Analyzing Video Game Characters Balancing Using Eigenvalues

Kenneth Ricardo Chandra - 13523022[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523022@std.stei.itb.ac.id, [2]www.20ricardo05@gmail.com

***Abstract*—Game balancing is a critical aspect of modern video game design, especially in competitive games where character attributes, known as stats, significantly influence gameplay dynamics. Imbalances often arise, with certain characters being underpowered or overpowered, which disrupts the metagame and forces developers to release frequent balancing patches. This paper explores the use of eigenvalues and eigenvectors as mathematical tools to analyze and enhance game balance. By representing character attributes as vectors and constructing matrices from these data points, eigenvalues are calculated to quantify the influence of specific attributes on a character's performance whereas eigenvectors provide insight into the directionality of these influences. The methodology was tested on *Rainbow Six Siege*, a popular competitive game, using character stats as the dataset. Results demonstrate that eigenvalues can highlight disparities in character attributes, providing developers with a quantitative basis for adjustments even if there are a lot more factors to be considered for game balancing. This study illustrates the potential of eigenvalues and eigenvectors as tools to achieve balance in competitive video games.**

***Keywords*—Game balancing, Eigenvalues, Eigenvectors, Video games**

I. Introduction

Video games have been around for a very long time and in those times there have been a lot of different types of games. There are fighting games, racing games, shooter games, sports games, and much much more. In these games, characters or items possess specific powers and/or attributes quantified by certain values which are commonly called stats or character statistics. These stats are what defines how powerful a certain character or a certain item is. The problem with this mechanism in a video game is that sometimes there are a few characters that are underpowered or overpowered due to the stats these characters have. These mechanisms, however, are also affected by other factors such as the skills of the player, the interactions of these characters or items with another such as team up moves in the game Marvel Rivals, and even the environment where the characters are used. All of these factors make it hard for each and every character to be perfectly balanced, especially in a competitive game like Valorant or League of Legends. Due to various characters being underpowered or overpowered, it eventually will affect how the metagame environment works in a video game. The metagame or usually referred to as meta in a video game is a condition where there are some characters which are better than the others which makes those characters to be highly valuable and highly used whereas other characters which don't belong in this meta class will meet very little usage and might even be abandoned by a lot of players, leading the developers of the video game to make a few adjustments in the forms of patches or updates in the game. These update patches usually come in a set interval of a few weeks or months depending on the game to let the game scene unfold by itself. If there are actually some characters that are underperforming, or too overpowering, the developers will intervene in making a balancing patch to make the game become more balanced. But due to those outside factors like the environment and skills of the player, these balancing patches sometimes don't make the game become balanced but instead make it worse. For example, in the previous version of Valorant prior to Patch 8.11, the character Neon has an ability where she could run for a certain amount of time and she could end the run with a slide, however during the time after she is sliding, she will enter an animation to arm her weapon and then she would be able to shoot. Now the changes made in Patch 8.11 is that Neon would be able to arm her weapon faster and that when she shoots for the first time after sliding, there would be no shooting inaccuracy so if the crosshair of the player is directly at an enemy's weak or critical point, the player using Neon has the opportunity to just hit and run without any downside, making it a very strong character and overpowered character to use depending on the aim of the player. This was one of the examples of game balancing going from an underwhelming character to a very powerful and overpowered character. Now in normal gameplay, this wouldn't be much of a problem because not everyone has the best aim but this change affects a lot of things in the professional gaming scene. In the professional Valorant scene where the players have really good aim, this becomes a big problem as Neon becomes one of the most picked characters to be used when there is a competition.

