VR Tennis Training Game

Realistic Tennis Experience

YIQING WANG, IRIS ZHAO, and JAYLYN ZHANG, University of Washington

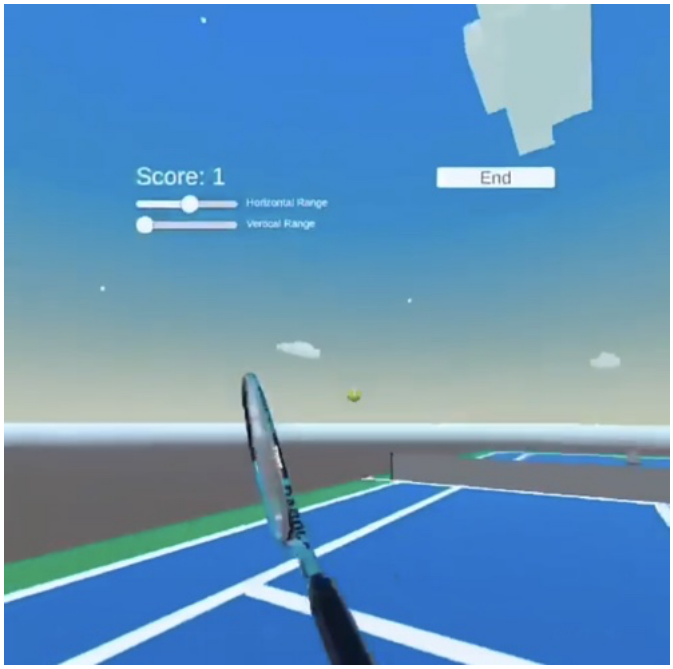

Fig. 1. A first-person VR view of our tennis training game in action.

This project presents a VR Tennis Training Game on Meta Quest 2, designed to provide users with a realistic tennis experience. Motivated by the potential of VR to enhance accessibility to sports training, our work focuses on the modeling of ball physics and racket interactions, carefully tuned to replicate authentic tennis gameplay dynamics. The results demonstrate a compelling virtual tennis experience with accurate ball trajectory and responsive gameplay. Future work may explore advanced stroke analysis and adaptive difficulty based on player performance, potentially expanding the application's use from casual entertainment to structured training.

1 INTRODUCTION

Virtual reality (VR) has rapidly evolved to deliver highly immersive experiences, with VR sports emerging as a leading application. Tennis, a sport that demands precise motor skills, spatial awareness, and quick reflexes, presents unique challenges in VR implementation. This project aims to overcome these challenges by developing a physics-accurate tennis simulator for the Meta Quest 2, bridging the gap between virtual practice and real-world play. By leveraging VR, we make tennis more accessible, removing barriers such as court availability, weather conditions, and equipment costs.

Existing VR tennis applications, such as Tennis League VR, have demonstrated the viability of this concept. We drew inspiration from these experiences, extracting the most valuable components to create a minimal and streamlined game focused on recreational tennis training. This project marks our first exploration of the Unity development pipeline for the Meta Quest 2, with an emphasis on achieving a realistic and immersive tennis experience.

Through our development process, we found that incorporating physics formulas, fine-tuning constants, and selecting the right Unity components were crucial in achieving a realistic and immersive gameplay experience. Additionally, advanced collision detection, particularly in accurately simulating spin, required extra effort to refine and make visually evident in the game. For future work, we can further enhance the complexity of the physics calculations, integrating more advanced mechanics and stroke analysis to elevate gameplay realism. These improvements will enable more precise shot feedback, making the VR tennis experience even more engaging as a training tool.

1.1 Contributions

We believe that the following contributions were made in our project.

- Contribution 1. Realistic Physics Implementation. We applied physics formulas and fine-tuned constants to achieve an immersive and accurate tennis simulation.

Authors' address: Yiqing Wang, ywang46@cs.washington.edu; Iris Zhao, jiyunz@cs.washington.edu; Jaylyn Zhang, jiali7@cs.washington.edu, University of Washington.

- Contribution 2. Enhanced Collision Detection. By implementing ray-casting, we ensured precise collision detection, even at high ball and racket speeds.
- Contribution 3. Future Expansion Potential. We established a foundational framework for ongoing development of the tennis game on the Meta Quest platform.

2 RELATED WORK

One of the most relevant commercial VR tennis games is Tennis League VR, available on the Meta Quest 2. While the source code for this game is not open, we were able to analyze gameplay through available videos and reviews. Tennis League VR offers a variety of features, including multiplayer capabilities, customizable avatars, and different game modes. The game's smooth gameplay and diverse functionalities provided a valuable reference for our own development, though we were unable to directly access or build upon their codebase due to its proprietary nature.

3 METHOD

3.1 Collision Detection

Our initial implementation relied on Unity's built-in Colliders attached to the ball and racket, configured with continuous dynamic collision detection to handle the interaction between these moving rigidbodies. However, this approach proved inadequate. In more than 50% of instances, the ball would pass directly through the racket mesh without triggering collision detection, particularly during rapid swings.

