Magnus Effect In Video Games

A thesis submitted for the degree of

COMPUTER GAMES TECHNOLOGY

by

AKSHAT AGRAWAL

Computer Games Technology

School of Design and Informatics

Abertay University

September, 2023

Declaration

Candidate's declarations:

I, AKSHAT AGRAWAL, hereby certify that this thesis submitted in partial fulfilment of the requirements for the award of COMPUTER GAMES TECHNOLOGY, Abertay University, is wholly my own work unless otherwise referenced or acknowledged. This work has not been submitted for any other qualification at any other academic institution.

Signed [candidate's signature]: Akshat Agrawal

Date : 12/09/2023

Supervisor's declaration:

I, Dr. JAMES , hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of COMPUTER GAMES TECHNOLOGY in Abertay University and that the candidate is qualified to submit this thesis in application for that degree.

Signed [supervisor signature] . . . . . .

Date . . . . . .

Certificate of Approval

I certify that this is a true and accurate version of the thesis approved by the examiners, and that all relevant ordinance regulations have been fulfilled.

Supervisor . . . . . .

Date . . . . . .

Acknowledgements

I would first like to thank my thesis advisor Dr. James Therlfall of the School of Design and Informatics at Abertay University. The door to Prof. Threlfall's office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this thesis to be my own work but steered me in the right direction whenever he thought I needed it.

I would also like to acknowledge Mr. Gareth Robinson of the School of Design and Informatics at Abertay University as the second reader of this thesis, and I am gratefully indebted to his very valuable comments on this thesis.

Finally, I must express my very profound gratitude to my parents and my brother for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Abstract

This project assesses the necessity of a mathematically accurate model of the Magnus effect in video games. A spinning ball experiences a force perpendicular to the direction of its motion. This force is called the Magnus force, and the associative effect is called the Magnus effect. The project builds a model of the Magnus effect in Unity 3D software. The model was built as a tennis simulation. This will provide a precise and realistic trajectory of a ball, which is an essential component of sports video games and simulation. To verify the accuracy of our model it is compared against data from other mathematical models. It is found that the model efficiently provides an accurate representation of ball trajectory under certain assumptions.

Keywords

Magnus effect, Magnus Force, Tennis simulation, Unity model.

Table of Contents

List of Figures

Chapter 1

Introduction

The video games industry is one of the leading consumers of modern computer technologies. To create the gameplay experience, video games use a wide range of game mechanics. Game mechanics are the rules and systems that govern how a game functions. These mechanics can include things like movement, combat, inventory management, scoring, and more. Every game uses aspects of mathematical models to incorporate game mechanics. With the evolution of technology, game engines have also evolved to handle more complicated mathematical systems. This in turn provides the players with an immersive and physically realistic experience.

In contemporary simulator games or sports games, players expect a high degree of realism. This can be achieved when the mathematical system for game mechanics takes into account all or as many as possible real-world laws of physics, that have a significant impact on the gameplay experience. In sports games involving the use of a spinning object, a certain phenomenon can be seen where this spinning object traveling in a fluid with some velocity experiences a lateral force perpendicular to its direction of motion. This effect on spinning objects is called the Magnus effect.

1.1 Motivation

The Magnus effect is a fascinating phenomenon observed when a spinning object interacts with a fluid medium, resulting in a deviation in its trajectory. This effect plays a crucial role in various sports, especially those involving ball games like soccer, tennis, and baseball. Understanding and harnessing the Magnus effect can significantly impact the performance and strategy of athletes, leading to improved gameplay and enhanced outcomes.

The Magnus effect is important due to its widespread applications, not only in sports but aerodynamics research, engineering, and various industries. It offers valuable insights into fluid dynamics, motion, and rotational phenomena, impacting diverse fields and inspiring new technologies and scientific discoveries. One such example is the Darrieus wind turbine or vertical-axis wind turbine (VAWT) (Fig 1.1). VAWTs can capture wind from any direction, operate with lower noise levels, require less space, and have easier maintenance, making them suitable for variable wind conditions, urban environments, and constrained spaces. The design of this turbine was patented by Georges Jean Marie Darrieus, a French aeronautical engineer, who filed for the patent on October 1, 1926 (Wikipedia contributors, 2023b). A study by Khadir and Mrad (2015) investigates the contribution of the Magnus effect in VAWTs, which provides the best configuration for better power efficiency than Horizontal-axis wind turbines (HAWT). Understanding and harnessing the Magnus effect can lead to improved performance and increased efficiency of such applications.

Although the concept of the Magnus effect in sports gained popularity in 1997 with the famous Banana kick of Roberto Carlos (Dream team, 2017), it has not been incorporated in video games till fairly recently. Even AAA companies like EA Sports, which spend hundreds of millions of dollars in development, introduced the Magnus

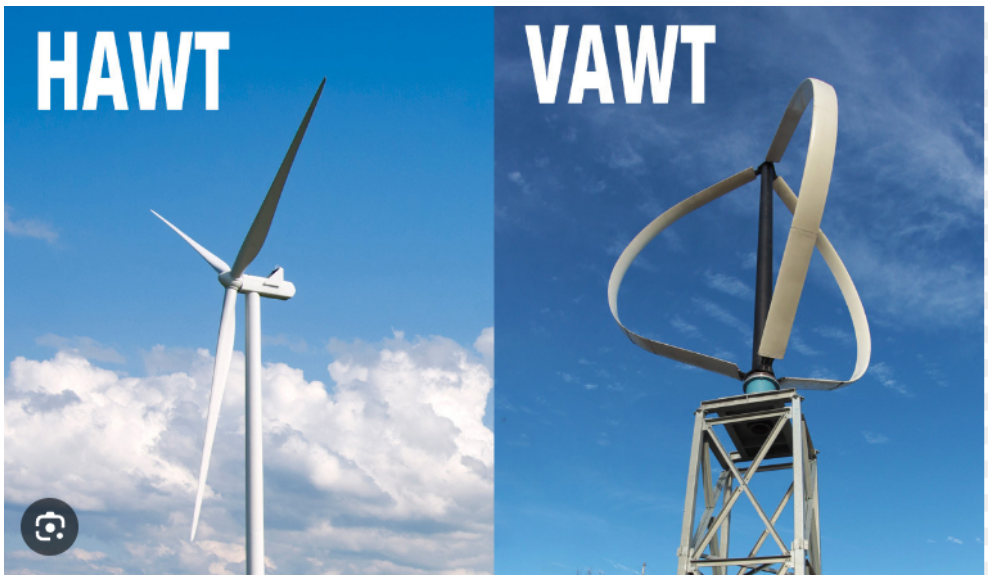

Figure 1.1
A figure (Nigam, 2019) showing the conventional turbine (HAWT) and modern turbine (VAWT).

effect in their FIFA series in 2014. Similarly, the concept of hitting the ball with topspin in Tennis (Fig 1.2) has been an important aspect of the sport since the 1970s and has only increased in importance with time (Tennis, 2023). When looking at men's tennis today, the player that has the highest spin rate on his forehand is Rafael Nadal. It is measured that the Spaniard could reach a topspin rate of 5000 rpm in his hitting. The sport has really developed over the last decade, as in the past people such as Pete Sampras and Andre Agassi hit their shots with an average of around 1800 rpm (Scholte, 2017).

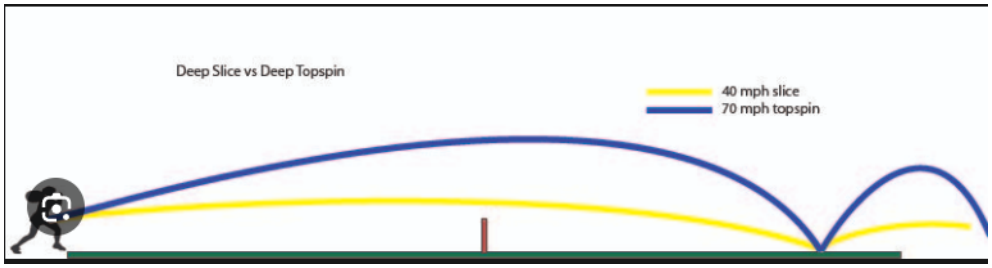

Figure 1.2
Topspin (blue) vs. slice Trajectory (yellow): The Magnus Effect tends to drag topspin balls down and lift slice balls, flattening their trajectory. A flat ball's trajectory is between these two (Phelps, 2023).

If we leave the AAA companies, Indie companies and small sports games still do

not implement the Magnus force in their games. For example, Tennis Clash (Wildlife studios, 2019) is a multiplayer online game developed by Wildlife Studios. It is one of the most famous mobile games for tennis. It has 100 million plus downloads and is considered one of the most difficult and advanced tennis games. But when taking a close look at the gameplay, it is noticeable that the game does not have mechanics for curving the ball. A quick search in the community forum of Tennis Clash suggests that the community wants to use topspin, backspin etc. in the game. This confirms that an easy-to-implement and efficient Magnus effect model would be useful for Indie sports games.

1.2 Thesis Aim

- **Theoretical Analysis:** Conduct a comprehensive literature review to understand the underlying principles of the Magnus effect and its relevance in sports. Study and analyze the factors affecting the Magnus effect. In regard to the physical aerodynamic states, there are numerous experimental conditions investigating the dependencies of lift and drag coefficients, such as the spin, the velocity, the shape, and so on (R. D. Mehta, 2001). For instance, R. Mehta (2008) observed that the drag and the lift coefficients are increased by larger spin rates for tennis balls. A similar observation was reported in C. Guzman (2016) for an American football. Reynolds numbers effect on a rotating golf ball in calm water were recorded in A. Kharlamov (2016). Baseball experiments were given in G.J.E. Santos (2019).