This results in the change of the metagame and makes other characters in the same class with Neon to be under utilised. In a way, the game balancing worked but in return it makes a certain character to be overpowered and overused while it makes some other to be underused which leads to another update launched by the developers to re-balance the game once more. As balancing a game character is a hard thing to do, eigenvalues can be a way to balance out game characters by analyzing the interactions and relationships of certain characters attributes or stats. Each of the attributes can be represented as a vector which in turn can be made into a matrix. Based on the data available, data on the characters can be implemented onto the matrix where the eigenvalues and eigenvectors of these matrices will be calculated. These eigenvalues represent how much the related data influence the characters and the eigenvectors will represent the directions of these influences to the attributes of the characters. Using the eigenvalues from these matrices will make it easier to balance out characters as it will be easy to see how much an attribute or stat influences the character.

II. Theoretical Basis

A. Video Games

Video Games are games made and run with an interactive design that can be played using an electronic device that has a screen. Video games can be traced back to the 1950s where in 1958, the first ever video game "Tennis for Two" was made by William Higinbotham. Modern video games have evolved a lot by the years, making it accessible to be played on almost anywhere on any platform. While there used to be console and computer games, now there are mobile games that can be played from your phone and there are even special handheld devices made just for playing games. Due to the evolution of hardware and technology, games itself have been evolving as well. While there used to only be games with no sound and only simple 2D displays like Tetris, the gaming world are now able to make some very realistic 3D worlds as if they are the real world itself, with very high quality models and sound effects, making gaming an even more immersive experience. Video games also have a wide variety of genres such as fighting games, racing games, shooter games and much much more that there could be infinite possibilities on what a video game is like. Hundreds of thousands of video games have been made and it has become something that can make someone's life as video games can be a source of income, whether making a successful career as a video game developer or going professional in a video game competitive scene like Valorant or Counter Strike.

B. Character Attributes

Character attributes refers to the properties of what a game character has such as strength, health, speed, and much much more. These properties will affect how a character behaves or moves, depending on the game and the situation. There are no limitations on the attributes that can be made on a character. For example, character attributes could just be as simple as having only health and damage output while some could be very intricate and detailed with height, weight, or even amount of weight a character can carry. These attributes are what defines a game character and it dictates how much a game character does in that game.

C. Game Balancing

Game balancing is a process of balancing a mechanic in a game so that the gameplay is fair for all players. Game balancing is what game developers need to do to make a good game. If a game is unbalanced, it could make the game unplayable and it could lead to players having a rough time and unable to enjoy themselves. This is why game balancing is very important, especially in a multiplayer game and a game where competitiveness is one of the important aspects of the game. With proper game balancing, each player would be able to play the game fairly and it would bring more stability to the game, making it more fun to be played equally. While some might still struggle in a balanced game, in the end the struggle depends on the player's own skill and how the player acts using the game character. All game balancing is needed to make things fair for all players, including for single player games where there might not be a competitive feeling, but game balancing is still needed so that the game is working seamlessly from the start until the end without feeling like something is amiss.

D. Eigenvalues and Eigenvectors

The word Eigen comes from the German word that has the same spelling that means characteristics or own. Eigenvalue is a value that represents a characteristic from a matrix. Eigenvalues are usually used to measure a magnitude of variations in data which in turn will help to identify the key components of the data. Eigenvectors on the other hand are vectors made up from the eigenvalues that represent the directions of the variations of the data. The eigenvectors highlight relationships between different variables on the data.

If a set of data is made into a matrix A with the size of n*n, then the equation of $Ax = \lambda x$ is true with λ being the eigenvalue of A and x is the corresponding eigenvector to the eigenvalue. An eigenvalue from a matrix A can be calculated by doing this.

$$Ax = \lambda x$$
$$IAx = I\lambda x$$
$$Ax = I\lambda x$$
$$(\lambda I - A)x = 0$$

and then calculate the determinant of $(\lambda I - A)x = 0$. By doing that, it will be possible to get the eigenvalue which is the roots from the characteristic equation made from using the determinant of $(\lambda I - A)x = 0$. The eigenvalues can be used to make the eigenvectors by substituting it to the matrix $(\lambda I - A)x = 0$. Each eigenvalue has their own eigenvectors and those eigenvectors in turn will make eigenspaces which are the bases for eigenspaces with a certain eigenvalue.

