A VIRTUAL REALITY TRAINING SIMULATION TO ASSIST IN HIGH-FIDELITY BASEBALL BATTING USING OCULUS QUEST 2 AND UNITY ENGINE

Brian C. Xu[1], Robert Gehr[2]

[1]Flintridge Preparatory School, 4543 Crown Ave,
La Canada Flintridge, CA 91011
[2]Computer Science Department, California State Polytechnic University,
Pomona, CA 91768

ABSTRACT

Often, people's busy schedules and lack of equipment make it difficult for them to get solid baseball training hours [1]. This paper seeks to remedy this issue by investigating the efficacy of a virtual baseball training solution [2]. The solution proposed in this paper involves developing a virtual reality baseball simulation aimed to accurately simulate an environment where baseball hitting can be trained in a context that does not require access to expensive baseball equipment or a huge time commitment. The baseball training solution was successfully made in the Unity game engine and deployed to the virtual reality Meta Quest 2 platform. One primary feature of this solution is the pitching mechanism where pitches are thrown to the player accurately [3]. Another feature of this solution is the ability to translate the player's movements in real life to the player's movements in the simulation. The solution was tested in the following experiments: one to test the improvements of player's skills, and one to test the entertainment levels of different age groups. After collecting data, we found that players did improve from our solution, and that little kids/older adults enjoyed our solution more than teens and those in their 20s. Based on the results of the data, we believe that further research in baseball training solutions that utilize virtual simulations would be worthwhile. Future methodologies may improve upon fidelity and accessibility.

KEYWORDS

VR, Baseball, Unity, Simulation

1. INTRODUCTION

Many people do not have access to baseball fields and batting cages or do not have time to go to public training facilities; some students have to work part time and most kids cannot drive on their own. This problem is significant because it not only prevents players from timing pitches, but it also prevents players from swinging the bat and making adjustments based off where the ball goes. Making adjustments is the most important part of hitting because it allows the player to develop their perfect swing and to be able to use that swing consistently. In the long run, this problem will affect many passionate baseball players who want to, but do not have the resources to get better and achieve their goals of playing baseball at the next level [4].

David C. Wyld et al. (Eds): DMSE, CSEIT, NeTCoM, SPM, CIoT, NCS, NLPD, ArIT, CMLA – 2024
pp. 115-125, 2024. CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.141409

Method A tried to improve baseball hitting skills by allowing players to face higher velocity and to see higher pitching speeds, so that when it comes to game time, slower speeds will be more hittable. However, because pitches come out of a machine, it lacks the randomness and deceptiveness that a real baseball pitcher provides. My project tries improving on this by having the VR pitcher generate pitches based on spontaneity: one does not know whether a fastball or an offspeed pitch will be coming. Method B tries using technological training methods to improve baseball hitting. Unlike our method, this method lets players swing a real bat at a baseball that was projected onto a screen. However, because the ball stays on the screen while the player swings in real life, there is a clear lack of depth perception that makes it hard to tell if the player actually hit the baseball. Our method puts both the player and the baseball in the same world, just like how it would be in a real baseball game. Instead of working on the techniques of swinging a bat, Method C focuses on the physical side: lifting weights and conditioning oneself. The problem with this is that without a solid swing, it doesn't matter how strong one is; one will not be able to hit the baseball. On the other hand, my method allows the player to directly perfect their techniques.

Our solution is a virtual reality (VR) baseball hitting simulator that allows players to practice hitting and to be able to face live pitching at home [5]. This solution solves many problems. One, players will not be wasting time sitting in cars going to and back from the batting cages/field. Instead, this lost time could be spent practicing more. Another problem our solution solves is not being able to reserve a field. Because fields are limited and Little League/high school teams get priority when practicing on them, it is hard for those who want extra practice to be able to do so. With our solution, having somewhere to practice is no longer an issue. With our solution, one can practice wherever they want, whenever they want, as long as they have their VR headset and their laptop to connect to. We believe that this solution is effective because it directly addresses many of the key factors that prevent people from practicing as well as provides a way for players to get better.

