR use in overall sports teaching.

Parallel to the objective of teaching sports, another potential use of VAMR-based sport simulation is to enable the sport to be played by people who otherwise could not participate in a real-world sporting event due to disabilities. A VAMR-based simulation can theoretically allow multiple levels of computer assistance. In the case of basketball, this includes easier movement around the playing court, easier execution of passing/shooting actions, full/partial guidance of shots, and player height adjustment.

Such assistive features potentially increase the accessibility of a VAMR sports simulator even further for a wider variety of users and are thus considered in this paper.

2 Simulator Design

The design of our work-in-progress basketball simulator fundamentally follows an iterative process, as the on-going research explores new ideas and variations on existing ideas in terms of user control and interface. The simulator is currently designed as a freeform basketball court “sandbox”, with a player able to freely interact with the ball and court with working physics, e.g., the user can aim and throw the ball anywhere on the court, including both goal baskets.

The features of our simulator were mostly inspired by the following previous work: an academic free-throw simulator [8] implemented using an XVR CAVE (a projected whole-room VR system) and motion tracking provided by Polhemus/DTrack magnetic trackers attached to the player’s hand; and, a commercial simulator [9] that was designed specifically for the HTC VIVE headset and controllers. The academic simulator focused primarily on free-throw simulation and was thus limited to visualizing the free throw shot’s direction and speed. On the other hand, the commercial simulator is more sandbox-like, allowing the player to shoot from any position in a half-court.

2.1 Hardware and Software Requirements

Our simulator is written in Unity version 2019.4, a commercial game engine that supports several extension frameworks to enable VR and AR rendering, but currently does not support OpenXR natively. For the simulator, it was decided to use the SteamVR extension framework to enable the use of a third-party hand gesture recognition plugin, HTC’s VIVE Hand Tracking SDK. Unity’s built-in XR subsystem is used for Microsoft HoloLens, which includes support for finger-pointing and tapping gestures.

Meanwhile, mobile platforms are supported using Unity’s AR Foundation extension framework. Hand gesture support is not available by default; instead, the authors’ own AMDG framework [7] is used. The AMDG framework offers users a handheld VAMR interaction style where the user’s primary hand executes actions, while the other hand holds the device up near the eyes (see Fig. 1, right), providing a VAMR experience without the need for a headset. AMDG supports open-hand, closed-hand, and finger-pointing gestures. The AMDG hand detection engine has been updated to support people occlusion detection (available on iOS A12 devices and later) and LiDAR-based depth mapping (available on iPad Pro 2020 and iPhone 12 Pro).

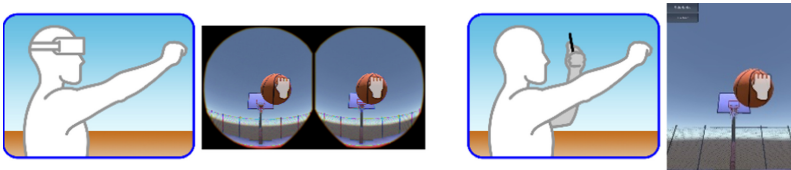


Fig. 1. Head-mounted (left) versus hand-held (right) configurations.

Table 1 summarizes the specific hardware and software configurations used to make our simulator available on four different types of devices: HTC VIVE, Microsoft HoloLens 1, iPhone 11 and iPad Pro 2020. These platforms were consciously chosen to cover a wide range of VAMR hardware archetypes while minimizing hardware acquisition costs and development time: the VIVE is a wide-FOV VR headset that is tethered to a computer, the HoloLens is a see-through AR headset with limited FOV but does not require a connected computer, and the iPhone and iPad are mobile handheld devices coming in small and large form factors, respectively. Other platforms, such as the Oculus Quest 2, HoloLens 2 and Android-based handheld devices, may be supported in later versions of the simulator as the hardware becomes available to the authors.

Table 1. Hardware and software configurations supported by the basketball simulator.