- **Experimental Setup:** Develop a computational model using numerical simulations to predict the behavior of spinning balls under different conditions. Validate the simulation results with experimental data, providing a powerful tool for further investigation and optimization. The model will be built in the Unity Game engine (Unity tech, 2005). Measure and test relevant parameters such as spin rate,

launch velocity, and trajectory deviation against experimental data.

- **Verification against external data:** The next step is to verify the results of our model to determine the accuracy and efficiency of the model. This will be done by collecting data from previous research experiments with similar setups. For example, Cross (2011) built a mathematical model to compare the trajectories of tennis balls under the influence of topspin. Comparing the results of our model with such experiments will provide valuable insights into the performance of our model.

- **Performance:** Since one of the goals of the project is to help Indie companies and small game developers provide a useful starting point to implement the Magnus effect in their games, the model needs to be efficient in performance. This means that the model should provide accurate results while using much less processing power. So, we will check the performance of our model by calculating the processing time and frame rates.

1.3 Thesis Contribution

Through the simulation built for this project, users will be able to explore and understand the behavior of spinning objects in response to aerodynamic forces. Although the model will be built as a tennis simulation, the algorithm depicts the working of the Magnus effect for any spinning object, so it can be used to understand the behavior of the Magnus effect in fields including fluid dynamics, aerospace, defense, etc (Seifert, 2012). The simulation will offer user interaction, allowing players to control parameters such as spin rate, ball velocity, and environmental conditions, observing the direct impact on the spinning object's trajectory. Since the simulation is built in Unity, it will provide a 3D visualization of the trajectory of the ball under the specified conditions, which further enhances the learning experience. The project seeks

to serve as an educational tool, offering insights into the physics behind the Magnus effect for students, educators, and sports enthusiasts. Since the core of the model is an algorithm, the model can easily be made compatible with other platforms. This and the easy implementation of our model will make sure that this model can be used by indie developers and other game developers looking to incorporate the Magnus effect in their games. The project can also be used as study material for companies looking to incorporate the Magnus effect in their games. Additionally, exploring real-world applications of the Magnus effect in sports training and equipment design will showcase its practical significance. Through comprehensive documentation, tutorials, and community engagement, the simulation in Unity aspires to be an informative, engaging, and fun resource for understanding the intriguing Magnus effect in sports and physics

Through this project, we aspire to bridge the gap between science and sports, showcasing the beauty of physics in action on the playing field. Ultimately, we believe that the knowledge gained from this project will unlock new possibilities for sports performance and contribute to the evolution of sports equipment technology.

Chapter 2

Literature Review

2.1 The Magnus effect

The Magnus effect is a phenomenon that occurs when a spinning object, such as a ball or a cylinder, experiences a sideways force, perpendicular to the direction of motion. This effect is caused by a difference in air pressure on opposite sides of the spinning object (Lukerchenko et al., 2012). The Magnus effect is responsible for the curving path of a spinning ball in sports such as baseball, cricket, and tennis. It is also used in some engineering applications, such as helicopter rotors and wind turbines, to generate lift and improve efficiency.

In fig 2.2 (Garcia, 2018), we can see a spinning ball traveling towards the right in the presence of air. The relative velocity of air at a point just above the top side of the ball is decreased since the air molecules colliding with the ball at this point experience resistance. This decrease in the relative velocity of air creates an area of high pressure. This fact comes from Bernoulli's principle (Wikipedia contributors, 2023a), which states that an increase in the speed of a fluid occurs simultaneously with a decrease in static pressure and vice versa. However, on the opposite side, the relative velocity of

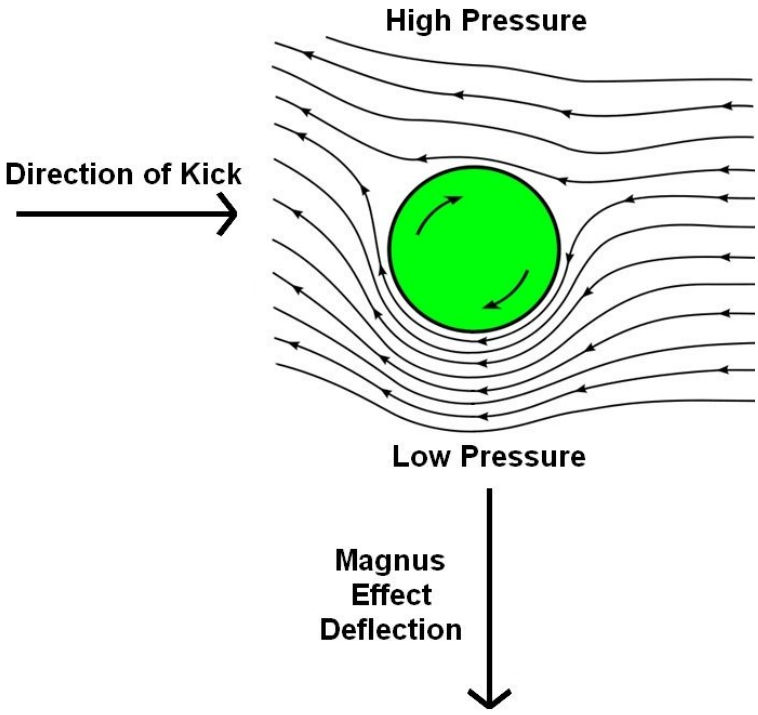

Figure 2.1
A spinning object traveling in the presence of air experiences a force perpendicular to the direction of its motion. (Image source: Garcia, 2018)

air is higher. The air molecules colliding with the ball at the point just below the ball experience a push in the direction of flow. This increase in relative velocity creates an area of low pressure, again using Bernoulli's principle. This pressure gradient creates a force that deflects the ball sideways as seen in Fig 2.1

The Magnus force for a spinning ball in any fluid is the difference in pressure between opposing sides of the ball scaled by the cross-sectional area. Mathematically the Magnus force F_M can be expressed as (Wikipedia contributors, 2023d):

$$F_M = \Delta p \cdot A = c_A \cdot \frac{\rho}{2}(v_1^2 - v_2^2) \cdot A \tag{1}$$

where,

- $\Delta p = c_A \cdot \frac{\rho}{2}(v_1^2 - v_2^2)$ is the pressure difference between opposite sides of the

ball.

- c_A is a scalar dependent on the shape and material of the rotating ball.
- v_1, v_2 are speeds of fluid relative to each surface at the top and bottom sides of the ball, respectively.
- ρ is the density of the fluid.
- A is the cross-sectional area of the ball.

The simplest case is when we consider the ball to be smooth. The resulting equation for a smooth ball takes the form (Juliana Javorova1, 2018):

$$F_M = \frac{1}{2} \cdot C_M \cdot \rho \cdot A \cdot v^2 \cdot \frac{(\vec{\boldsymbol{\omega}} \times \vec{\mathbf{v}})}{|(\vec{\boldsymbol{\omega}} \times \vec{\mathbf{v}})|} \tag{2}$$

Here, $A = \pi r^2$ is the cross-section area of the ball. C_M is the Magnus coefficient. $\vec{\boldsymbol{\omega}}$ is the angular velocity of spinning object, and $(\vec{\boldsymbol{\omega}} \times \vec{\mathbf{v}})$ denotes the cross-product between the two.

2.2 Tennis

The Magnus effect in tennis is a phenomenon that affects the flight path of a tennis ball due to its spin. When a tennis ball is hit with topspin or backspin, it experiences a difference in air pressure on its upper and lower surfaces, causing it to curve or dip during its trajectory.

- Topspin: When a tennis player hits the ball with topspin, the ball rotates forward, creating a lower air pressure on top and a higher air pressure underneath. As a result, the ball experiences an upward force, which helps it stay in the air longer and curve downward when it bounces (Fig).

- Backspin: Hitting the ball with a backspin means it rotates backwards, generating a higher air pressure on top and a lower air pressure underneath. The backspin causes the ball to slow down its forward motion and lift higher, landing farther than expected. Backspin shots are often used in defensive situations, such as lobs or defensive slices.

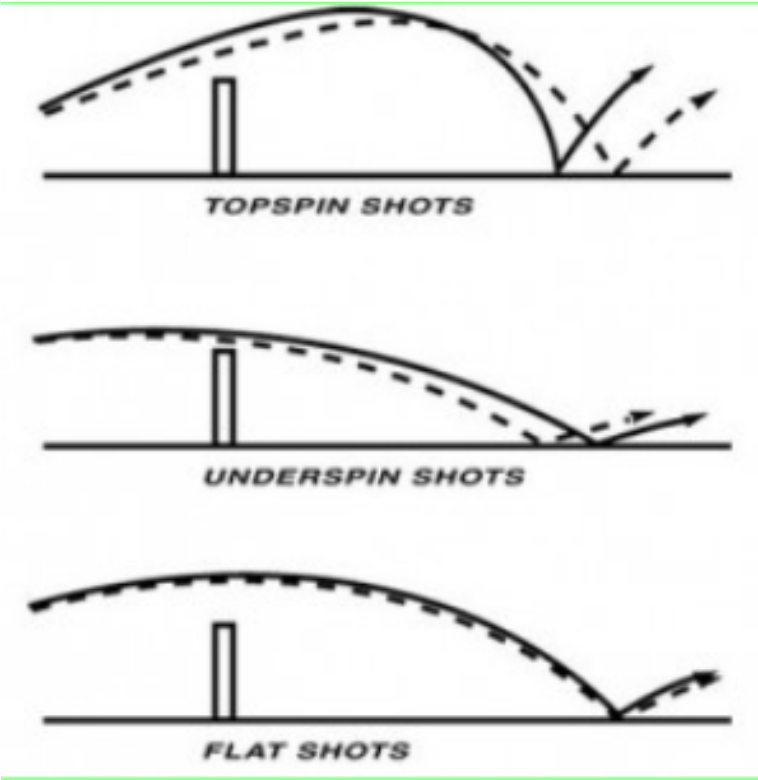

Figure 2.2
An image (Vassilev, 2014) showing trajectories of topspin, Backspin (underpin), and normal shots (flat shots) in the solid line, and the trajectories under no spin in dotted lines.

