Optimize Gaming Effectiveness using Decision Tree on *Plains of Eidolon* from *Warframe*

Nadia Mareta Putri Leiden - 13520007[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*13520007@std.stei.itb.ac.id*

***Abstract*—Computer's decision-making skill in a game (*Warframe*, in this case) is a crucial part of any gameplay in order to analyze the user's inputs and determine the next state that may be possible while fighting the in-game bosses. Moreover, to rule out every possibility that the user might choose to do during the session will require a specific algorithm, an efficient and powerful one, to ensure the best gaming experience, as the failure of choosing the right algorithm will result in the slowness of game processing time. It is also to make the game as realistic as possible and easier to maintain. Therefore, one of many algorithms that may help programmers optimize their code is the concept of Tree or usually known as Decision Tree.**

***Keywords*—Tree, Warframe, Decision, Eidolon.**

I. Introduction

Warframe is science fiction themed third-person shooting game which was initially released in 25th of March 2013 and can be accessed through steam accounts. The developers provided multiple features which enables users to experience various types of gaming experiences. However, the main idea of Warframe itself revolves around TPS-oriented gaming with a choice of close or ranged combat. Moreover, each weapon can be installed with *Mods*, an instance that can increase the effectiveness of weapons' properties—Mods can also enable weapons to get additional elemental properties that will help the users to fight specific types of enemies which will be elaborated further on Chapter III.

Referencing to what the author has stated on the first paragraph, Warframe supports multiple gaming experience, the player usually progressed in game through finishing available sequential quests and unlocking the maps on each planet (or *star charts*) with a unique mission type on each map. The types are spy, rescue, capture, excavation, sabotage, exterminate, defense, mobile-defense, survival, open-world, and assassination (Boss fight) missions. However, the author would limit the scope of the paper on the open-world missions, although there were three maps that supports open-world experience in Warframe, the paper will only focus on one specific open-world map which is the *Plains of Eidolon*.

Through accessing Plains of Eidolon user will be able to farm for stronger items and last but not least, fighting a challenging boss, Eidolon Teralyst, Gantulyst and Hydrolyst, a 3-phase boss which can be accessed if the player has fulfilled the requirements while they were fighting the Eidolon. If they did not fulfill the requirements, the Eidolon will be killed instead of captured. This will cause the player to not be able to proceed to the next phase. Therefore, to make it easier for readers to understand the phases, the author would like to illustrate the phases as below:

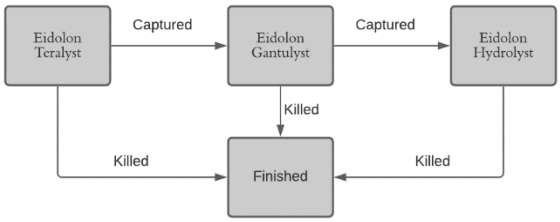

Figure I Author's illustration on the Eidolon Boss Phase

As the readers can see, the player has to fight Eidolon Teralyst before they can proceed to Gantulyst, and so forth. This will be the basis of the paper main discussion on the next Chapters. Furthermore, whether the Eidolon is killed or captured, each choice will yield different rewards respectively.

II. Theoretical Foundations of Tress

Trees are a part of graph. However, it is only one part of a Graph, due to the fact that the definition of Trees itself is an undirected connected graph which contains no cycles. Therefore, in order to illustrate how trees work, the author would like to attach the illustration below:

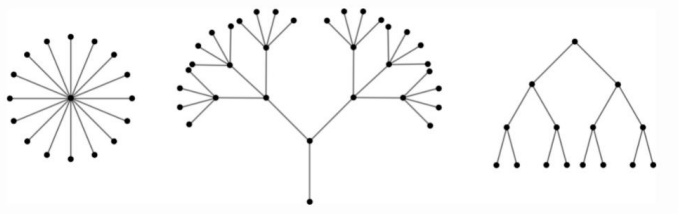

Figure II Graphs that can be considered as Trees
Source: Paul Johnson Lecture 6 on https://ptwiddle.github.io/MAS341-Graph-Theory-2017/lecturenotes/lecture6.html