	HTC VIVE	Microsoft HoloLens 1	iPhone 11	iPad Pro 2020
Form factor	Tethered VR headset	Untethered AR headset	Mobile phone	Mobile tablet
Hand gesture support	Full support (via VIVE Hand Tracking SDK)	Finger-pointing and tapping only	Full support (via AMDG)	Full support (via AMDG)
Hand gesture capture hardware	VIVE front-facing camera	HoloLens depth camera	Front-facing camera with people occlusion detection	LiDAR scanner
Additional software libraries	SteamVR 1.16.10 VIVE Hand Tracking SDK 0.9.4	n/a	AR Foundation 2.1.16 AMDG OpenCV for Unity 2.4.3	AR Foundation 2.1.16 AMDG OpenCV for Unity 2.4.3

As a special case, since the HoloLens 1 hardware can only detect finger-pointing gestures, for this application, all open-hand gestures can also be done via finger-pointing (i.e., all subsequent references to open-hand and finger-pointing are equivalent).

2.2 Features Implemented

Our simulator borrows main features from the two previously described academic and commercial simulators, as well as expanding them with new features to turn our simulator into a full-fledged learning and training tool. Each feature in this discussion is thus described with appropriate comparisons to the previous systems.

Ball-Throwing. Both previous simulators track the player’s hand motion to estimate the ball’s motion after the player’s intent to throw has been detected. For the academic simulator, the point of throwing is determined by velocity and acceleration thresholds;

once these thresholds were exceeded, the position, velocity, and acceleration at that point in time are sampled and used as the ball's initial parameters, and then simulated further using a Matlab ode45 solver. The commercial simulator, on the other hand, uses the VIVE controller's grip button to determine whether the ball has been released, and collects real-time position measurements of the VIVE controller to calculate velocity; the actual computation is unfortunately not documented due to the simulator's commercial nature. However, since the VIVE was the flagship hardware of the SteamVR platform, it is suspected that the computation is similar to that of the SteamVR plugin's throw computation [10]. SteamVR uses two possible estimation strategies for computing the release velocity of a thrown object when using the VIVE controller: a simple estimation that uses the previous three frames of hand positions to calculate the velocity on release, and an advanced estimation that buffers up to 10 frames of previous hand positions, finds the frame with the peak speed, and calculates the velocity with the help of the previous and next frame surrounding the peak frame.

For our simulator, since hand gestures are used instead of controllers, it was initially decided that the closed-hand gesture would be used to initialize ball-throwing, and the open-hand gesture would be used to signal the point of release, with the in-between motion of the hand used for estimating ball velocity (see Fig. 2). However, there is a noticeable delay in the detection of the open-hand gesture, possibly causing unexpected throwing behavior. Also, the HoloLens equivalent gesture (bringing a finger down then up to throw) may seem unintuitive for users. Thus, an alternate mode that uses velocity thresholds, similar to that of the academic simulator, is available as an option. The option is presented so that future usability testing may determine which method (close-move or close-move-open) would be ideal for actual use. The velocity (i.e., speed and direction) of the throw is then calculated using the last 3 frames, as with SteamVR.



Fig. 2. Shooting. From left to right: closed hand, forward movement, open hand (optional).

Player Teleport/Turning. The previous commercial simulator allows for easier movement around the court via the standard SteamVR “teleport” command. Teleportation can be used by disabled or less-athletic people to move around the virtual court without having to physically move in the real world. A teleport command is executed by pressing a physical controller's touch pad, which then shows a visual indication of the destination location of the player, and releasing the touchpad teleports the player to the said location. In addition, pressing the left or right edges of the touch pad allows players to rotate their viewpoint counterclockwise or clockwise, respectively. All viewpoint changes resulting from teleport or turning are instantaneous, with no in-between motion; this is recommended by the creators of SteamVR to prevent motion sickness.