Magnus force in Tennis was first described by G. T. Walker in 1671 when observing a tennis ball Benedetti, 1989. The mathematical equations leading to the trajectory of a spinning tennis ball were first solved in Klvaňa, 1998, and it was determined that the top spin enables the earlier drop of the ball to the ground. We will follow the paper Juliana Javorova1, 2018 that analyzed the aerodynamics of a projectile in flight and the interplay between the drag and the Magnus forces. The motion is governed by a set of ordinary differential equations, in a 3D plane under the presence of drag and uniform Magnus force. Using the equations from their formulation as a reference, we

can further solve our problem.

The following assumptions are introduced in advance. The air environment has steady-state parameters and the influence of the wind actions is not taken into account. Differences due to the deformity of the ball are also ignored. The main forces acting on the ball are gravitational force, Drag force, and the Magnus force (Fig 2.3).

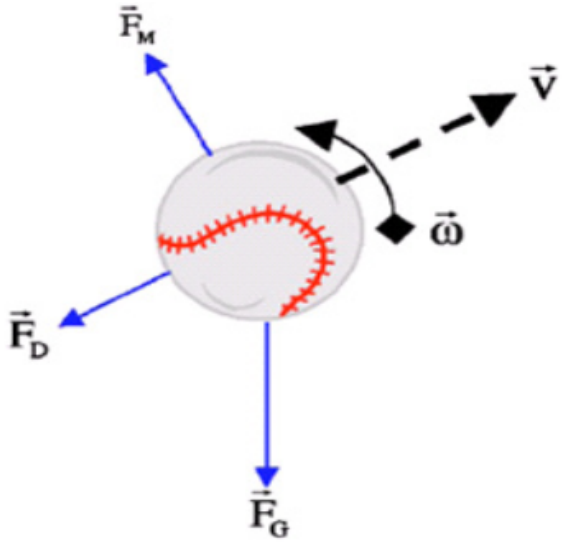

Figure 2.3
An image showing three main forces on a ball travelling with a velocity and spin (Nathan, 2008).

The gravitational force is:

$$\vec{\boldsymbol{F_G}} = -mg\hat{\mathbf{k}} \tag{3}$$

The next force that acts on the soccer ball is a drag force. Its action is in the opposite direction of the velocity of the ball (Fig 2.3). This force depends on the square of that velocity and it is determined by the formula (Juliana Javorova1, 2018):

$$F_D = -\frac{1}{2} \cdot C_D \cdot \rho \cdot A \cdot v^2 \cdot \frac{\mathbf{v}}{|\mathbf{v}|}$$

Here C_D is the drag coefficient. This above equation can be rewritten as:

$$F_D = -\frac{1}{2} \cdot C_D \cdot \rho \cdot A \cdot v\vec{\mathbf{v}} \tag{4}$$

Let $K_D = \frac{1}{2} \cdot C_D \cdot \rho \cdot A$. This gives:

$$F_D = -K_D \cdot v\vec{\mathbf{v}} \tag{5}$$

We have already defined the Magnus force in Eq. (2). Define our rotation axis $\vec{\boldsymbol{\omega}} = (\omega_x\hat{\mathbf{i}}, \omega_y\hat{\mathbf{j}}, \omega_z\hat{\mathbf{k}})$, and $K_M = \frac{1}{2} \cdot \frac{C_M \cdot \rho \cdot A \cdot v^2}{|(\vec{\boldsymbol{\omega}} \times \vec{\mathbf{v}})|}$ to state the magnitude and direction of F_M as:

$$\vec{\boldsymbol{F_M}} = k_M \cdot (\vec{\boldsymbol{\omega}} \times \vec{\mathbf{v}})$$

where,

$$(\vec{\boldsymbol{\omega}} \times \vec{\mathbf{v}}) = (\omega_x\hat{\mathbf{i}}, \omega_y\hat{\mathbf{j}}, \omega_z\hat{\mathbf{k}}) \times (v_x\hat{\mathbf{i}} + v_y\hat{\mathbf{j}} + v_z\hat{\mathbf{k}})$$

$$= (\omega_y v_z - \omega_z v_y)\hat{\mathbf{i}} - (\omega_x v_z - \omega_z v_x)\hat{\mathbf{j}} + (\omega_x v_y - \omega_y v_x)\hat{\mathbf{k}}$$

So,

$$\vec{\boldsymbol{F_M}} = k_M[(\omega_y v_z - \omega_z v_y)\hat{\mathbf{i}} - (\omega_x v_z - \omega_z v_x)\hat{\mathbf{j}} + (\omega_x v_y - \omega_y v_x)\hat{\mathbf{k}}] \tag{6}$$

By Newton's second law of motion, we have:

$$\vec{\boldsymbol{F_{net}}} = m\vec{\mathbf{a}} = m(a_x\hat{\mathbf{i}} + a_y\hat{\mathbf{j}} + a_z\hat{\mathbf{k}}) \tag{7}$$

where,

$$\vec{\boldsymbol{F_{net}}} = \vec{\boldsymbol{F_G}} + \vec{\boldsymbol{F_M}} + \vec{\boldsymbol{F_D}} \tag{8}$$

Combining Eq. (3)-(8) gives:

$$-mg\hat{\mathbf{k}}+k_M[(\omega_y v_z-\omega_z v_y)\hat{\mathbf{i}}-(\omega_x v_z-\omega_z v_x)\hat{\mathbf{j}}+(\omega_x v_y-\omega_y v_x)\hat{\mathbf{k}}]-K_D\cdot v(v_x\hat{\mathbf{i}}+v_y\hat{\mathbf{j}}+v_z\hat{\mathbf{k}}) = m\vec{\mathbf{a}}$$

which we can rearrange as,

$$(k_M(\omega_y v_z - \omega_z v_y) - K_D v v_x)\hat{\mathbf{i}} + (-K_M(\omega_x v_z - \omega_z v_x) - K_D v v_y)\hat{\mathbf{j}}$$

$$+(-mg + K_M(\omega_x v_y - \omega_y v_x) - K_D v v_z)\hat{\mathbf{k}} = m(a_x\hat{\mathbf{i}} + a_y\hat{\mathbf{j}} + a_z\hat{\mathbf{k}})$$

Writing acceleration and velocity in their differential forms and comparing the coefficients along the axis, we get three second-order differential equations, as shown below:

$$m\frac{d^2x}{dt^2} = K_M(\omega_y\frac{dz}{dt} - \omega_z\frac{dy}{dt}) - K_D v\frac{dx}{dt} \tag{9}$$

$$m\frac{d^2y}{dt^2} = -K_M(\omega_x\frac{dz}{dt} - \omega_z\frac{dx}{dt}) - K_D v\frac{dy}{dt} \tag{10}$$

$$m\frac{d^2x}{dt^2} = -mg + K_M(\omega_x\frac{dy}{dt} - \omega_y\frac{dx}{dt}) - K_D v\frac{dz}{dt} \tag{11}$$

To solve these three second-order differential equations, they will be simplified into six first-order differential equations.

Substitute:

$$x = a_1, \frac{dx}{dt} = a_2, y = a_3, \frac{dy}{dt} = a_4, z = a_5, \frac{dz}{dt} = a_6 \tag{12}$$

This gives us the desired required first-order differential equations as follows:

$$m\frac{da_2}{dt} = K_M(\omega_y a_6 - \omega_z a_4) - K_D v a_2 \tag{13}$$

$$m\frac{da_4}{dt} = -K_M(\omega_x a_6 - \omega_z a_2) - K_D v a_4 \tag{14}$$

$$m\frac{da_6}{dt} = -mg + K_M(\omega_x a_4 - \omega_y a_2) - K_D v a_6 \tag{15}$$

2.3 External data

We have jotted down the equations required to be solved in order to calculate the trajectory of the ball under the influence of the Magnus and the Drag force. The next step is to verify that the obtained results are accurate. We need theoretical as well as experimental data to achieve this. External data is of paramount importance in simulating the Magnus effect because it serves as a crucial foundation for validating and refining the simulation models. This validation process helps ensure that the simulations accurately represent the physical behaviour of the Magnus effect and the associated lift forces. As mentioned external data will be of two forms:

- *Theoretical data:* This includes data from all the sources that built a relevant model to test the Magnus effect. There are a lot of research papers involving certain aspects of the Magnus effect. To qualify as relevant data for our test, the research should have two things: A mathematical model of the Magnus

effect and the model should be built on a sport involving curve trajectories in air (like Tennis, football, golf, basketball, etc.). This excludes data from games like Billiards, hockey, etc. Notice that we can use data from sports other than tennis for our comparison by varying the initial conditions like the diameter of the ball, Magnus coefficient, etc.

Cross (2011) in his book, calculated the minimum angle at which the tennis ball should be served to just cross the net and the minimum angle at which the ball reaches the end of the court while keeping other values constant. He also calculated these angles under topspin and compared the results.

Juliana Javorova1 (2018) built a mathematical model in Matlab to compare soccer side kicks and the velocities and angles required for the ball to enter the goal.

Zhengqiu Zhu (2017) built a free-kick simulation and training system designed to assist daily training of football players. Experiments are implemented to analyze the effect of parameters such as the kicking point, the kicking force, and the kick angle on the trajectory of football.

- *Experimental data:* This includes data that has been collected from experiments done in real life. In reality, there are a few more variables that affect the path of the ball. This includes variable Drag coefficient, Variable Magnus coefficient, ball deformity, etc. Since we couldn't make our algorithm factor for these variables, it's a good idea to compare the model against real-world data for accuracy. However real-world data for tennis or other sports is scarcely available, as found out in our literature analysis.