Trees seemingly to resemble regular graph, it is because Trees indeed a part of Graph. However, in order to give a better demonstration, the author would also like to attach another illustration so that the reader could see the difference, which the

author will also elaborate further below:

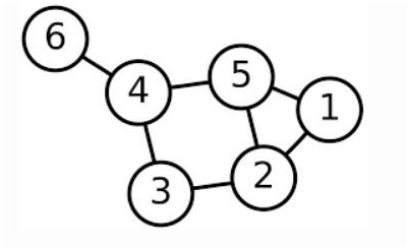

Figure III Graphs that cannot be considered as Trees
Source: Paul Johnson Lecture 6 on https://ptwiddle.github.io/MAS341-Graph-Theory-2017/lecturenotes/lecture6.html

As the readers can see in Figure III the graph itself can form a cycle, with a starting point of 1. For instance, a cycle of 1-2-3-4-5-1 can be formed without passing the same edges more than once.

However, in Figure II, there were not any cycle that can be formed without passing the same edges more than once. Therefore, the definition and properties of Trees can be further elaborated as below.

If T = (V, E) is an undirected simple graph with a vertex of *n*, V is a number of vertices, and E is number of edges, then the statement below determines that T is a tree (the statements are equivalent):

1. T is a tree
2. Between two vertices in T = (V, E) there will always be a unique path.
3. Any removal of edges will cause T to become disconnected
4. T does not contain any cycles, but adding any edges will cause the T to have a cycle. For example, adding one edge into T will create a cycle, but will only create one cycle.
5. T is connected, has no cycles, and contains n-1 edges.

Some programmers in their implementation of some algorithm would make multiple trees. Hence, exist the definition of Forest. Forest is a set of disconnected Trees.

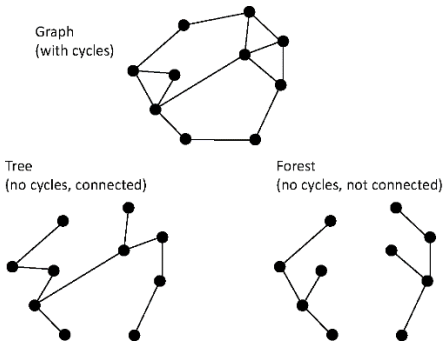

Figure IV Trees, Forest, and Regular Graph Comparison
Source: Cambridge University Press
https://www.cambridge.org/core/books/abs/applying-graph-theory-in-ecological-research/shapes-of-graphs-trees-to-triangles/6F7A9487D10EF5ED28A83F7B035E7FDC

Trees is a powerful algorithm that enables programmers to create an efficient application, such as Binary Search and even enables an automated decision-making skill—there is a specific implementation for this use, which is widely-known as *Decision Tree*. The use of Decision Tree is frequently implemented on various branch of Computer Science, for example Machine Learning is known for utilizing Decision Tree to make a strategic approach or precisely calculated decision, mostly for economic reasons.

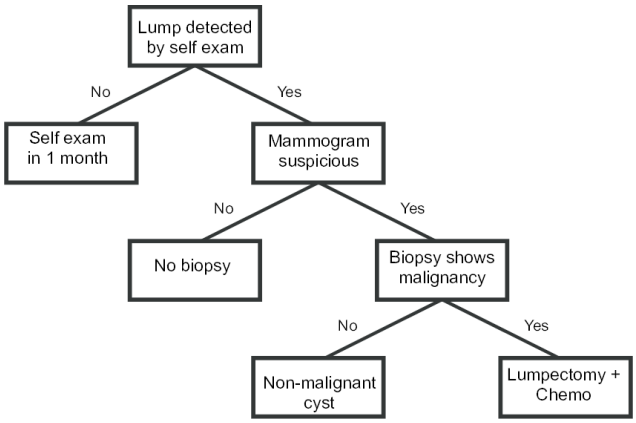

Figure V Decision Tree used in Assessing Breast Cancer (Machine Learning)
Source: Research gate paper on https://www.researchgate.net/figure/An-example-of-a-simple-decision-tree-that-might-be-used-in-breast-cancer-diagnosis-and_fig2_24442935

However, Decision Tree is not exclusively used on Machine Learning, its practices and concepts are applicable to the other branches as well. In this case, the use of Decision Tree can also generate an automated response from a human's input which will be highly valuable in game development. Hence, creating realistic and possible scenarios in game experiences will surely enhanced from Decision Trees, as the use of Trees will not only be able to speed up the game performance, but is easier to maintain if the programmer had already grasped the general idea of Linked Lists.