Eigenvalues are used in this analysis as eigenvalues can show properties between data which will in turn tell how much bigger or smaller of an impact the properties have on the data which is why it can be used to analyze game balancing as the eigenvalues can show which attributes have higher values than the other, making the character imbalances, if any, to be discovered easier.

III. Data

For the sake of this analysis, the video game characters used will be from the video game Rainbow Six Siege (R6). Rainbow Six Siege is a shooter game that revolves around two teams fighting to plant a bomb defuser (the attacker) or to stop the other team from defusing the bomb (the defender). The ranked mode of this game allows players from both sides to ban one character each from the attacking side and the defending side. This analysis will be using the character's pick and win rate, difficulty usage, speed, and health. The pick and win rate is a statistic that shows how much a character is picked and then proceeds to win the game with the same character. The difficulty usage is a statistic shown in game to help players determine which characters are easier to use. The speed and health attributes are two statistics that show how fast the characters can go and how much health the characters have. This is due there being an official release of some of these stats by the R6 developers and by using these metrics, an analysis of the game balance can already be seen. Since the game is still being updated, and that the current season is Year 9 Season 4, the data used will be from the mentioned season.

The following are the attributes of attackers that is playable in Rainbow Six Siege taken from the game itself

Operators	Difficulty	Speed	Health	Side
Striker	1	2	2	Attacker
Sledge	1	1	3	Attacker
Thatcher	1	1	3	Attacker
Ash	2	3	1	Attacker
Thermite	1	2	2	Attacker
Twitch	2	2	2	Attacker
Montagne	3	1	3	Attacker
Glaz	2	3	1	Attacker
Fuze	1	1	3	Attacker
Blitz	3	2	2	Attacker
IQ	3	3	1	Attacker
Buck	1	2	2	Attacker
Blackbeard	2	1	3	Attacker
Capitao	2	3	1	Attacker
Hibana	1	3	1	Attacker
Jackal	3	2	2	Attacker
Ying	2	2	2	Attacker
Zofia	1	1	3	Attacker
Dokkaebi	2	3	1	Attacker
Lion	1	2	2	Attacker
Finka	1	2	2	Attacker
Maverick	2	3	1	Attacker
Nomad	3	2	2	Attacker
Gridlock	1	1	3	Attacker
Nokk	3	2	2	Attacker
Amaru	2	2	2	Attacker
Kali	2	2	2	Attacker
Iana	1	2	2	Attacker
Ace	1	2	2	Attacker
Zero	1	3	1	Attacker
Flores	2	2	2	Attacker
Osa	2	1	3	Attacker
Sens	2	3	1	Attacker
Grim	1	3	1	Attacker
Brava	3	3	1	Attacker
Ram	2	1	3	Attacker
Deimos	2	2	2	Attacker

Table 1. Attributes of Attacker in Rainbow Six Siege as of Year 9 Season 4

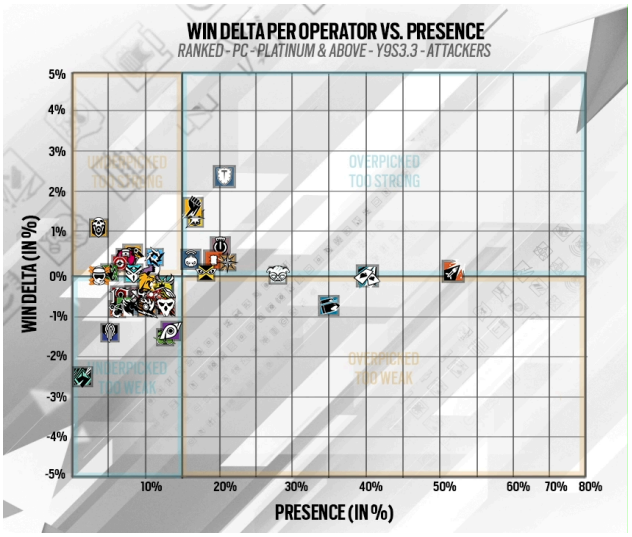
The following are the attribute of defenders that is playable in Rainbow Six Siege taken from the game itself