The first experiment tested if the simulation actually helped the people who used it improve. This experiment was set up in a batting cage, where there was access to machines. Participants first swung according to pitches thrown from the machine. Then, they put on my VR headset and swung according to pitches thrown from the simulation [6]. Lastly, they ended with more pitches from the machine to see if they improved the amount of solid hits that they got. The most significant finding was that on average, every single participant improved in the amount of solid hits that they got. This data is important because it essentially answers the question of "Was my project successful in helping others get better", which the answer is yes. The second experiment tested if the simulation gave joy to those who played it. For this experiment, there were more participants and the age range varied greater for maximum results. From this experiment it was discovered that little kids and older adults tend to have more enjoyable experiences than teens and those in their 20s. This may be the case because all little kids think about having fun, and anything that relates to playing will give them that joy. For teens and those in their 20s, they may have enjoyed the game less because they are very busy with things like school and work, leaving little room left for entertainment. Older adults may have enjoyed the game because they developed memories of their childhood of having fun, and they probably wish to return to them.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. The Pitching System

One major component in my program is the pitching system. I had to consider how I might be able to simulate the behaviors of a real pitcher in my program. This includes the pitcher's delivery as well as the many different pitches that the pitcher has. Something that could potentially cause problems is the fact that in real life, the pitches the pitcher throws are based off of what the catcher and the coach calls for. In a simulation, this is not possible for the simulation runs based on what the code says. A possible solution to this could be to research the order that real pitchers throw pitches in, and then have the game pitcher copy that order.

2.2. Implementing Physical Forces

Another major component in my program is implementing physical forces. There were many forces that I needed to utilize to accurately simulate a baseball pitch [7]. However, many of them weren't included in Unity's physics system. For example, rotational forces and forces generated by differences in air pressure that play a key role in defining the curved trajectory of baseballs as they fly through the air were not normally taken into consideration in Unity's default simulation. As a result, to solve this problem, I had to determine how to integrate new force calculations into the already existing Unity system in order to produce different baseball pitches that had not only realistic behaviors, but that could also be thrown for strikes.

2.3. Connecting the Physical Movement

One final major challenge in my program is connecting the physical movement of the player to the game. I had to consider how real life swings using the VR gear would translate into the bat velocity produced in the game. To solve this issue, I will have to use virtual reality libraries integrated with Unity and make sure that they are able to provide data from the user's controller. It will then be essential to implement this data in a manner that can be interpreted as physical properties for the physics system to accurately resolve collisions between the baseball bat and the baseball.

3. SOLUTION

The main structure of the program can ultimately be described by four systems running in parallel: the player input system, the pitch generation system, the baseball force simulation system, and the bat collision detection system. These four systems run continuously throughout the program's operation and continuously pass relevant output data to each other that is then interpreted uniquely by each system in order for them to successfully fulfill their purposes. The player input system continually gathers sensory input data from the virtual reality apparatus, which is used to update the position and rotation of the player camera and the virtual baseball bat. The player input system also continually outputs relevant position data to be continuously interpreted by the bat collision system. In parallel, the pitch generation system manages a timer and generates a new baseball pitch with unique parameters once that timer reaches zero. The pitch generation system repeats this process continuously and passes on the pitch parameter data to the baseball force simulation system with each pitch generation. Once a pitch has been generated, the baseball force simulation system uses those pitch parameters to continuously simulate the magnus effect and other relevant forces throughout the lifetime of the ball's trajectory until it collides with an object. It is this relationship between inputs and outputs of these continuous systems that define the general process of this program.

The Player Input System - The purpose of this component is to take in the data of the player swinging the bat, and to transfer that data to the computer, so that the player can interact with the simulation. The player input is received through the VR device and the data is handled by Unity's XR toolkit library. From there, the data goes through the other systems of the game, including baseball to baseball bat collision detection and player camera and player hand position adjustments.

This component requires a special concept, which is VR. Instead of pressing keyboard buttons, data is gathered by just moving around one's body.