  R. D. Mehta (2001) conducted an experiment to obtain the flow visualization on a 28-cm diameter tennis ball model to establish the boundary layer separation locations and Reynolds number effects for both non-spinning and spinning cases. Asymmetric boundary layer separation and a deflected wake flow, depicting the

Magnus effect, were observed for the spinning ball. However, it does not involve the trajectories of the ball that we want for our model.

Chapter 3

Methodology

A tennis simulation model is built in Unity. First, a 3D tennis court environment using Unity's scene editor was built. Central to the simulation was the implementation of physics-based rigid body components for the tennis ball. These components governed the movement and interactions of the objects, ensuring a realistic representation of the Magnus effect in the simulation.

Building the whole model can be distinguished into two categories:

- Creating the Algorithm.
- Creating visual simulation.

3.1 Creating the Algorithm

The purpose of the algorithm is to construct a trajectory for the ball given initial parameters. We have already seen that a tennis ball experiences three main forces i.e. Gravitational force, Drag force, and the Magnus force in Eq (8). Upon solving them, we got six ODE (Eq (13)-(15)). These Ordinary differential equations (ODE) were solved using Euler's and the Runga-Kutta method. The key difference between the two

methods is speed and accuracy. The model was built around both these algorithms and tested for performance and accuracy. No significant amount of difference was found in the results of either of these algorithms. Both algorithms took an average of 4.4 seconds to complete their first shot, with the algorithm using Euler's method executing earlier by 50 to 60ms in most cases. It should be noted that a difference of 50ms for each frame would be a lot, but this difference is for the whole trajectory of the ball. The results for the trajectory of the ball were also similar, with the algorithm using Runga-kutta being slightly more accurate. Overall, the difference in the results and performance for both these methods was not significant in our case. It might be significant when the travel distance is much larger and the game loop needs to update for a longer time. Since the Runga-kutta method provides slightly better results, we will use this method in our simulation.

Although this model was built as a tennis simulation, the algorithm provides the trajectory of a ball under the influence of the Magnus force. The goal of this research was to build a physics-based mathematical model for the Magnus effect and test it against available data to compare its performance. The physics involved in the trajectory of a ball for any sport is the same with varied conditions, so the algorithm that we built can be implemented in any sports video game. The sports video game requires a lot of processing power to ensure a smooth game experience. Some of these processes involve Gameplay mechanics, network communications, AI, etc. So we need to make sure that our algorithm is fast and accurate. Although we are using the Runga-Kutta method in this project, the code of our algorithm contains both Euler and Runga-Kutta methods. So I will provide a brief explanation of both these methods.

3.1.1 Euler's Method

Euler's method is a simple numerical technique used to approximate solutions to ordinary differential equations (ODEs). It's based on the idea of approximating the derivative of a function using a linear approximation over a small interval (Lakoba, 2012).

In the context of our system, Euler's method can be applied to approximate the change in each variable over a small time step h. The update for each variable x_i using Euler's method is:

$$x_i(t + h) = x_i(t) + h \cdot x_i'(t)$$

3.1.2 Fourth order Runga-Kutta mehod

The fourth-order Runge-Kutta method (RK4) is a numerical technique used for solving ordinary differential equations (ODEs) and is particularly popular for its accuracy and versatility. It's a family of iterative methods that approximate the solution of an ODE by considering multiple intermediate steps within each time interval.

Given the ODE: $\frac{dy}{dt} = f(t, y)$

where:

t is the independent variable (usually time)

y is the dependent variable

$f(t, y)$ is the derivative of y with respect to t

The RK4 algorithm calculates the value of y at time $t + h$ (where h is the time step) using the following equations:

1. Calculate the intermediate values $k_1, k_2, k_3,$ and k_4 as follows:

$$k_1 = h \cdot f(t, y)$$
$$k_2 = h \cdot f\left(t + \frac{h}{2}, y + \frac{k_1}{2}\right)$$
$$k_3 = h \cdot f\left(t + \frac{h}{2}, y + \frac{k_2}{2}\right)$$
$$k_4 = h \cdot f(t + h, y + k_3)$$

2. Compute the weighted average of these intermediate values to get the new value of y at $t + h$:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Here, y_n represents the value of y at the current time step t_n, and y_{n+1} represents the value of y at the next time step t_{n+1}. This process is repeated for each time step in the simulation to approximate the values of y at different points in time.

3.1.3 Applying these Method to the System

The algorithm we created provides a solution to Eq (13)-(15) to provide a position coordinate for the ball in 3D space at each update. The algorithm is explained below:

1. **Initialization:** Define the Magnus and drag coefficients K_M and K_D, and set initial conditions for a_1 to a_6.

2. **Update Loop:** Calculate time step Δt using `Time.deltaTime`. Update the new coordinates of the ball by adding the change in coordinates to the previous values.

3. **Defining differential equations:** We created a `DifferentialEquations` function, which takes the values of a_1 to a_6 and defines the ODEs as in Eq

(13)-(15).

4. **Solving the System:** We create two methods, `SolveEuler` and `SolveRungaKutta` function to update each variable using the Euler's and Runga-Kutta method respectively. The `DifferentialEquations` function is passed into these functions to solve the ODEs.

5. **Updating the Ball's Position:** We use the Runga-Kutta method to update the values of a_1, a_3, and a_5 to determine the new ball position. Euler's method is commented out in the code, which can be easily uncommented for use instead of Runga-Kutta.

3.2 Making the simulation

We made a simple tennis simulation game to showcase and test the algorithm. A tennis simulation is a virtual representation of a tennis match or game created using computer software. This simulation aims to replicate the trajectory of the tennis ball in a real tennis match, allowing users to understand the Magnus effect and experience the sport in a digital environment. Here's a short rundown of how the simulation was made:

- **Setting Up the Scene:** Create a 3D environment to represent the tennis court, including court lines, net, and surroundings. The court is made with the exact dimensions of the real tennis court, shown in fig 3.1. The net is also made to the specifications.

- **Ball Physics** Create a ball object to represent the tennis ball. Implement the above-created algorithm in this ball. The script will generate values for the position of the ball at each update, according to the initial conditions. Additionally, the ball is kept at a height of 2.87m from the ground, which is the average serving height of the ball in tennis.

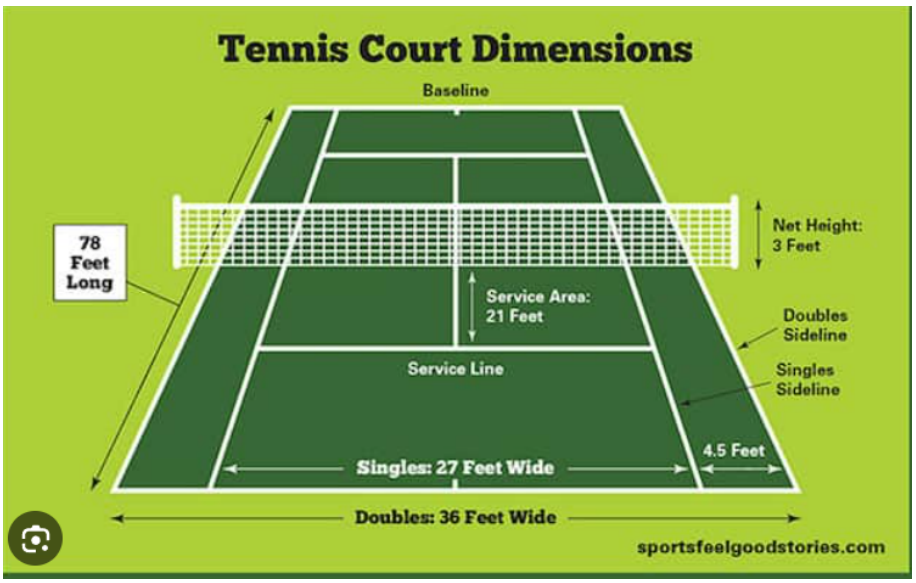

Figure 3.1
An image of the tennis court with dimensions (SportsFeelGood, 2023).

- **Camera Control** The camera is made to provide the best viewing angles. Users can change the camera to two different positions using the F1 (for the secondary camera) and F2 (for the primary camera) keys.
- **Testing parameters** The user is allowed to change the testing parameters explained in the next section. This allows the user to test the model under different conditions.
- **Axis Allignment** The XYZ axis is aligned such that, the positive X-axis points into the screen, the positive Y-axis points upwards, and the positive Z-axis point towards the left. This is because Unity follows Fleming's left-hand rule (Fig 3.2).

3.3 Testing parametres

The simulation was made to test and understand the trajectory of the ball in real-life conditions. There are three main forces acting on the ball in real-life tennis, gravitational force, drag force, and Magnus force. All of these forces depend on a number of initial value conditions. We have allowed the user to play with these values, and test the result

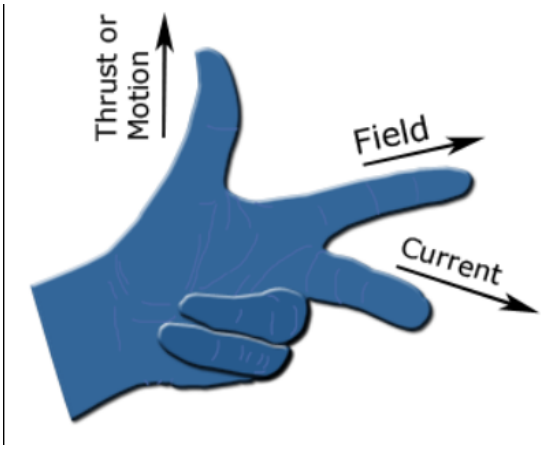

Figure 3.2
Fleming's left-hand rule (Wikipedia contributors, 2023c).