Operators	Difficulty	Speed	Health	Side
Sentry	1	2	2	Defender
Smoke	2	2	2	Defender
Mute	1	1	3	Defender
Castle	2	2	2	Defender
Pulse	3	3	1	Defender
Doc	1	1	3	Defender
Rook	1	1	3	Defender
Kapkan	1	2	2	Defender
Tachanka	1	1	3	Defender
Jager	2	2	2	Defender
Bandit	1	3	1	Defender
Frost	1	2	2	Defender

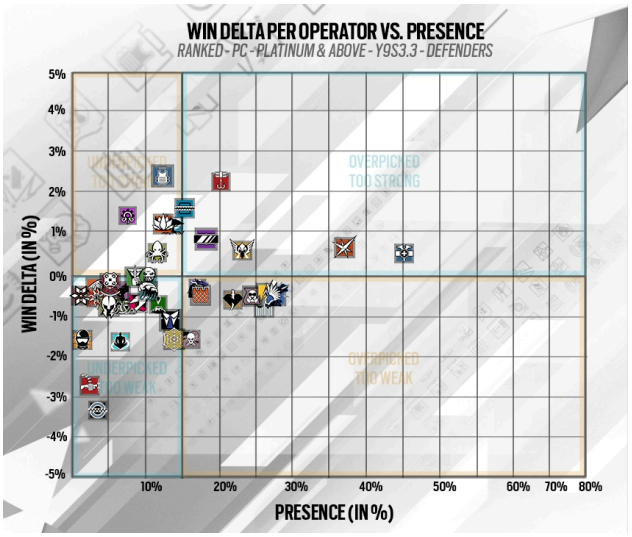
Valkyrie	2	2	2	Defender
Caveira	3	3	1	Defender
Echo	3	1	3	Defender
Mira	3	1	3	Defender
Lesion	1	2	2	Defender
Ela	1	2	2	Defender
Vigil	3	3	1	Defender
Maestro	2	1	3	Defender
Alibi	3	3	1	Defender
Clash	3	1	3	Defender
Kaid	2	1	3	Defender
Mozzie	2	2	2	Defender
Warden	2	1	3	Defender
Goyo	2	2	2	Defender
Wamai	2	2	2	Defender
Oryx	2	2	2	Defender
Melusi	1	1	3	Defender
Aruni	2	1	3	Defender
Thunderbird	1	2	2	Defender
Thorn	1	2	2	Defender
Azami	3	2	2	Defender
Solis	3	2	2	Defender
Fenrir	3	2	2	Defender
Tubarao	2	2	2	Defender
Skopos	2	2	2	Defender

Table 2. Attributes of Defenders in Rainbow Six Siege as of Year 9 Season 4

The following are the official data of the player base on the winning and picking rates of the operators.



Picture 1. Win Delta Per Operator vs Presence, Attacker.



Picture 2. Win Delta Per Operator vs Presence, Defender

In both charts, operators are positioned based on two factors: their pick rate and win delta. Operators placed further to the right have higher pick rates, while those positioned higher on the chart have greater win deltas. The win delta represents the percentage increase in round wins during ranked matches when the operator is selected.

These attributes will be made into 5 categories, underpicked and weak, overpicked and weak, balanced, underpicked and strong, overpicked and strong with the number 1-5 corresponding to the categories for the sake of analysis. The following is the result of the official data made after it is divided into the 5 categories.