It is important to note that the use of Decision Tree is not restrictive, it is only one of many other powerful algorithms to implement in computer's decision-making skill. The general idea is, using Decision Tree is one of the easier ways to implement automated decision-making skill. Game development will need an implementation of Decision Tree in a lot of their implementation like the author has stated before, for instance, the implementation of Non-Playable Character (NPCs) below demonstrates the foundational concept of how Decision Tree can be implemented in game development:

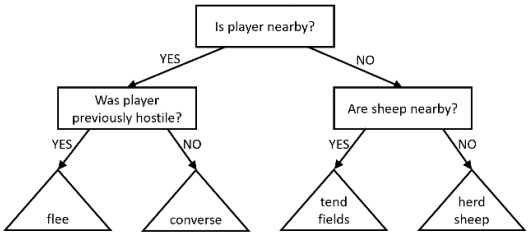

Figure VI Decision Tree used in NPCs (Non-Playable Characters)
Source: Yale University CPSC 223 - Data Structures and Programming Techniques on https://zoo.cs.yale.edu/classes/cs223/f2020/Exercises/20-questions.html

Above diagram explains how the NPCs should react if whether a player was nearby or not, there were two possibilities whether to flee or to converse. Then, it is also elaborated that how will the NPCs react if a sheep is nearby, whether to tend fields or to herd the sheep. Therefore, the developers can further modify and develop the Decision Tree to fit the game atmosphere or if they want to make the game as interactive and as realistic as possible. This general concept is what the next Chapters will be focusing on about but with a more optimized Decision Tree and more factors to be considered (elements, storyline, and in-game implementation).

III. Problem Decomposition

A. General Concept

As the author has stated in the previous chapter, there are integral parts in the Eidolon boss fight which consist of:

1. Damage Type
   The Eidolon has specific weaknesses, that mainly focuses on some specific element. Some damage types will deal more damage compared to the other. The right choice of damage type will damage the Eidolons faster and break the *Synovia* easier. *Synovia* is a special armor that protects the Eidolon specific parts which the player has to destroy before they can damage the Eidolon's health directly. If the player's damage type does not match with Eidolon's weaknesses, then the player's damage type will get a damage reduction. However, this also applies if the player's damage type does match with Eidolon' weaknesses, the player will obtain damage boost. Furthermore, Decision Tree will be used to determine which effect the player will get based on the damage types dealt from the weapons used. The effect within this scope is exclusively on damage dealt directly from the weapon (excluding animation effects such as the *Bleed* and any status effect immunity).

   Nevertheless, there are limitations for this scope as in the game itself damage types will be a complex factor. Therefore, the author will limit the scope of the buffs and penalties, in a condition only after the Eidolon shield is already depleted. It is important to note that Eidolon have a shield and health level, shield will keep regenerating if the Eidolon does not receive any damage after a moderate delay, the player has to deplete the shield before they can damage the health directly. Hence, player will need to damage the Eidolon's health level in order to kill the enemy, this phase will be the main focus of the author in analyzing the Decision Tree.

   There are other status effects that is not being calculated into the paper such as *Bleed* effect and status immunity of the Eidolons. Although it will damage the enemy's health, but it is not a directly-inflicted damage from the weapon.

2. Eidolon Lures
   Eidolon Lures are specific tools needed to capture the Trio Eidolon Bosses (Eidolon Teralyst, Gantulyst, Hydrolyst) which will be abbreviated as Tridolon. The player will be required to charge the Eidolon Lures by charging it with the Eidolon Vomvalysts, a special enemy which spawns on the Plain of Eidolons during night cycle. Moreover, after one *Synovia* is destroyed, the Tridolon will teleport to another location, which makes the player hard to track the location of Tridolon. Hence, in order to prevent it from teleporting away, it is necessary for players to bring enough Eidolon Lures. The player will need more than one Eidolon Lures in order to capture and prevent the Tridolon from teleporting away.