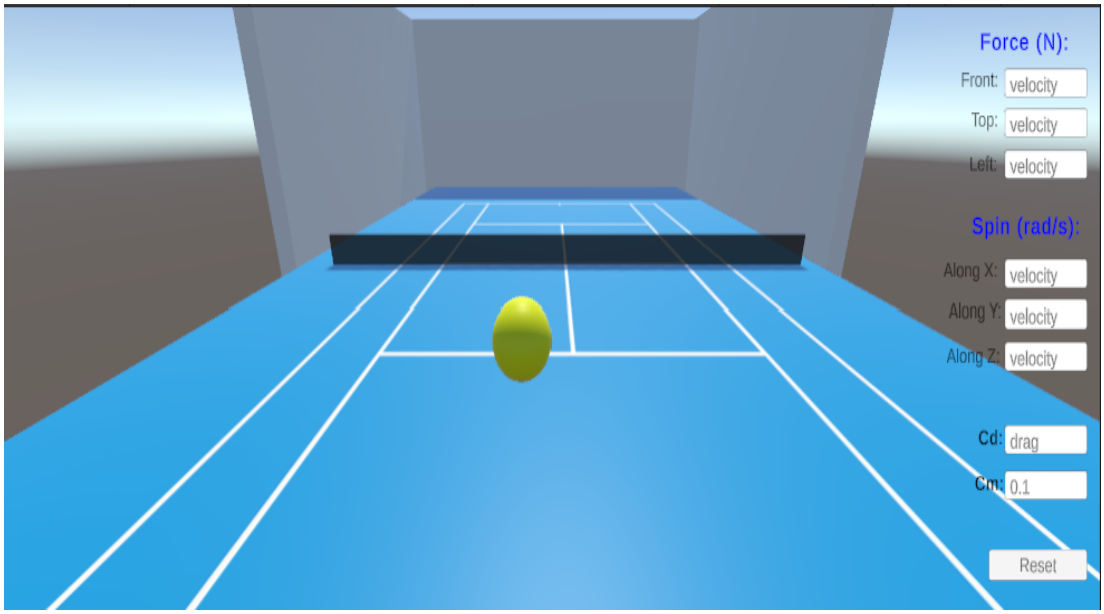

Figure 3.3
An image of the starting game screen which shows the ball set at a height of 2.87m from the ground and the testing parameters to be set.

for themselves. These values include:

- **Initial Force:** The initial force of the ball can be set to test how far the ball goes. This initial force gives us the initial velocity of the ball, which is used in further calculations. It is divided into three components, Front meaning x-component, Top meaning y-component, left meaning z-component.
- **Initial spin:** The ball is given an initial spin in any direction, due to which the ball experiences the Magnus effect. The spin is also broken down into three components. Spin algo a particular axis represents the ball will spin around that axis. So, spin along the X-axis means the ball will spin around the X-axis, and the direction of spin will be given by Fleming's left-hand rule.
- **Mass of the ball:** Mass of the ball is an important factor, as it decides the acceleration of the ball under any force. This is not available to change in the game.
- **Density of atmosphere:** The density of the atmosphere plays an important role in both drag and the Magnus force, it is set to the density of pure air ($1.29kg/m^3$). This is not allowed to change in the game.
- **Magnus coefficient:** The value of the Magnus coefficient decides the magnitude of Magnus force.
- **Drag coefficient:** The value of the Drag coefficient decides the magnitude of the Drag force.

Chapter 4

Results

4.1 verification and Performance

We want to verify that our model correctly calculates the velocity, forces, and coordinates. We will do this by comparing data against a mathematical model built on software like Matlab, Unity, etc. A mathematical simulation built into this software provides us with a graphical and real-time 3D visualization of the trajectories. This is beneficial since our goal is to create interactive simulations that involve objects experiencing the Magnus force, as you can see the effects of the force in real time.

We are going to compare the trajectories of the ball against data from two different sources, one of which is in 2D and the other in 3D. All the data mentioned for our model is taken as an average of 5 trials unless otherwise mentioned.

4.1.1 Verification against data from 2D model

Cross (2011) in his book, formulated the drag and the Magnus forces to calculate the trajectories of a ball hit from a height of $1.0m$, at a velocity of $30m/s$ and various angles to the horizontal under the influence of these forces. He found that in the absence

of spin (i.e. no Magnus force) the minimum angle at which the tennis ball should be served to just cross the net is $4°$ upwards from the horizontal. But if the ball is given a spin of 20 revolutions per second, this angle goes up to $5.5°$. Similarly, the minimum angle at which the ball reaches the other end of the court came out to be $8.1°$ from the horizontal, under no spin. But if the ball is given a spin of 20 revolutions per second, this angle goes up to $11.9°$. This is shown in Fig 4.1. It should be noted that all other values were kept constant.

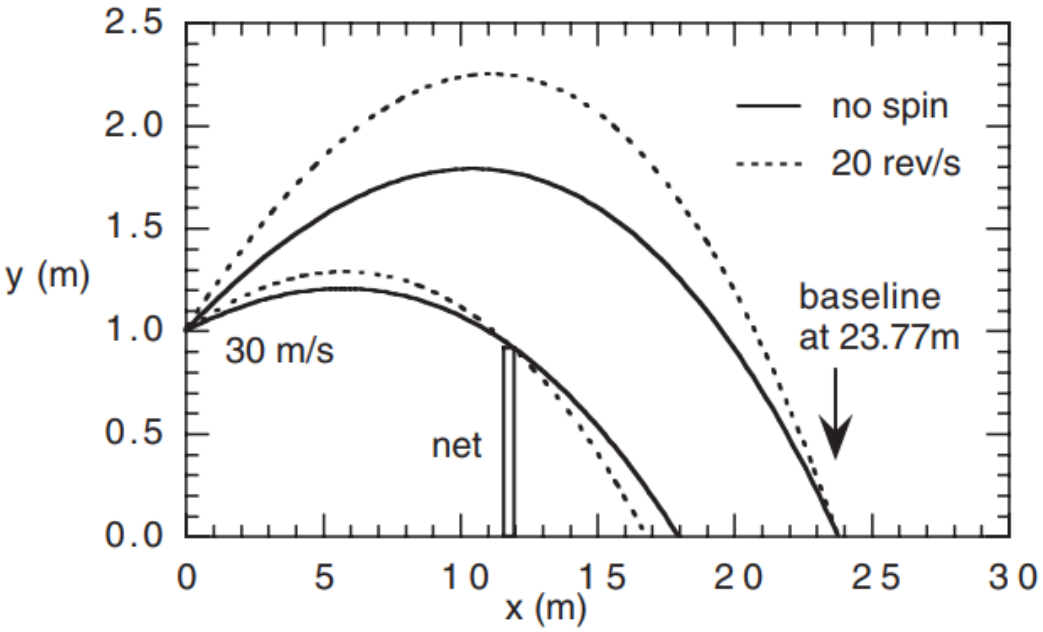

Figure 4.1
A figure showing the ball's trajectories at different angles, the solid line represents trajectories without spin and the dashed lines represent trajectories with topspin of 20 rev/s. (Cross, 2011).

To compare these results from that of our model, it was necessary to recreate all the original values of the variables. Our model was also set to hit the ball from a height of $1.0m$ with a velocity of $30m/s$. The density of the atmosphere was set as $1.21kg/m^3$. The radius of the tennis ball is around $0.033m$. The height of the tennis net at the centre is $0.9144m$. The exact values of the Magnus and drag coefficient were not explicitly mentioned, the book provides a possible range of these values. Fortunately, I was able to reverse-engineer these values from the results. The values in the model for this result are $c_m = 0.175$ and $c_d = 0.55$ for the Magnus coefficient and drag coefficient

respectively. The results we got are mentioned below and can be seen in the fig 4.2.

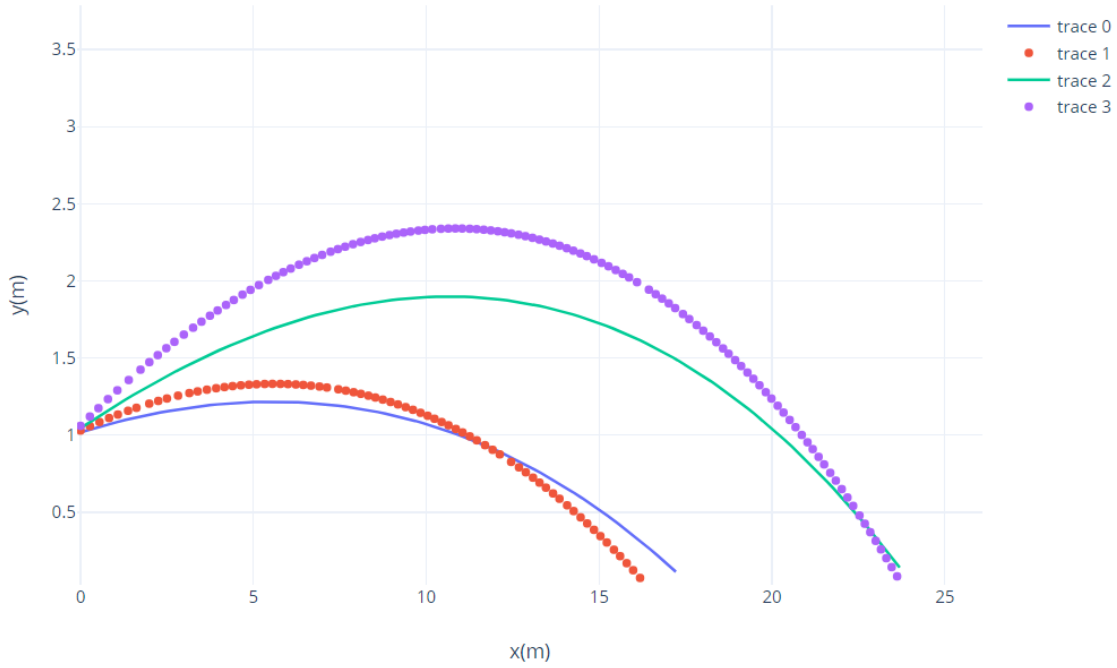

Figure 4.2
A figure showing the ball's trajectories at different angles, the solid line represents trajectories without spin and the dotted lines represent trajectories with topspin of 20 rev/s.