Attackers	Pick/Win Rate	Defenders	Pick/Win Rate
Striker	3	Sentry	1
Sledge	1	Smoke	2
Thatcher	3	Mute	2

Ash	3	Castle	3
Thermite	3	Pulse	3
Twitch	3	Doc	5
Montagne	5	Rook	4
Glaz	4	Kapkan	5
Fuze	3	Tachanka	1
Blitz	5	Jager	1
IQ	3	Bandit	2
Buck	2	Frost	4
Blackbeard	4	Valkyrie	5
Capitao	1	Caveira	3
Hibana	3	Echo	3
Jackal	1	Mira	3
Ying	1	Lesion	5
Zofia	3	Ela	3
Dokkaebi	3	Vigil	2
Lion	3	Maestro	1
Finka	5	Alibi	3
Maverick	1	Clash	1
Nomad	5	Kaid	2
Gridlock	4	Mozzie	1
Nokk	1	Warden	1
Amaru	3	Goyo	3
Kali	3	Wamai	1
Iana	3	Oryx	1
Ace	3	Melusi	3
Zero	1	Aruni	3
Flores	1	Thunderbird	1
Osa	4	Thorn	1
Sens	1	Azami	2
Grim	3	Solis	1
Brava	4	Fenrir	3
Ram	3	Tubarao	3
Deimos	3	Skopos	4

Table 3. Categories of Pick/Win Rates in Rainbow Six Siege as of Year 9 Season 4

IV. Implementation

From the data that is collected, all of the operator's stats will be calculated using a python program that will give the eigenvalues of each attribute. This is the program that is used to calculate eigenvalues.

```
import numpy as np

# Function to calculate eigenvalues and eigenvectors
for non-square data
def calculate_eigenvectors_non_square():
    try:
        # Input the data matrix as a multi-line string
        print("Paste your data matrix (rows separated by
new lines, values separated by spaces):")
        matrix_input = []
        while True:
            line = input()
            if line.strip() == "":  # Stop when an empty line
is entered
                break
            matrix_input.append(list(map(float,
line.split())))

        # Convert the list to a NumPy array
        matrix = np.array(matrix_input)

        # Calculate the covariance matrix
        covariance_matrix = np.cov(matrix,
rowvar=False)

        # Calculate eigenvalues and eigenvectors of the
covariance matrix
        eigenvalues, eigenvectors =
np.linalg.eig(covariance_matrix)

        # Display the results
        print("\nCovariance Matrix:")
        print(covariance_matrix)
        print("\nEigenvalues:")
        print(eigenvalues)
        print("\nEigenvectors (columns correspond to the
eigenvalues):")
        print(eigenvectors)

    except Exception as e:
        print("Error:", e)

# Run the function
calculate_eigenvectors_non_square()
```

Table 4. Python Code For Calculating Eigenvalue and making Eigenvectors

By using this and making the matrix having 4 columns of attributes being difficulty, speed, health, and pick/win rates, the following eigenvalues are attained for the attacker operators

[4.97553909e-01 9.92833043e-01 1.76186530e+00 4.61512844e-17]

and this is for the defender operators

[1.77856766e+00 9.62000300e-01 5.19191796e-01

1.96436547e-16]

Using these eigenvalues, the eigenvectors for the attackers are

[9.32041372e-01 -3.61490379e-01 2.49717200e-02 -4.37712227e-17]
[-2.19971066e-01 -5.89452495e-01 -3.22735940e-01 -7.07106781e-01]
[2.19971066e-01 5.89452495e-01 3.22735940e-01 -7.07106781e-01]
[-1.85807269e-01 -4.17631679e-01 8.89415223e-01 4.52596033e-19]

and this is for the defenders
[[1.27622864e-01 4.56871545e-01 8.80329936e-01 1.13988645e-16]
[-2.53181937e-02 6.28711028e-01 -3.22616540e-01 7.07106781e-01]
[2.53181937e-02 -6.28711028e-01 3.22616540e-01 7.07106781e-01]
[-9.91176262e-01 2.67072578e-02 1.29831967e-01 -2.87478458e-17]]