Figure 1. Screenshot of the VR system

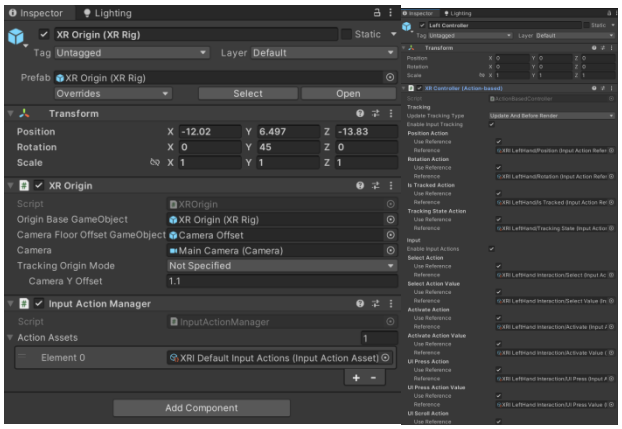

Figure 2. Screenshot of code 1

Fortunately, the scripts inside the XR toolkit library handled all the integrations of the VR headset with Unity [8]. Above are shown three essential scripts relating to the XR toolkit: XR Origin, Input Action Manager, XR Controller (Action-based) [9]. The XR Origin script takes in references to VR related game objects in the 3D scene, including the player's camera and the VR control rig [10]. This allows the important data from these game objects to be passed into the XR origin backend. The Input Action Manager is responsible for defining the relationship between particular input patterns from the player and converting them into particular outputs into the program. This script takes in an Input Action Asset that defines these inputs and outputs. We used the default Input Action Asset, giving expected behavior in the program relating to the player's movements. Lastly, the XR Controller script functioned similarly to the Input Action Manager, holding references to input action data in order to determine the appropriate output with respect to the player's input through the controller.

The Pitch Generation System - The purpose of this component is to facilitate the continuous generation of new pitches to be thrown towards the player. In order to create this system in a simplified way, we distilled the concept of the human pitcher into two simple concepts: a timer and a single action of initiating a pitch. As a self-contained system, the simulated pitcher begins by generating a random value in seconds and decreasing that value over time. Once this value reaches less than or equal to zero, a new pitch action is initiated by the pitcher, and the timer resets to a new random value in seconds. Additionally, the pitch system includes the concept of pitch presets with a random preset being applied to each pitch action. These pitch presets change the nature of the pitch; for example, fastballs, curveballs, and sliders. These pitch preset values are passed into the force simulation system, which is how they take effect on the pitch.

Figure 3. Screenshot of the system 1

```
// Update is called once per frame
Unity Message | 0 references
void Update()
{
    UpdateCountdownTimer();
    UpdatePitch();
}
1 reference
void UpdateCountdownTimer()
{
    if (countdown > 0)
    {
        countdown = countdown - Time.deltaTime;
    }
}

1 reference
void UpdatePitch()
{
    if (countdown < 0)
    {
        animator.SetTrigger("Start Pitch");
        countdown = UnityEngine.Random.Range(minimumTime, maximumTime);
    }
}
```

```
0 references
public void SpawnBaseball()
{
    //Create new baseball and access it's Baseball script
    GameObject newBaseball = Instantiate(baseballPrefab, baseballSpawn.position, UnityEngine.Quaternion.identity);
    Baseball baseballScript = newBaseball.GetComponent <Baseball>();

    if (baseballScript == null)
        baseballScript = newBaseball.GetComponent<TestBall>();

    int profileNumber = UnityEngine.Random.Range(0, pitchProfiles.Count);

    if (!profileOverride)
    {
        baseballScript.InitializePitchData(pitchProfiles[profileNumber]);
        Debug.Log("Pitch Profile " + pitchProfiles[profileNumber].name + " activated");
    }
    else
    {
        baseballScript.InitializePitchData(pitchProfiles[profileSelection]);
        Debug.Log("Pitch Profile " + pitchProfiles[profileSelection].name + " activated");
    }
}
```