In the absence of spin, the minimum angle at which the tennis ball should be served just to cross the net is $4.1°$ upwards from the horizontal. But if the ball is given a spin of 20 revolutions per second, this angle goes up to $6.3°$. Similarly, the minimum angle at which the ball reaches the other end of the court came out to be $8.2°$ from the horizontal, under no spin. But if the ball is given a spin of 20 revolutions per second, this angle goes up to $12.8°$.

We can see that the angles of the ball horizontal while just clearing the net and while reaching the other end of the court in the absence of the spin are similar to the angles that the book suggests. The $0.1°$ difference can be scrapped as calculation errors. On the other hand, the difference in angles of trajectories when the ball is spinning with a topspin of $20rev/s$ is quite significant. This is because of the fact that the book does

not mention the exact values of the Magnus and drag coefficient, which led to a logical deduction of these values on my part. These values play a huge role in the trajectories of the ball and even a small change in these values can result in a noticeable change in results. For example: when the ball is hit at an angle of $11.9°$ with a topspin of $20rev/s$ and $c_m = 0.175$ in our model, the ball travels $22.86m$ in the horizontal direction. But if we change the value of c_m to 0.15, the ball will reach $23.4m$. So, using the exact values of these coefficients is important for accuracy. It was also not mentioned which algorithm was used to solve the ODE in the book. We used 4th-order runga-kutta in our model, which provides a more accurate result than Euler's method. This verifies that the results of our model are accurate.

4.1.2 Verification against data from 3D model

Although topspin is a very important aspect of the tennis game, it does not convey the full role of the Magnus effect in sports. Side-spin or curve shots are an equally important aspect of tennis as well as many other sports. To properly assess the model, we need to test the model against data against these curve trajectories, which can be done only when we compare 3D results and graphs. Due to the lack of papers which explore tennis simulation in 3D, I am using a mathematical model built for football. Juliana Javorova1 (2018) formulated the equations of forces on a football. There are three major forces acting on a football, similar to a tennis ball. Besides the obvious changes in the physical characteristics of the ball, like the mass and radius of the ball, the equations to find the ball trajectory under drag and the Magnus force remain the same. The paper built a simulation in Matlab to solve the differential equations and find the ball trajectory. They considered a direct free kick that aims to the goal. The starting point from which the footballer performs the free kick (point O in Fig. 4.3) is located at a distance of 20 m from the goal line in the direction of the central longitudinal line of the football pitch. A wall of defenders is built at a distance in the same direction

(Fig 4.3).

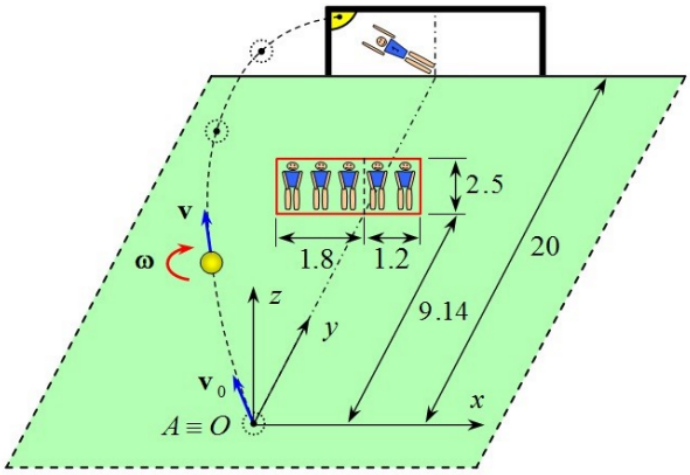

Figure 4.3
A figure showing one possible trajectory of the football to reach a goal at a distance of $20m$. The ball can either curve from left (as shown in figure) or right to reach the goal bypassing the barrier (Juliana Javorova1, 2018).

In the result, they presented three different free-kick trajectories to a target on the right side. All the initial conditions except the velocity were the same for these three kicks. The result they found is plotted in Fig 4.4. The kick that reaches the target, travels a distance of around $3.2m$ on the x-axis and $2.2m$ on the z-axis. This shows that a soccer ball can reach the target only under the proper values of the initial conditions.

To compare these results from that of our model, it was necessary to recreate all the original values of the variables. The physical parameters of the ball were changed to $m = 0.425kg$, $r = 0.11m$. The exact values of the Magnus and drag coefficient were not provided, but fortunately, I was able to reverse-engineer these values based on the results and the range given in the paper. The initial velocities of the ball were set as $v = 19.32m/s$, $v = 24.15m/s$ and $v = 28.91m/s$. The angular velocity was $61.86rad/s$. The trajectory of our model is shown in Fig 4.5 and Fig 4.6.

The results of our model show that the kick that reaches the target (marked with red in Fig 4.5) travels $3.3m$ on the x-axis and $2.1m$ on the z-axis. This is very close to the results mentioned in the paper. Overall our model achieves a 98% similar results as

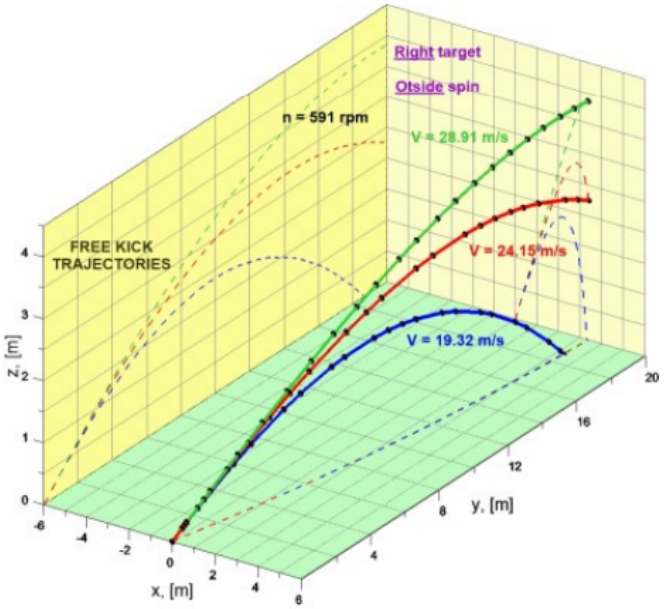

Figure 4.4
A figure showing 3 kicks with different velocities. Only the kick trajectory highlighted in red reaches the target. Both of the other kicks are a miss. (Juliana Javorova1, 2018).

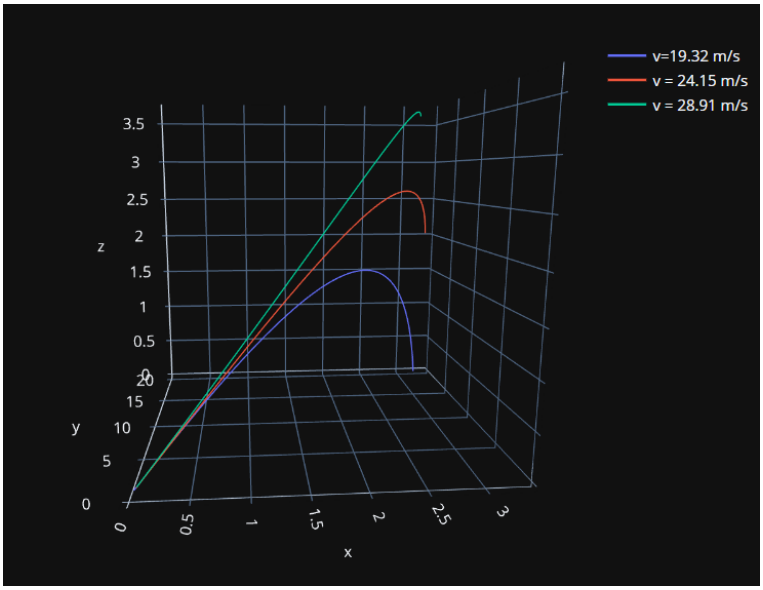

Figure 4.5
A figure showing 3 kicks with different velocities. Only the kick trajectory highlighted in red reaches the target. Both of the other kicks are a miss.

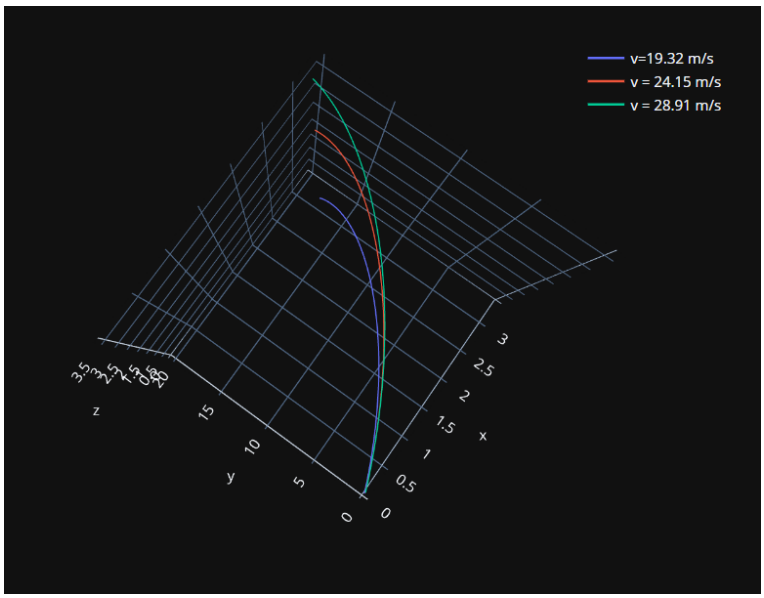

Figure 4.6
Top view of Fig 4.5.

mentioned in the paper. This means that the data points of the trajectories in our model are in $\pm 2\%$ error zone of the data mentioned in the paper.