Figure VII Eidolon Lures on Warframe
Source: Warframe Wikipedia on
https://warframe.fandom.com/wiki/Eidolon_Lure

   Decision Tree will be used to determine whether the user has brought enough Eidolon Lures to prevent the Tridolon to teleport or not. As each Eidolon phases from Teralyst to Gantulyst and to Hydrolyst will require different number of Eidolon Lures.

Figure VIII Eidolon Vomvalyst on Warframe
Source: Warframe Wikipedia on
https://warframe.fandom.com/wiki/Eidolon_Vomvalyst

3. Reward Table
   The reward table depends on whether the player choose to capture or kill the Eidolon. The choice will yield different rewards, killing the Eidolon will not drop the item necessary to proceed to the next phase. Each captured Eidolon bosses will drop a *Radiant Shard* or *Brilliant Shard*. This item will be required for the player to progress to the next phase. The player will have the choice to stop or to proceed to the next phase. If the player chose to sacrifice the shard, they can proceed to the next phase, however player could also choose not to

sacrifice the shard and leave the mission. There will be a sacrificial altar in the game that can be accessed by the user in the middle of the map. The player can choose whether to use the shard on the altar or to leave the mission.

Figure IX Sacrificial Altar
Source: Author's personal documentation

B. Statistics and Essential Data

1. Damage Type Penalty and Bonus

Robotic Health	
Radiation	25%
Electricity	50%
Puncture	25%
Toxin	-25%
Slash	-25%
Alloy Armor	
Puncture	25%
Cold	25%
Radiation	75%
Slash	-50%
Magnetic	-50%
Electricity	-50%

Table I Damage Types (Left) and Effects (Right)

The Tridolon health consist of two components, which is Robotic Health and Alloy Armor. Each component will have its own effect as shown in Table I. On the left side of the table is the damage types and the right side is the bonus or penalty applied to the damage. However, there are many other damage types that are not listed, this means that the damage is neutral or will not be applied any additional bonus effect, whether it is a buff or penalty.

The concept is the health will be layered by an armor, that will provide a default damage reduction to any damage dealt to the enemy. Any damage dealt to the enemies will be directly dealt toward their health number, not armor. To make the concept clear, the formula is as follows:

$$ID = SD * DM \quad (1)$$

ID is Inflicted Damage. This will be the damage dealt by the player after all of the buffs and penalties are all calculated. SD is abbreviation of Starting Damage which implies the weapon's raw damage, that will be multiplied by the Damage Modifier or DM. The buffs and penalties listed above will be calculated in DM with the following formula:

$$DM = \frac{300}{300 + AR(1 - AM)}(1 + AM)(1 + HM) \quad (2)$$

AR is total target armor, with AM is additional buffs gained against the armor (Alloy Armor, in this case). HM is additional buffs against the health type (Robotic, for this case). This is due to the fact that each armor and health has its own buffs against some type of damage. This formula will be used in the Decision Tree to provide a clear demonstration of the cause and effect.

2. Eidolon Lures Requirement

Eidolon Type	Minimum Lures Requirement	
	Capture	Prevent Teleport
Teralyst	2	1
Gantulyst	3	1
Hydrolyst	3	1

Table II Minimum Eidolon Lures for each activity

Table II listed minimum requirements for each fully charged Eidolon Lures. Eidolon Lures will need at least three Eidolon Vomvalysts to be fully-charged and it will be indicated with a blue light on the Lures. Thus, if player brings lures which satisfy each requirement without being fully-charged, it will not be able to do the desired action.