This feature is likewise implemented in our simulator but is activated using hand gestures instead; the user points at a location on the floor and taps that location to teleport. However, if the teleport is instantaneous as with SteamVR, this gives a possibly unfair playing advantage to teleporting users over physically-moving users.

To reintroduce balance, it was decided to add a delay in the form of a circular progress bar (see Fig. 3). The progress bar fills up over time, and completion time is proportional to the travel distance; releasing the tap before completion shall cancel the teleport action. The teleport is also cancelled by moving the hand left or right, which causes a viewpoint rotation action to occur instead.



Fig. 3. Teleporting. From left to right: point, tap (with progress bar), release to teleport.

Presence Indicators. A persistent problem in VAMR is the diminished ability of users to establish the presence of other virtual objects, due to the limited field-of-view (FOV) of the hardware. This problem is especially severe on Microsoft HoloLens with an FOV of only 30° (horizontal). For example, sudden turning and movement can make the player lose their bearings and consequently lose track of the positions of the ball or goal. The previous commercial simulator avoided this problem by truncating the court into a half-court (so only one goal basket is available), and a new ball is automatically thrown at the user after every shot to ensure that the user always sees the ball. This simplification unfortunately rules out full-court simulation with multiple players.

To address this problem, our simulator uses icons at the edges of the visible screen (see Fig. 4) to act as presence indicators pointing to the off-screen positions of objects. When the player does not possess the ball, a ball icon indicates the off-screen position of the ball. If the ball is currently possessed by the player, a presence indicator of the player's (correct) goal appears instead.

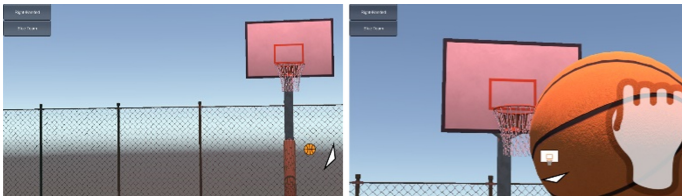


Fig. 4. Presence indicators. Left: missing ball. Right: correct goal location.

Shooting Assistance. To assist people with disabilities, our simulator offers multiple levels of shooting assistance. While this is partly supported by the commercial simulator, which offers two slider bars for controlling shot power and angle auto-correction, the configurable parameters for our simulator are altered to specifically provide disability assistance or training as opposed to just simplifying the simulation for casual use.

The first parameter is an automatic power adjustment variable, which, when set to 0, makes it so that initial speed of the ball is automatically calculated from the player's distance from the goal. When set to 1, the full speed is taken from the magnitude of the player's own hand movement. A value between 0 and 1 has the effect of amplifying weaker shots and attenuating stronger shots. Players can start at 0 to effectively allow a simple "close-open" gesture to launch the ball (perfect for beginners or disabled players), then gradually increase the value to 1 as they get better at estimating shot power.

The second parameter is an angle adjustment variable, which works similarly to angle auto-correction. However, instead of correcting the angle to always make the goal (which is not very useful for progressive training), a slider value of 0 uses the player's forward gaze as the ball's direction, 1 uses the direction of the player's hand movement vector, and in-between values interpolate between these two vectors. This way, players who are still developing their shooting aim can intuitively use their gaze instead to guide the shot (and still learn the proper angles for taking shots, which may be enough for the needs of younger users in physical education programs).

A third parameter, not present in the previous simulator, is player height adjustment. Controlling the player's height can be useful to allow shorter or disabled players experience ball shooting in the same way as standing or taller players. Conversely, taller players can also use this setting as a handicap for training.

3 Simulator Prototype Issues and Evaluation

As mentioned in the previous section, the simulator's design was iterated upon, and features were added or improved as needed. Early versions of the simulator did not incorporate teleportation and used only "close-open" gestures (with no interim hand movement) for shooting. Teleportation was initially added as an option that can only be triggered using external controllers (or the touchscreen on iOS devices) but is later simplified to be performed as a hand gesture by using context sensitivity: pointing towards the ground allows teleportation, while pointing upwards allows throwing. This simplifies interaction immensely and eliminates the need for separate controllers.