To resolve this issue, we implemented a ray-casting solution. This technique projects an infinite line along the ball's velocity vector. As the ball moves, the ray travels with it and detects intersections with any collider in its path, and it calculates the surface normal at the point of collision. This approach dramatically improved our collision detection reliability, successfully registering interactions even during high-speed swings.

3.2 Ball-Ground Bounce Behavior

A tennis ball usually bounces on the ground before a hit. To achieve natural physics, we implemented reflection based on surface normals and applied a velocity reduction factor of 0.8 to simulate friction and energy loss during bounces. This coefficient was determined through experimental testing to achieve the most realistic feel.

During development, we encountered an unexpected inconsistency. The ball bounced at different heights when viewed in VR compared to the Unity editor game mode, despite our constant ball launch velocity. After thorough investigation, we identified the source of this discrepancy as a frame rate synchronization issue. The Meta Quest 2 headset operates at 90 Hz, corresponding to a time step of approximately 0.0111 seconds per frame (1/90=0.0111s). We resolved this issue by adjusting Unity's fixed timestep parameter to match the Quest's display frequency, ensuring consistent physics calculations across both environments and eliminating the height variation.

3.3 Ball-Racket Bounce Behavior

To achieve realistic ball-racket interaction, we decomposed the post-collision ball velocity calculation into two components. The first component is the reflected velocity, calculated based on the incoming ball velocity and the surface normal at the point of impact. The second component accounts for the velocity transfer from the racket to the ball during contact.

Through extensive testing, we found that an equal weighting of these two components produced the most natural and responsive gameplay feel. As a result, the exit velocity of the tennis ball is calculated using the following formula:

$$\vec{v}_{\text{exit}} = 0.5 \cdot \vec{v}_{\text{reflected}} + 0.5 \cdot \vec{v}_{\text{racket}}$$

This balanced approach ensures that both the angle of incidence and the player's racket movement significantly influence the resulting ball trajectory, creating an responsive control experience.

3.3.1 Racket Velocity. We calculate the racket velocity using the following formula:

$$\vec{v}_{\text{bat}} = \frac{\vec{p}_{\text{current}} - \vec{p}_{\text{previous}}}{\Delta t}$$

3.3.2 Ball Spin and Magnus Effect. To simulate the ball's spin after impact, we compute the spin axis and the spin amount based on the velocity of the ball at the point of impact. The direction of the spin is calculated by finding the cross product of the hit surface normal and the velocity vector. The spin amount is determined using the magnitude of the hit velocity, the angle between the velocity vector and the normal, and a multiplier that controls the spin intensity.

The spin axis is calculated as:

$$\vec{\omega}_{\text{spin}} = \frac{\vec{v}_{\text{hit}} \times \vec{n}}{|\vec{v}_{\text{hit}} \times \vec{n}|}$$

where $\vec{v}_{\text{hit}}$ is the velocity of the ball at the point of impact.

The spin amount is calculated as:

$$\text{spinAmount} = |\vec{v}_{\text{hit}}| \cdot \sin(\theta) \cdot \text{spinMultiplier}$$

where θ is the angle between the hit velocity $\vec{v}_{\text{hit}}$ and the surface normal $\vec{n}$.

Finally, we apply the spin to the rigidbody using torque:

$$\vec{\tau} = \vec{\omega}_{\text{spin}} \cdot \text{spinAmount}$$

To simulate the Magnus effect, which influences the trajectory of a spinning object moving through air, we calculate the force exerted on the ball based on its angular velocity and velocity. The Magnus force is given by:

$$F_{\text{Magnus}} = \frac{4}{3}\pi\rho r^3 |\vec{v}| \vec{\omega} \times \vec{v}$$

where ρ is the air density; r is the radius of the ball; $|\vec{v}|$ is the velocity magnitude; $\vec{\omega}$ is the angular velocity; $\vec{v}$ is the velocity vector of the ball.

Although we were confident in the theoretical foundation, the spin was not evident during gameplay, even after adjusting the spinMultiplier value through experimentation. Debugging this issue was particularly challenging, as we could not print out debug values while playing in the headset. As a result, we have marked this as a task for future work to address.

4 IMPLEMENTATION DETAILS

We developed our VR tennis game using the Meta Quest 2 headset and its accompanying controllers. To enable interaction within the virtual environment, we leveraged Unity's XR Interaction Toolkit, which provides a robust framework for handling VR input and interactions.
Unity Assets we used:
- Unity Asset Store "Free Sports Kit"
- Unity Asset Store "Simple Sky"

4.1 Feature: Ball Machine Shooting Randomization

To enhance the challenge of the ball machine, we introduce random horizontal and vertical offsets to the ball's shooting direction. These offsets can be dynamically adjusted during gameplay using a UI slider, allowing for real-time customization of the difficulty.