Figure 4. Screenshot of code 2

Example one displays the full update loop of the pitcher script. As we can see, the update loop consists of two steps: 1) updating the timer (decreasing over time), and 2) checking when the timer reaches zero, and initiating a pitch if it does, and resetting the timer. This process repeats throughout the duration of the simulation, causing a continuous stream of pitches towards the player. Lastly, in example two, we see another function that lives in the pitcher script that is responsible for spawning a new baseball instance into the simulation and passing the random pitch preset data into the ball. While this function does not appear in the update loop of example one, it is indeed called inside this update loop through the Unity animation system. When animation.SetTrigger("Start Pitch") is executed in the UpdatePitch function, it begins an animation which will eventually call the SpawnBaseball function at the appropriate point in the swing animation using a Unity Animation Event.

The Baseball Force Simulation System - Unlike the continuous pitch generation system, the Baseball Force Simulation System comes into play intermittently when a new baseball is

spawned, and deactivates under certain conditions. The logic of the Baseball Force Simulation System primarily lives inside the baseball script. This script is a component of the spawned baseball object and is where all the pitch data is passed to. The primary function of the baseball script is to update and apply the forces of the baseball based on the unique pitch parameter data that it was given. In order to create a pitch that behaves realistically, the baseball force simulation system accounts for air pressure forces generated by the fast rotation of the ball in the air. These forces are known as the Magnus Effect. For every frame of the ball's trajectory before it collides with an object, a new magnus force and drag force are calculated based off of the pitch parameters, giving realistic and unique pitch trajectories. There are two conditions that terminate an instance of the baseball force simulation system: when the baseball hits a surface or comes into contact with the baseball bat. Once the baseball makes contact with the baseball bat, the special force simulation system is deactivated and the new trajectory is handled by Unity's built in rigidbody system.

Figure 5. Screenshot of the system 2

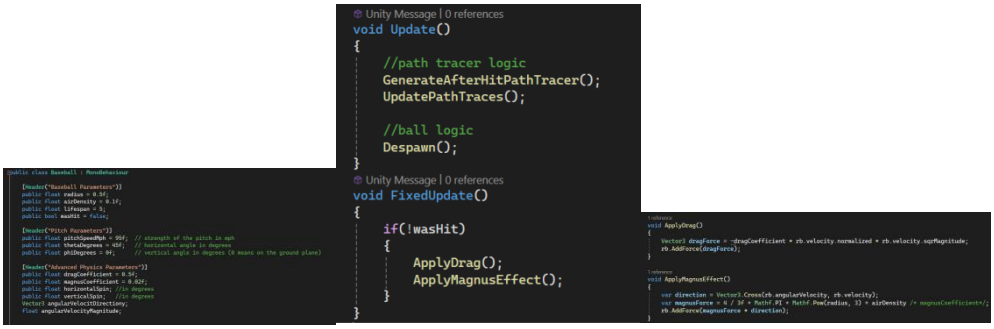

Figure 6. Screenshot of code 3

Example one shows the data associated with a baseball instance. The data is split into three categories: "Baseball Parameters," "Pitch Parameters," and "Advanced Physics Parameters." The Baseball Parameters contain general data about the baseball's properties as they pertain to the force calculations, as well as information regarding the theoretical air density value. The Pitch Parameters represent the obvious information relevant to generating a baseball pitch– the speed of the throw, and the angle at which it is thrown. Lastly, the Advanced Physics Parameters represent the less obvious information relevant to the pitch– taking into account the spin on both axes with which the ball left the pitcher's hand, the angular velocity of the ball, and the coefficients of drag and of the magnus force. All of these properties are taken into account in the calculations of the Baseball Force Simulation's update loop during the lifetime of the baseball. Example two outlines the baseball instance's update loop, which is represented by the contents of the Update() and the FixedUpdate() function. Primarily, drag and magnus forces are calculated

each frame in the FixedUpdate loop using the parameter data described above for as long as the ball is not hit. Example three shows the details of the calculations of the magnus forces, showing how the formulas are calculated pertaining to the parameter data.