The current design of the simulator is geared towards implementing multi-user support, which inspired the implementation of teleportation progress bars. Future multi-user support would add gestures for throwing the ball towards teammates and stealing the ball from opponents. The system will also have to allow for event-based scripting to handle refereeing (e.g., moving into the space of another player should cause a foul event, which in turn can auto-arrange players into a free-throw configuration). Event-based scripting can also enable directed learning activities, such as training for consecutive free throws, 3-point shots, or ball-passing exercises. This can further be expanded to visualization and execution of playing tactics, similar to [11].

An evaluation of the system would involve observed user trials, with users testing the system on each supported device and evaluating their usability based on metrics

such as SUS [12]. While these tests are currently not possible due to global pandemic lockdowns, it will be useful in the future to gauge the system's effectiveness as a learning and training tool, and to perform cross-evaluation of different device configurations, both head-mounted and hand-held.

4 Conclusion

This study developed a VAMR basketball simulator intended for distance-based learning and training, with features to assist disabled or disadvantaged players. The simulator primarily uses hand-gesture-based interaction to eliminate the need for additional input devices. The simulator is designed for both head-mounted and hand-held platforms to maximize its potential userbase, allow for future cross-platform networking, and permit forthcoming usability comparisons of head-mounted and hand-held form factors for effective VAMR-based sports training.

References

1. Soltani, P., Morice, A.H.P.: Augmented reality tools for sports education and training. *Comput. Educ.* **155**, 103923 (2020)
2. Pato, A.S., Remilllard, J.D.: eSport: Towards a Hermeneutic of Virtual Sport. (eSport: hacia una hermenéutica del deporte virtual). *Cultura, Ciencia y Deporte* (2018)
3. Hergüner, G., Yaman, Ç., Sari, S.Ç., Yaman, M.S., Dönmez, A.: The effect of online learning attitudes of sports sciences students on their learning readiness to learn online in the era of the new coronavirus pandemic (Covid-19). *Turkish Online J. Educ. Technol.* **68** (2021)
4. Huang, W.: Investigating the novelty effect in virtual reality on STEM learning. Ph.D. dissertation, Arizona State University (2020)
5. Rodrigo, M.M.T., Vidal, E.C.J.E., Caluya, N.R., Agapito, J., Diy, W.D.: Usability study of an augmented reality game for philippine history. Presented at the 24th international conference on computers in education, Mumbai, India (2016)
6. Khronos Group Inc.: The OpenXR Specification: What is OpenXR? https://www.khronos.org/registry/OpenXR/specs/1.0/html/xrspec.html#_what_is_openxr. Accessed 19 Mar 2021
7. Vidal, E.C.E., Rodrigo, M.M.T.: Hand gesture recognition for smartphone-based augmented reality applications. In: Chen, J.Y.C., Fragomeni, G. (eds.) *HCI 2020*. LNCS, vol. 12190, pp. 346–366. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49695-1_23
8. Covaci, A., Postelnicu, C.-C., Panfir, A.N., Talaba, D.: A virtual reality simulator for basketball free-throw skills development. In: Camarinha, L.M., Shahamatnia, E., Nunes, G. (eds.) *DoCEIS 2012*. IACT, vol. 372, pp. 105–112. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28255-3_12
9. Boninblue Design Laboratory: VR SHOOT AROUND - Realistic basketball simulator, <https://store.steampowered.com/app/671740/>. Accessed 11 Nov 2020
10. Valve Software: SteamVR Unity plugin. https://github.com/ValveSoftware/steamvr_unity_plugin. Accessed 22 Mar 2021
11. Tactic Training. *IEEE Trans. Vis. Comput. Graph.* PP, (2020). <https://doi.org/10.1109/TVCG.2020.3046326>
12. Sauro, J.: A practical guide to the system usability scale: Background, benchmarks & best practices. Measuring Usability LLC (2011)