4.2 Feature: Scoring Mechanism

The overall scoring logic is that the score increments by one every time the player successfully hits the ball into the legal area. We used a collider to detect whether the ball landed in the scoring area. More specifically, we implemented a boolean variable to check if the ball landed after being hit by the player and another boolean field to ensure the score increments only once per hit.

4.3 Feature: Start And End Game UI

To create a seamless user experience, we designed an intuitive start and end game UI that allows players to engage with the game efficiently. The implementtation ensures that users can easily start a session, track their performance, and view results after completing a game. The UI elements are designed with World Space Canvas allowing them to appear naturally in the environment. Interactions are implemented using Unity's XR Interaction Toolkit, which enables intuitive hand tracking and controller-based interaction. Player can point at buttons using their controllers and select them via trigger input.

5 EVALUATION OF RESULTS

5.1 Collision Detection Accuracy

To access the reliability of our collision detection system, we recorded instances where the ball correctly interacted withthe racket versus instances where it failed to register a hit. Using our ray-casting approach, the accuracy of collision detection improved significantly compared to Unity's Build-in collision detection:

- Initial Unity Collider-based Approach: 40-50% collision registration success rate.
- Ray-casting Implementation: more than 90% collision registration success rate.

This demonstrates that our ray-casting method effectively mitigated the issue of missed racket-ball interaction, particularly during high speed swings.

5.2 Ball Physics and Realism

To evaluate the realism of ball physics, we compared our bounce mechanics to real-world tennis ball behavior by measuring the height of the bounce and the reduction in speed upon impact with the ground. Our velocity reduction factor (0.8) produced bounce patterns consistent with actual tennis play. The velocity reduction after bounce aligned with the expected energy loss due to surface friction. However, spin dynamics remained a challenge. Despite implementing spin calculations, the Magnus effect was not visually apparent in the game.

5.3 User Feedback and Playability

We conducted informal user testing with a group of five players with varying levels of tennis experience. Feedback highlighted several strengths and areas for improvement.
Strengths:

- Players found the ball physics realistic, especially regarding the bounce dynamics.
- The collision detection system felt accurate and responsive.
- The adjustable ball machine difficulty provided a customizable training experience.

Limitations:

- The spin effect was not visually noticeable.
- Some users reported a learning curve in accurately timing their swings due to lack of haptic feedback.
- The lack of multiplayer support limited the game's long-term engagement potential.

Summary: Our results indicate that the VR Tennis Training Game successfully delivers a realistic and engaging player experience, particularly in ball physics and collision accuracy. However, improvements in spin simulation, haptics, feedback, and multiplayer features could enhance immersion and usability further.

6 FUTURE WORK

Our ball spin algorithm did not perform as expected. One major challenge we encountered was the inability to print debug logs to the Unity console while running the game in VR. This limitation made it difficult to diagnose issues related to spin dynamics. To address this, future iterations of our project could integrate a VR-compatible debugging console that displays real-time logs within the headset. By printing the torque direction upon collision, we can determine whether the issue lies in the direction or magnitude of the applied force. This improvement would provide deeper insights into the spin mechanics and enable more precise adjustments to achieve realistic ball behavior.
Additionally, we could enhance the game with advanced features, such as improved stroke analysis by leveraging machine learning for spin prediction and ball trajectory estimation. This would enable more accurate feedback on player performance and technique. Furthermore, we could implement an adaptive ball machine that dynamically adjusts difficulty based on the player's skill level, creating a more personalized and engaging experience.

7 CONCLUSION

In this project, we developed a VR Tennis Training Game that simulates realistic tennis gameplay using Unity's physics engine and VR capabilities. Our implementation focused on accurate collision detection, realistic ball physics, and smooth performance to provide

an immersive training experience.

Through evaluation, we confirmed that our ray-casting-based collision detection significantly improved accuracy, addressing issues with missed racket-ball interactions in high-speed swings. Our physics calculations produced bounce dynamics closely aligned with real-world tennis ball behavior, though improvements in spin visualization remain necessary. Performance testing demonstrated that our system runs efficiently on the Meta Quest 2, maintaining a high frame rate with minimal latency.

User feedback indicated strong engagement with the training mechanics, though some limitations, such as the absence of haptic feedback and limited spin effects, were noted. These insights provide a foundation for future enhancements, including more advanced physics modeling, multiplayer functionality, and improved feedback mechanisms.

Overall, our VR Tennis Training Game successfully achieves its goal of providing an engaging and realistic training environment for players. Future work can further refine realism and interactivity, making it a more effective tool for tennis training and skill development.

ACKNOWLEDGMENTS

We thank our course instructors and TAs for their guidance and feedback, which helped refine our physics modeling and gameplay mechanics. We also appreciate our testers for their valuable insights in evaluating the VR Tennis Training Game. Lastly, we acknowledge the University of Washington for providing resources that supported our development.

REFERENCES

- FPS Full Game Tutorial
- How to fix Pink Materials in Unity"
- Oculus Quest2 set up guide