Figure X Fully Charged Eidolon Lure
Source: Author's personal documentation

Figure XI Captured Eidolon Teralyst
Source: Author's personal documentation

3. Reward Table

Eidolon Type	Reward Table (Drops)	
	Captured (Table A)	Killed (Table B)
Teralyst	Eidolon Shard	Eidolon Shard
	Flawless Sentient Core	
	Brilliant Eidolon Shard	
	Teralyst Articula	
Gantulyst	Flawless Sentient Core	Eidolon Shard
	Eidolon Shards (3)	
	Brilliant Eidolon Shards	
	Radiant Eidolon Shards	
Hydrolyst	Eidolon Shards (5)	Eidolon Shards (5)
	Brilliant Eidolon Shards (2)	
	Radiant Eidolon Shards (2)	
	Riven Transmuters (2)	
	Hydrolyst Articula	

Table III Reward Table for each Eidolons

The reward table will be used for demonstration in the Decision Tree. For research purposes, the author has excluded other drops and only used some important drops that are related to the main topic. However, the drops listed in Table III should be sufficient to demonstrate the path after the player has chosen certain action.

IV. Decision Tree Design

The author will demonstrate the Decision Tree that can be utilized as the tool for creating possible algorithm for the problem described on Chapter III. There will be a Decision Tree for each category. The problem has been classified into three categories that is interconnected, each problem with its own Decision Tree, as in the figure below:

1. Damage Type Penalty and Bonus (Damage Calculation Tree)

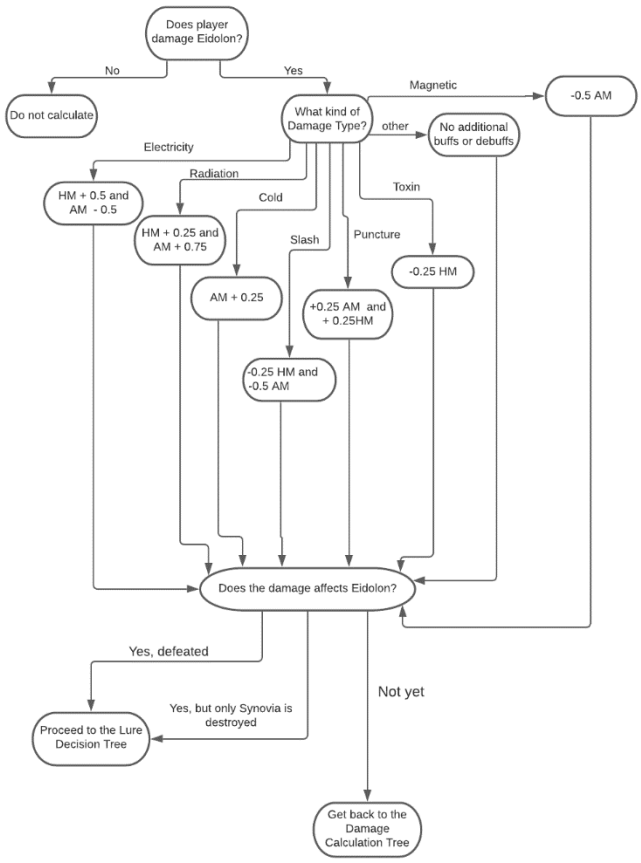

Figure XII Decision Tree for Damage Buff and Penalty
Source: Author's personal documentary

On the demonstration above, the damage bonus and penalty are based on formula (2) with (+) means user gains a bonus and (-) means user will get a penalty based on the percentage on Table III. If the damage does affect the Eidolon, then the program will execute the next sequence on the next Decision Tree, Eidolon Lures' Decision Tree. However, if the user attacked the Eidolon but the Synovia has not destroyed and the Eidolon is neither captured nor killed, the program will be back on calculating the damage done by the player.

2. Eidolon Lures

Below is the Decision Tree on how will the number of Eidolon Lures the player bring (must be fully-charged) affects how the program will execute the command.