This error can also be justified with the same argument as in the previous section. The paper does not mention the exact values of the Magnus and drag coefficient, which led to a logical deduction of these values on my part. These deductions can have some errors.

4.2 Performance

Now that our model's accuracy is verified, we will briefly look into the performance of our model. One aim of the model was to be used in Indie games and provide helpful insights to anyone looking to implement the Magnus force in their games. For a model to be considered fit to implement in any game, it has to fulfil several requirements. Some of these requirements are discussed are:

- *Compatibility:* The model should be compatible with different game engines and

support any third-party plugins. The base of our model is an algorithm which calculates the position coordinates of the ball by solving the ODEs provided to it. This code was written in c# script. I have tested the model in different Unity versions, and it is compatible with all of them. Testing of the model in different game engines like Unreal game engine, could not be completed in time since it's not straightforward. But a script written in c# can be transferred to C++ without a lot of effort.

- *CPU usage:* CPU usage refers to the amount of processing power your computer's CPU (Central Processing Unit) is using to perform calculations and run the simulation. Monitoring CPU usage is important to ensure that your simulation runs smoothly and efficiently. We used Unity's built-in Profiler window to monitor CPU usage (Fig 4.7).

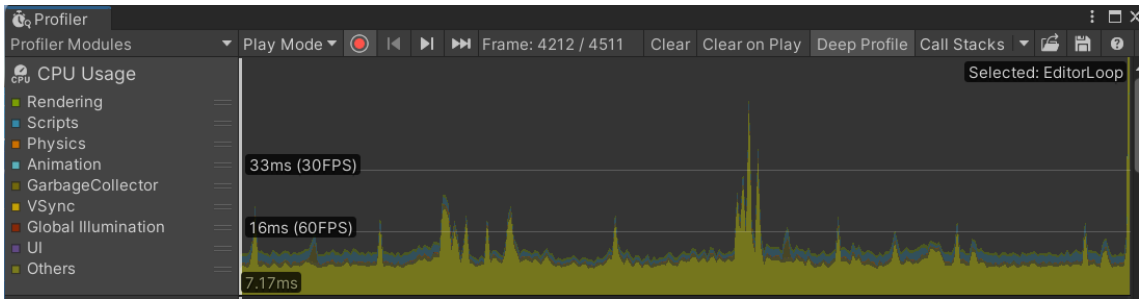

Figure 4.7
CPU usage in Unity profiler. The graph represents the usage time and FPS during the play. The graph below the line of 60 FPS represents the game is running in a frame rate better than 60 FPS.

In Fig 4.7, the left side represents the variables that require CPU usage. The graph represents the FPS and the time taken at each update. We can see most of the graph is below the line of 60 FPS, which signifies that the simulation is running at a frame rate higher than 60. There are a few spikes, where the game drops down to less than 30 FPS for some milliseconds. In Fig 4.8, we can see the average time taken by the CPU is around $9.11ms$ in a frame in the middle of the graph. The menu at the bottom shows the CPU usage of game loops. We can see that the Editor loop is using the most CPU in this case. This is also the reason

for the bigger spikes in the graph. The editor loop contains the resources used by the Unity editor and does not affect a stand-alone build of the game. What this means is that this performance glitch only happens when we play in the Unity editor. This gives us a very low CPU usage of the game, with a higher frame rate.

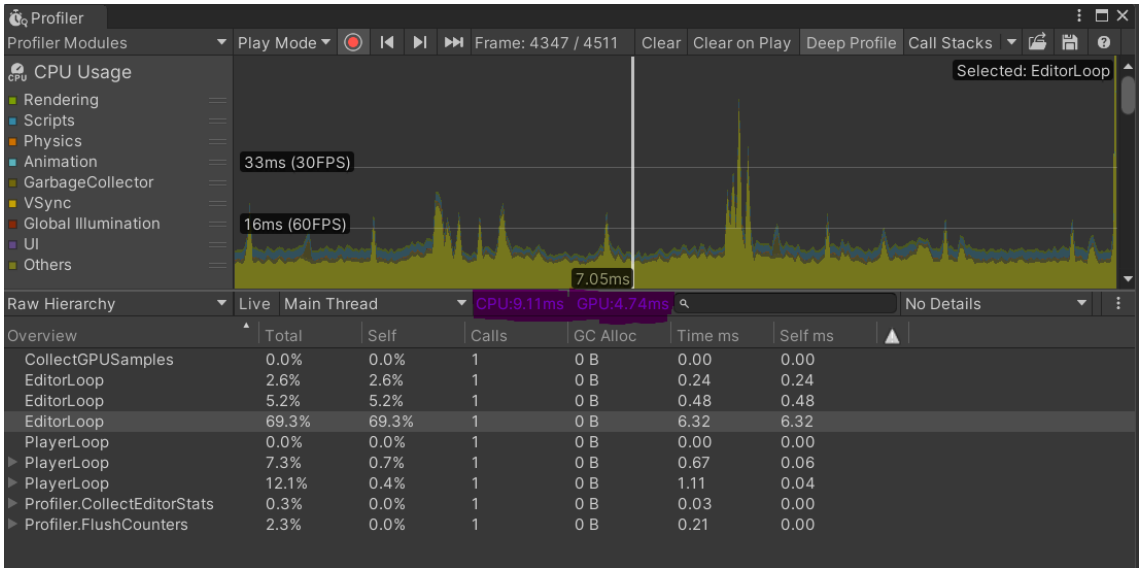

Figure 4.8
CPU usage in Unity profiler. The highlighted text in purple shows the CPU and GPU time taken at a frame in the middle of the graph. The menu at the bottom shows the CPU usage of individual game loops in that frame.

4.3 Discussion

The above two sections provide the results and insights on the capability of our model. In the verification section, we were able to show that our model provides accurate results when compared against data from papers on other mathematical simulations. This accuracy was measured by comparing the trajectories of the ball under a similar condition used in these papers. There was some inconsistency in the results of the ball trajectory under the effect of spin when compared against results from Cross (2011). The angle from horizontal required to cross the net, and the angle from horizontal required to reach the other end of the court for our model, when the ball has a topspin of $20rev/s$ is around $1°$ larger than the angles mentioned in the book. This difference was

explained by the lack of precise values of the initialization variables (the Magnus and drag coefficient). Even though this explains the difference, it still raises the question of the correctness of our model. To answer this question and to further verify the accuracy of our model, we compared our model against the data from the paper by Juliana Javorova1 (2018). They provide a more thorough solution of the equations defining the trajectory of the ball. They also consider the decrease in angular velocity of the ball due to air resistance, which we have also implemented in our model. They compared the trajectories of free kicks under three different velocities. When compared against trajectories from our model under similar conditions, we get results which are 98% accurate, i.e. the data we got lies in an approximately 2% error range of the data from the paper. This suggests that our model provides accurate results for a simulation.

Another important aspect of the simulation is the performance. We want the model to be fast and accurate. We saw that the model provides a very good frame rate (> 60) most of the time, with a few spikes to lower frame rates in between. The CPU usage is very low if we ignore the editor loop, which takes the most time during the play. Since the editor loop involves the Unity editor performance which will not be an issue in a stand-alone build, we can safely say the performance of the simulation is good enough to be implemented in any Indie game.

To properly conclude the accuracy of our model, we needed to compare the data of our model against real-world data. There are a few factors that affect the trajectory of the ball in the real world. This includes variable drag and Magnus coefficients, changing atmospheric pressure, and deformation of the ball. All these parameters were assumed to be constant for the simplicity of our model, so a comparison against data from the real world would have provided an insight into how close to real our simulation is. But obtaining experimental data on tennis or any other sport, precise enough to be compared against our model is very hard. It is one thing to measure

the distance travelled by the ball when hit with a certain velocity but to measure the drag and the Magnus coefficients and trajectory of the ball requires a proper setup. As such, I could not find any paper that measured these parameters and provided data for comparison. The closest comparison I could find was the experimental analysis done by Cross (2012). Under certain initial conditions, he found the trajectory of the ball as shown in Fig 4.9. The abnormal trajectory of the ball is due to the fact that it experiences a drastic change in the values of the Magnus coefficient. The Magnus coefficient also changes to a negative value in between which is due to the relationship between drag and the Magnus coefficients. This effect is a little complicated and out of the scope of this research. This kind of drastic change in trajectories happens under very specific conditions even in real life and is not a good reflection of the real-world comparison anyway.