each of these eigenvectors correspond with each of the attributes that is in the matrix. Taking one of the eigenvectors from the attacker into account, [2.19971066e-01 5.89452495e-01 3.22735940e-01 -7.07106781e-01], this one in particular, it can be seen that the biggest value in the eigenvector is the on the speed column, this means that there are a lot of varieties in the speed attributes of the attackers in Rainbow Six Siege which may lead into an imbalance in the game. However, these different speed attributes are also a result of balancing due to the weapons and skill sets that the operators have, which is not included here. But if the only attributes that are taken into account is this, it can be seen that the speed attribute is not really balanced. Aside from that, looking at the difficulty and health attributes in the eigenvector, there are some varieties but it isn't too much so much so that it needs lots of rebalancing, and that includes the pick/win rate which in reality does not contribute immediately to the balance to game as the other attributes does but looking at the value of the eigenvector, it can be seen that the pick/win rate does not influence how the game balance work.

However, this is very much different on the defender's eigenvector, especially this one [-2.53181937e-02 6.28711028e-01 -3.22616540e-01 7.07106781e-01]. From this eigenvector, it can be seen that the highest value is from the pick/win rate attribute, which is the exact opposite of the attacker's data. This can be said that there are a lot of varieties on the pick/win rate on the defenders, showing that there are not many balanced operators from the defender's side that is on one side of the spectrum only but instead is spread out pretty big. It can be said that this is a good thing, however it also can be said that it is very unbalanced since this implies that there are some operators who are really weak and some who are really strong. Aside from the pick/win rate having the highest value, the speed attribute also has a high value which as mentioned before, has a high variety but this too is caused by a lot of other factors that are not used in this analysis. What all of this shows actually is that we could see how eigenvalues and eigenvectors actually work in helping to analyze game balancing. So this means if a game needs a balancing in attributes, the developers can use eigenvalues and eigenvectors to see if their game is really balanced or if there is an attribute that has too much imbalance.

V. Conclusion

To conclude this analysis, it can be seen that there are still a lot of factors that can be inserted into the dataset. However it will take a much longer time to analyze as not all of the data can be quantified. This however shows that analyzing game balancing can be done by using Eigenvalues and Eigenvectors. Although there are lots of other ways to see if a game is truly balanced or not, this is one of the ways a game developer can check the balance of their game.

VI. Acknowledgment

Before this paper is finished, I would like to express my deepest gratitude to those who have supported me throughout this journey. First, I would like to thank God for guiding me and being my constant source of strength and inspiration since the beginning of my life. Next, to my family whom I am forever grateful for your unwavering support and belief in me, which has been the foundation of all my achievements. I also extend my thanks to my friends in IF'23 ITB, who have been my trusted companions and to my teachers and professors, I deeply appreciate your dedication and exceptional guidance, which has been invaluable in helping me complete this paper. Finally, to you, the reader, thank you for taking the time to engage with this paper. I hope this paper helps.

References

[1] Duffy, K. (2022, September 15). What was the first video game ever made? Exploring the origins of gaming history. USA Today. Retrieved January 2, 2025, from https://www.usatoday.com/story/tech/2022/09/15/what-was-the-first-oldest-video-game/8021599001/

[2] Ubisoft. (n.d.). Y9S4 designer's notes. Rainbow Six Siege. Retrieved January 2, 2025, from https://www.ubisoft.com/en-gb/game/rainbow-six/siege/news-updates/qvAE3p87KtfLUhvTVTjUg/y9s4-designers-notes

[3] Munir, R. (2023). Nilai eigen dan vektor eigen (bagian 1). Informatika STEI ITB. Retrieved January 2, 2025, from https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-19-Nilai-Eigen-dan-Vektor-Eigen-Bagian1-2023.pdf

[4] Ubisoft Montreal. (2015). Tom Clancy's Rainbow Six Siege [Video game]. Ubisoft. Retrieved January 2, 2025

STATEMENT

With this, I declare that my paper that I wrote is my own writing and its not a translation or a copy of another paper and it is not a plagiarism of any sort.

Bandung, 27 Desember 2024

Kenneth Ricardo Chandra 13523022