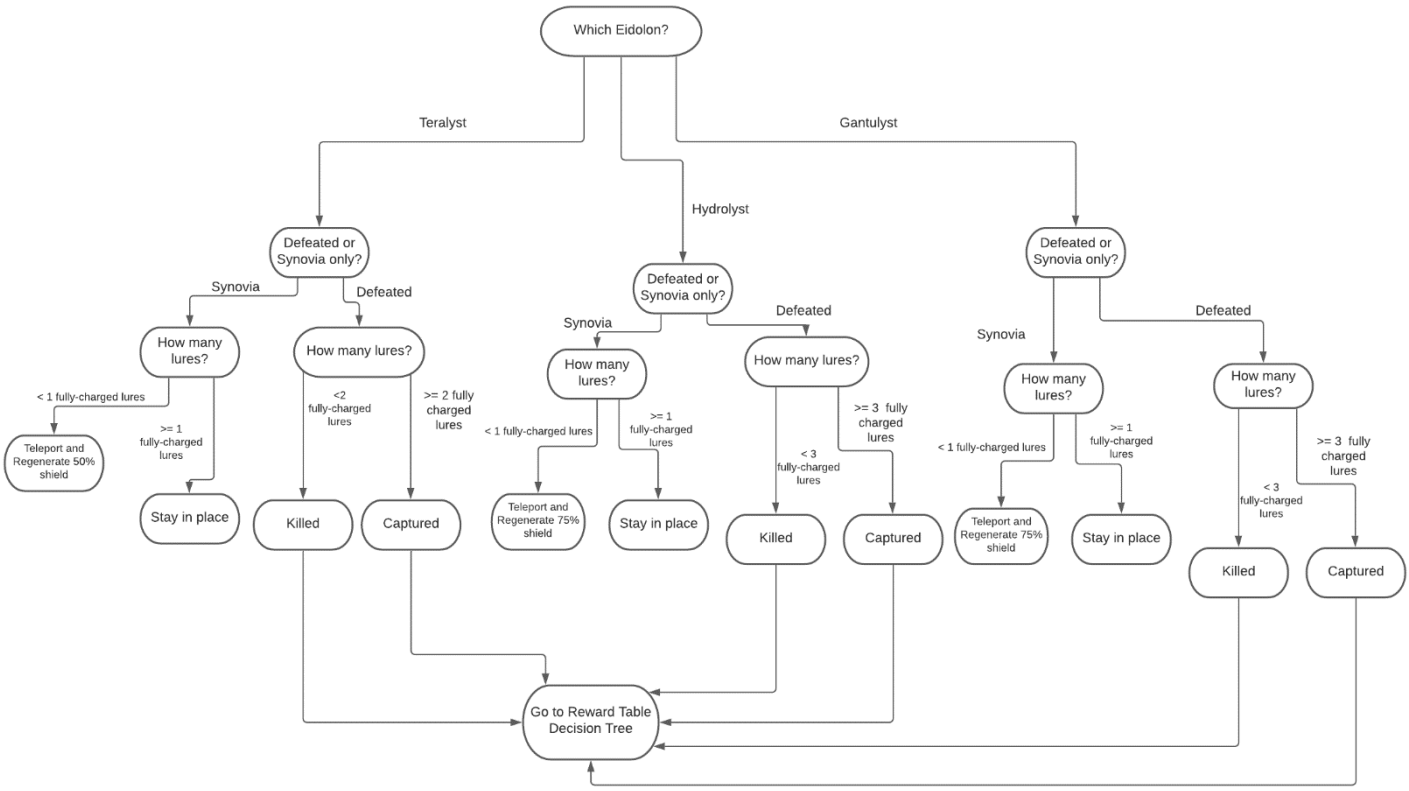

Figure XIII Decision Tree for Eidolon Lures
Source: Author's personal documentary

On each command path there are indeed some similarities on the result. However, the difference will be apparent as soon as the program begin to execute the command on the Reward Table Decision Tree. The Eidolon Lures Decision Tree was made based on the data from Table II. After the player has killed or captured the Eidolon, then the program will proceed to the next Decision Tree, the Reward Table.

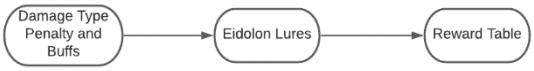

Figure XIV The Flow of Decision Tree
Source: Author's personal documentary

Figure XIV shows the flow of three Decision Tree that the author mainly focusing about. It also explains the possible structure of algorithm or the basic idea on how the actual program can be implemented.