The Bat Collision Detection System - The purpose of the Bat Collision Detection System is to translate the movements of the bat through the player's input into data that can be interpreted by Unity's physics system, and then inserting that data into the physics system so that baseball collision are handled by Unity in a way that takes into account the player's real life movements. Ultimately, the Bat Collision Detection System converts the player's movements of the baseball bat from their real world behavior into velocity values that can be applied to colliders along the length of the virtual baseball bat. Once the colliders of the baseball bat have been updated with velocity values that are derived from real world arm movements, the Unity rigidbody system will be able to take that high velocity into account when the simulated baseball collides with the simulated bat, allowing for an accurate force to be applied to the ball in an accurate direction based on the player's swing.

Figure 7. The component

```
Unity Message | 0 references
private void Update()
{
    //update the position
    Vector3 destination = _batFollower.transform.position;
    _rigidbody.transform.rotation = transform.rotation;

    //recalculate velocity
    _velocity = (destination - _rigidbody.transform.position) * _sensitivity;

    //apply velocity
    _rigidbody.velocity = _velocity;

    //update physical position
    transform.position = destination;
    transform.rotation = _batFollower.transform.rotation;
}
```

Figure 8. Screenshot of code 4

In the above code example, we can see the update loop of a single collider on the length of the simulated baseball bat. Through the XR toolkit and by the nature of the baseball bat residing within the hierarchy of the virtual VR controller, the baseball bat's position and rotation are already being updated accurately based on the player's movements. These baseball bat colliders reside under the same hierarchy and are therefore also updated by the player's movements. In order to convert these updated positions into usable data for the collision, we can see that the updated loop is responsible for generating an accurate velocity vector for the capsule by subtracting its previous position vector by its current position vector, effectively giving us the rate of change of the collider's position over time. Once this velocity has been calculated, we

send it to be interpreted by Unity's physics system by setting the rigidbody component's velocity to equal this new velocity.

4. EXPERIMENT

4.1. Experiment 1

The most important reason why I made this program was to create a way to help real life baseball players improve their baseball hitting skills. And so, I decided to create this experiment which would see if my simulation actually helped baseball players improve and get better.

To test whether or not my simulation actually helps baseball players, I will select baseball players at random and take them to a batting cage. There, I will first have the participants swing five times from pitches thrown from the machine to see how many solid hits they got. Then, they will use my simulation and swing five times according to pitches thrown from the machine. Lastly, they will take off my VR headset and swing five more times to the machine to see if the amount of solid hits they got increased, which shows that they improved. The experiment is set up this way because it is able to record both the data without using my VR and the data with using my VR, without either affecting each other.

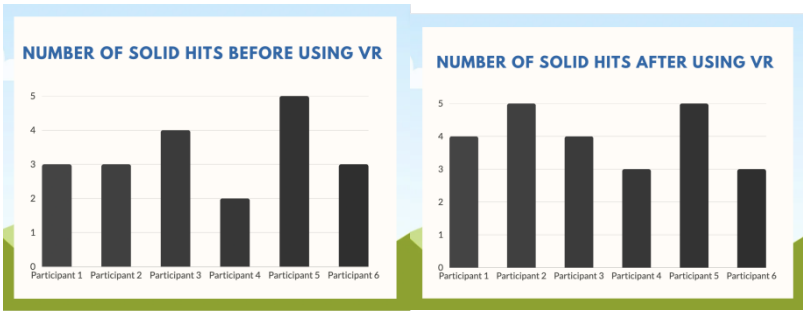

Figure 9. Figure of experiment 1

From my data, it can be seen that most of the participants improved with the usage of my VR simulation. The lowest number of solid hits before VR was two, and that number moved to three after using my VR simulation. There was one really good baseball player who had five solid hits before using my VR, and that number stayed the same after using my VR, so it was hard to tell whether or not the VR helped him. However, from the other results, I am confident that my VR baseball simulator is able to help baseball players improve their batting skills.