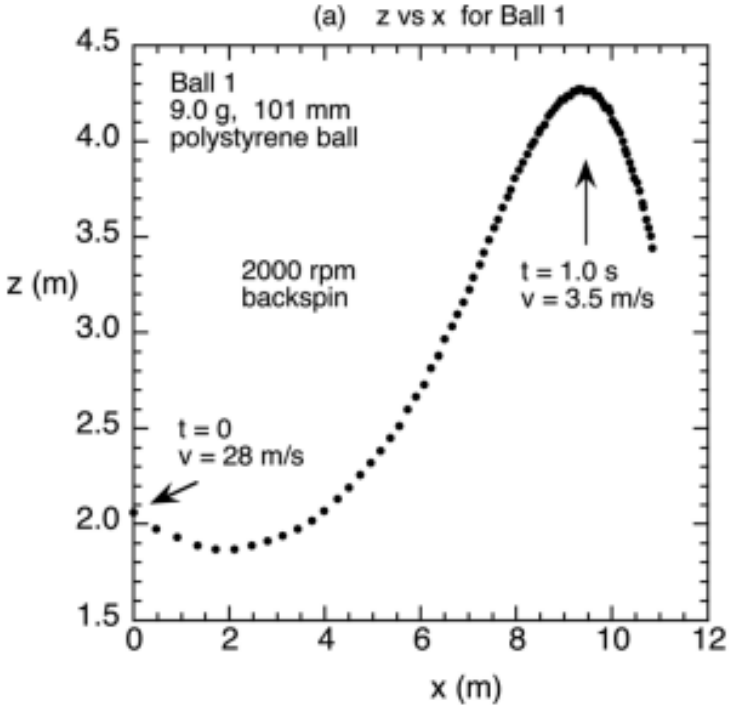

Figure 4.9
Figure showing results obtained with a 101 mm diameter polystyrene ball launched with backspin at 28 m/s (Cross, 2012).

Chapter 5

Conclusions and Future Work

5.0.1 Conclusion

The project proposed to build a model showcasing the implementation of the Magnus effect, with the goal to be useful as a resource for Indie game developers or anyone looking to implement the Magnus effect in their games. The model was built as a simulation of the Tennis game. To fulfil the goal of the project, the model needed to be easy to implement, accurate, and fast. In this report, we were able to prove that the model fulfils all these requirements. It was shown that the algorithm used in the model can easily be implemented in any other programming language. The model was then verified against data from two different papers, to conclude that the results of our model are accurate under the assumptions of constant Magnus and drag forces. The differences in results from the papers in certain conditions were explained by the lack of precise values of initial parameters in these papers. The performance of the model is compared in terms of CPU usage and frame rate. The results show that the model is pretty fast, with an average processing time of $10ms$ for each frame. This gives us a frame rate of around 100. This will be further increased when using a stand-alone build of the game since around 50% of the processing time of the game is used by the Unity editor which will not be a factor in the actual game.

We can conclude that our model can be used as a starting point for any game looking to implement the Magnus effect. The model will need to be moulded according to the game and the algorithm could be evolved according to the needs of the game. The tennis simulation helps understand and visualize the implementation. Although thorough research needs to be done before saying that the model will have any significant market value, our literature research suggests that there is a lack of Magnus effect models which can be implemented in the game. I could not find a single free-to-use model of the Magnus effect which has results anywhere close to our model. Including this with the fact that even the most popular mobile tennis game, Tennis Clash, has not implemented the Magnus effect in their game, it is safe to assume that our model can have significant value.

5.0.2 Future Work

The model was built under certain assumptions for simplicity. Removing these assumptions will make the algorithm much more complicated. However since the ultimate goal is to build a model which can simulate real-world trajectories of the ball accurately and efficiently, I would like to work on this in the future. This will require a significant amount of research since all the papers that I encountered on the Magnus effect made these assumptions. Zhengqiu Zhu (2017) built a football free kick simulation where they analyzed the effect of kicking point and shape of the ball in addition to implementing the Magnus effect. But even they did not consider the effect of weather and variable drag and the Magnus coefficient values.

If given 3 to 4 weeks more to work on the project, I would like to work on the individual mechanics of the game. Each game has some particular mechanics unique to the game. In tennis the ball bounces after crossing the net and before the opponent

receives the ball. If the ball is not spinning, it will go straight to the opponent after the drop, but a spinning ball changes its direction depending on the forces acting on the ball and spin direction. This can lead to abnormal behaviours and could make it extremely difficult for the opponent to judge which direction the ball is going to go. There are even cases when the ball bounces back to the net after bouncing off the ground (BBC SPORT, 2023). Implementing these mechanics involves more work in the application, which I was not able to do since most of the time went into literature research for data collection. So, if given more time I would definitely work on this.

Finally, as I said this model is built to help Indie developers implement the Magnus effect in their games. I consider myself an Indie developer and would love to build a whole tennis game myself, based on this model.

Bibliography

A. Kharlamov Z. Chara, P.V., 2016. Magnus and drag forces acting on golf ball. *Colloquium fluid dynamics (institute of thermomechanics as cr, prague, 2007)*, pp.1–9.

BBC SPORT, b., 2023. *Wimbledon 2023: 'never seen that before' - ball bounces on the net three times* [Online]. Available from: https://www.bbc.co.uk/sport/av/tennis/66149831.

Benedetti, G., 1989. Flight dynamics of a spinning projectile descending on a parachute. [Online]. Available from: https://www.osti.gov/biblio/6356823.

C. Guzman C. Brownell, E.K., 2016. The magnus effect and the american football. *Sports eng. 19*, pp.13–20.

Cross, R., 2011. Ball trajectories. in: physics of baseball & softball. *Springer, new york, ny. pp 368-374* [Online]. Available from: https://link.springer.com/chapter/10.1007/978-1-4419-8113-4_3.

Cross, R., 2012. Aerodynamics in the classroom and at the ball park [Online]. Available from: http://www.physics.usyd.edu.au/~cross/PUBLICATIONS/58.%20Aerodynamics%20AJP.pdf.

Dream team, 2017. Roberto carlos admits 20 years on from famous free-kick against france that he has no idea how he scored it.

G.J.E. Santos M.A. Aguirre-Lopez, e.a., 2019. On the aerodynamic forces on a baseball, with applications. *Front. appl. math. stat. 4)*, pp.1–9.

Garcia, R., 2018. The lore and science behind the spin of the soccer ball. *Ratio scientiae blog*.

Juliana Javorova1, A.I., 2018. Study of soccer ball flight trajectory. *Matec web of conferences 145, 01002 (2018)* [Online]. Available from: https://www.matec-conferences.org/articles/matecconf/abs/2018/04/matecconf_nctam2018_01002/matecconf_nctam2018_01002.html.

Khadir, L. and Mrad, H., 2015. Numerical investigation of aerodynamic performance of darrieus wind turbine based on the magnus effect. *The international journal of multiphysics*, 9(4), pp.383–396.

Klvaňa, F., 1998. Trajectory of a spinning tennis ball [Online]. Available from: https://api.semanticscholar.org/CorpusID:123145960.

Lakoba, T.I., 2012. Simple euler method and its modifications. *University of vermont*.

Lukerchenko, N., Kvurt, Y., Keita, I., Chara, Z., and Vlasak, P., 2012. Drag force, drag torque, and magnus force coefficients of rotating spherical particle moving in fluid. *Particulate science and technology* [Online], 30 (), pp.55–67. Available from: https://doi.org/10.1080/02726351.2010.544377.

Nathan, A.M., 2008. *Physics behind pitching* [Online]. Available from: http://ffden-2.phys.uaf.edu/211_spring2009.web/alex_lauritzen/pitching.html.

Nigam, D., 2019. *What are vertical axis wind turbines (vawts)?* [Online]. Available from: https://blog.arborwind.com/what-are-vertical-axis-wind-turbines.

Phelps, T., 2023. *Tennis without talent* [Online]. Available from: https://www.tenniswithouttalent.com/Ballistics.html.

R. D. Mehta, J.M.P., 2001. Sports ball aerodynamics: effects of velocity, spin and surface roughness materials and science in sports.

R. Mehta F. Alam, A.S., 2008. Review of tennis ball aerodynamics. *Sports technol*, pp.7–16.

Scholte, S., 2017. The influence of the magnus effect in tennis.

Seifert, J., 2012. A review of the magnus effect in aeronautics. *Progress in aerospace sciences* [Online], 55, pp.17–45. Available from: https://doi.org/https://doi.org/10.1016/j.paerosci.2012.07.001.

SportsFeelGood, 2023. *Tennis court dimensions, size, diagram* [Online]. Available from: https://www.sportsfeelgoodstories.com/tennis-court-dimensions-size-diagram/.

Tennis, E., 2023. *The evolution of tennis* [Online]. Available from: https://www.essentialtennis.com/the-evolution-of-tennis/.

Unity tech, 2005. *Unity game engine* [Online]. Available from: https://unity.com/.

Vassilev, E., 2014. *Virtual tennis academy lesson 1 slice & topspin in tennis* [Online]. Available from: https://www.linkedin.com/pulse/20140920022202-82311677-virtual-tennis-academy-lesson-1-slice-topspin-in-tennis/.

Wikipedia contributors, 2023a. *Bernoulli's principle — Wikipedia, the free encyclopedia* [Online]. [Online; accessed 7-September-2023]. Available from: https://en.wikipedia.org/w/index.php?title=Bernoulli%27s_principle&oldid=1170771195.

Wikipedia contributors, 2023b. *Darrieus wind turbine — Wikipedia, the free encyclopedia*. https://en.wikipedia.org/w/index.php?title=Darrieus_wind_turbine&oldid=1162243457. [Online; accessed 6-September-2023].

Wikipedia contributors, 2023c. *Fleming's left-hand rule for motors — Wikipedia, the free encyclopedia* [Online]. [Online; accessed 7-September-2023]. Available from: https://en.wikipedia.org/w/index.php?title=Fleming%27s_left-hand_rule_for_motors&oldid=1169206092.

Wikipedia contributors, 2023d. *Magnus effect — Wikipedia, the free encyclopedia* [Online]. [Online; accessed 7-September-2023]. Available from: https://en.wikipedia.org/w/index.php?title=Magnus_effect&oldid=1173763670.

Wildlife studios, e.a., 2019. *Tennis clash* [Online]. Available from: https://wildlifestudios.com/games/tennis-clash/.

Zhengqiu Zhu Bin Chen, e.a., 2017. Simulation and modeling of free kicks in football games and analysis on assisted training. *Springer nature singapore pte ltd. 2017*.