3. Reward Table

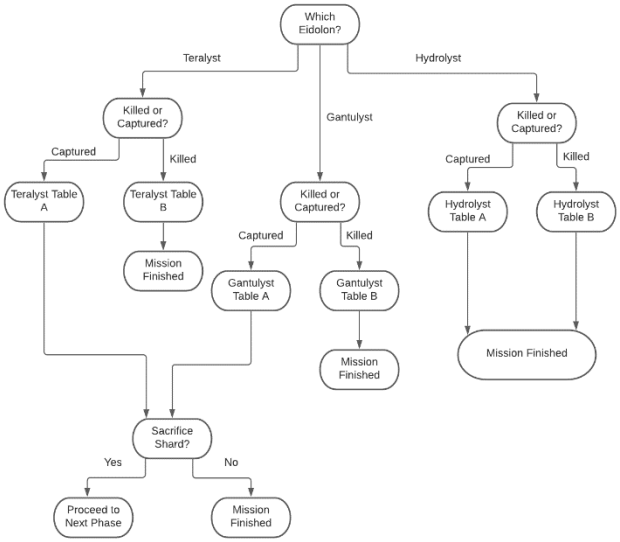

Figure XV Reward Table Decision Tree
Source: Author's personal documentary

Figure XV shows the design of Decision Tree for the Reward Table. The 'Table A' and 'Table B' are defined in Table III each with its own rewards, each Eidolon type will have the 'Table A' and 'Table B' with its

exclusive rewards respectively. The Decision Tree display the rewards that will be obtained by the player based on their decision to kill or to capture the Eidolon. If the Eidolon is killed then the mission will end (this is the case for Gantulyst and Teralyst). However, there is a special case for Hydrolyst whereas either captured or killed, the mission final state will be the exact same, that is finished, which indicates that Hydrolyst is the final phase of the Tridolon boss. Finally, the player will be given two options, to sacrifice the shard or to finish the mission without any penalties.

V. Conclusion

The use of Decision Tree will assist programmers to create the general concept of the game that will affect the maintenance process as well as the efficiency of the game. As the program become more complex, using Decision Tree will support the readability of the program itself. The use of Decision Tree can play a critical role in the topic discussed, which is focusing on Warframe's Plains of Eidolon, this is because many of the game mechanics will have to accept the player input and to give the player the output based on their choice—a very basic component of Decision Tree.

VI. Acknowledgement

The paper is finished because of God's grace and His blessing on every minute and hour the author has spent finding the materials needed on each problem. Furthermore, the author is honored to be given such opportunity by Bapak Rinaldi Munir to explore a very interesting course, Discrete Mathematics, through the lens of practical perspective. The author would also like to express our deepest gratitude to the Warframe community for providing such detailed information including the damage formulas on the newest game patch, as it has played a huge role on helping the author to use it as a reference for concept demonstration. Finally, the author would like to express our apology if there are still some flaws on the paper. Nevertheless, we highly hope that the paper could be used as a reference in the future or perhaps can be used by Warframe community for game research purposes.

References

[1] Yale University. *CPSC 223 - Data Structures and Programming Techniques.* https://zoo.cs.yale.edu/classes/cs223/f2020/Exercises/20-questions.html .

[2] Johnson, Paul. *MAS 341: Graph Theory*. https://ptwiddle.github.io/MAS341-Graph-Theory-2017/lecturenotes/lecture6.html

[3] Dale, M. (2017). Shapes of Graphs: Trees to Triangles. In Applying Graph Theory in Ecological Research (pp. 37-53). Cambridge: Cambridge University Press. doi:10.1017/9781316105450.003

[4] Warframe Fandom. *Damage*. https://warframe.fandom.com/wiki/Damage

[5] Digital Extremes. (2017, June 28). https://n8k6e2y6.ssl.hwcdn.net/repos/hnfvc0o3jnfvc873njb03enrf56.html

[6] Munir, Rinaldi. 2015. *Pohon (Bagian I)*. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/matdis21-22.html

[7] Munir, Rinaldi. 2015. *Pohon (Bagian II)*. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2021-2022/matdis21-22.html

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 14 Desember 2021

Nadia Mareta Putri Leiden (13520007)