4.2. Experiment 2

The second reason for why I made this program was to help make the game of baseball more fun for kids and even adults. Instead of having value only as a training tool, I intend for this program to also have value as a source of entertainment. With this program, people can try something new in their lives, and maybe even develop a passion for baseball.

To test whether or not my game is entertaining and gives joy to people who play it, I will experiment with five age demographics: ages 7-12, ages 13-18, ages 19-25, ages 26-35, and ages 36-50. All the participants will have no background in baseball so that my data is not biased. Furthermore, I will have each participant fill out a questionnaire at the end. The questionnaire will ask them if they had fun playing and if they were able to enjoy their time. To prevent fake

answers in the questionnaires, I will use a lie detector on each participant when they are filling it out. By using a lie detector, I am able to produce 100% accurate data.

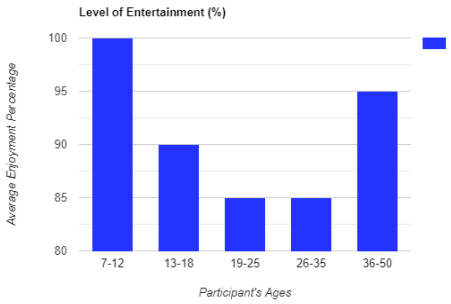

Figure 9. Figure of experiment 2

In the data, the average enjoyment percentage across all ages was 91%. The participants aging from 7-12 had the most fun, while the participants aging from 19-25 and 26-35 had a lower percentage. What surprised me the most was that the age group 36-50 had the 2nd highest percentage. I didn't expect older adults to enjoy a game made for kids, but the data proved otherwise. I believe that this just shows that everyone still wishes to have fun, no matter what life they are living.

5. RELATED WORK

The solution that this scholarly source gives to getting better at hitting a baseball is to face higher pitching speeds as well as to just track a fast pitch [11]. By doing this, one can get used to faster pitching speeds so that slower speeds in real games will seem easier to hit. Its limitations are that the pitches come out of a machine, which is completely different from pitches that a pitcher throws; the release point, the motion, the craftiness are all different. Something that my project has that a machine doesn't is the spontaneity of when the pitch will come out and the type of pitch as well. For a machine, one needs a second person to put the ball in, and settings need to be adjusted for different pitches to be thrown out. However, with my app, both of these problems are solved for you.

This paper attempts to record the effectiveness of technological training methods to improve baseball skills [12]. In the study, the solution to this goal is to utilize VE (Virtual Environment) into the training of a select group of baseball players. The solution was for the participants to swing a baseball bat at a simulated projected baseball that was projected onto a screen. The results were that the participants who trained under the VE showed significantly greater improvements as well as reached higher levels of competition. Limits of this solution include not being able to see the bat come into contact with the baseball, and not being able to determine the depth perception of the ball in relation to the player. For my project, because the player is also in the VR world, the player's swings do in fact come into contact with the pitch. Furthermore, because the player's eyes are the eyes of the batter in the VR world, it is easy to track the pitch, just like how one would track a baseball pitch in a real baseball game.

This paper tackles the same problem that I do: how to improve baseball skills [13]. This paper resorted to developing novel physical strength and conditioning methods, rather than developing a technology-based training solution. They created a whole offseason workout routine to build strength, which in turn, translates over to baseball in hitting the ball harder and pitching the ball

faster. This solution is effective when the training regiment is implemented. However, because this solution focuses only on building strength and not the actual techniques that baseball players need to succeed, they will see improvements based off of their physical strength and endurance only, instead of from their techniques and skill. My paper, on the other hand, focuses on the techniques and the real-life skills of hitting a baseball, instead of the physical training on the body.

6. CONCLUSIONS

The first limitation of my project is how the pitcher is always accurate - in a bad way. Real life pitchers are human, and make mistakes from time to time. However, my simulation does not take this fact into account. All of the pitches that the pitcher throws in my simulation are strikes and each one lands in the exact same spot every single time. I believe that I need to add more variants to pitches and more room for error. For example, I would like to mix in pitches from time to time that are actually balls and that the batter should not be swinging at. Furthermore, in addition to adding "mistake" pitches, I also want to add randomness to the pitches, in order to make each pitch type land in a new position every time, making the batter have to adapt in order to do well. Another limitation of my project is that the tactile sensation of playing baseball in my simulation does not match the sensation of playing baseball in real life. First off, the weight of the controller is not similar to the weight of the baseball bat [15]. This difference is significant because it makes swinging the virtual controller very easy and light, when in real life, swinging a 30 - 33 oz bat is slower and takes more effort. Secondly, because the baseball is virtual, one does not feel the contact point, and thus does not experience the feeling of squaring up a baseball. One last limitation of my project is that it is not as immersive as real life, which may distract the player because the player is in a different environment. An easy solution to this would be to just develop the VR world so that it looks more realistic, either by implementing realistic features, or by adding more details to make each component look better.

In conclusion, a training mechanism for baseball hitting practice was successfully created using the Unity game engine and a VR headset. Based on the data of the experiments discussed above, this training mechanism was proven to be effective. Those who trained with this VR training solution showed improvements to real world baseball performance compared to those who did not, which suggests that more developments towards training systems of this nature should be pursued [14]. While the training system developed in this paper was successful, it leaves many areas for future improvements in the realm of immersion, realism, and tactile sensation. Additionally, there are some inherent limitations, such as the cost of equipment, that limit the accessibility of this solution. As future solutions improve upon this one, accessibility would be an important point of consideration.

REFERENCES

[1] Newton, Robert U., and Kerry I. McEvoy. "Baseball throwing velocity: A comparison of medicine ball training and weight training." The Journal of Strength & Conditioning Research 8.3 (1994): 198-203.

[2] Gray, Rob. "Transfer of training from virtual to real baseball batting." Frontiers in psychology 8 (2017): 2183.

[3] Fleisig, Glenn S., et al. "Kinetics of baseball pitching with implications about injury mechanisms." The American journal of sports medicine 23.2 (1995): 233-239.

[4] Rottenberg, Simon. "The baseball players' labor market." Journal of political economy 64.3 (1956): 242-258.

[5] Anthes, Christoph, et al. "State of the art of virtual reality technology." 2016 IEEE aerospace conference. IEEE, 2016.

[6] Desai, Parth Rajesh, et al. "A review paper on oculus rift-a virtual reality headset." arXiv preprint arXiv:1408.1173 (2014).

[7] Fortenbaugh, Dave, Glenn S. Fleisig, and James R. Andrews. "Baseball pitching biomechanics in relation to injury risk and performance." Sports health 1.4 (2009): 314-320.
[8] Gomes, Arlindo, et al. "Extended by Design: A Toolkit for Creation of XR Experiences." 2020 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct). IEEE, 2020.
[9] Kern, Florian, et al. "3D printing an accessory dock for XR controllers and its exemplary use as XR stylus." Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology. 2021.
[10] Saxena, Ashutosh, Min Sun, and Andrew Y. Ng. "Make3d: Learning 3d scene structure from a single still image." IEEE transactions on pattern analysis and machine intelligence 31.5 (2008): 824-840.
[11] Kohmura, Yoshimitsu, et al. "Effects of batting practice and visual training focused on pitch type and speed on batting ability and visual function." Journal of Human Kinetics 70.1 (2019): 5-13.
[12] Gray, Rob. "Transfer of training from virtual to real baseball batting." Frontiers in psychology 8 (2017): 2183.
[13] Klein, Brooks, et al. "Offseason workout recommendations for baseball players." Current Reviews in Musculoskeletal Medicine 14 (2021): 174-184.
[14] George, Thomas R. "Self-confidence and baseball performance: A causal examination of self-efficacy theory." Journal of sport and exercise psychology 16.4 (1994): 381-399.
[15] Nathan, Alan M. "Characterizing the performance of baseball bats." American Journal of Physics 71.2 (2003): 134-143.