

# Dynamic Game Difficulty Adjustment through Player Profiling: A Proof of Concept

Chadley Mercieca

*Institute of Information and Communication Technology*

*MCAST*

Paola, Malta

chadley.mercieca.a100026@mcast.edu.mt

**Abstract**—Game difficulty has always been an integral part of gaming. The fun factor and game difficulty usually intertwine together. This also depends on the player. Too much difficulty can be a problem for certain player, or if the players are exceptionally skilled, then the game can be too easy and lose its fun factor. A new method of determining the perfect game difficulty for players can be achieved by using a set of algorithms within the code that can acknowledge a series of variables of how the player is performing in a game. Game features can be changed accordingly to make the game easier or harder depending on the player's performance.

**Index Terms**—Algorithms, Game Design, Dynamic Game Difficulty Balancing, and Player Profiling

## I. INTRODUCTION

What is considered difficulty in games? What constitutes that difficulty is dynamic? How can player profiling play a role in difficulty? In this experiment, it is expected that once a player meets a set of goals in a certain amount of time, the player's performance will be profiled along a set of algorithms and the difficulty will adjust corresponding to the performance of the player. How effective will player profiling be in determining a fair and fun experience for the player? Can difficulty be distributed effectively by using a set of algorithms that are updated when the player is playing? My aim for this research is to determine if and how dynamic game difficulty balancing can replace fixed difficulty settings which are commonly found in modern games. Whilst various video game categories exist, I will be focusing my research mainly on how single player games can be affected either positively or negatively by dynamic game difficulty balancing.

## II. LITERATURE REVIEW

Game Difficulty is a concept that is not always translated properly because every player is skilled differently, but what if there was a way to scale a game's difficulty based on the player's different set of skills. However, in doing so will this translate to better gameplay or will it make it impossible for the player to adapt and learn how the game functions properly.

Identify applicable funding agency here. If none, delete this.

Game designers are tasked with creating a balanced and fun experience that is able to grip players in their world. Difficulty can be done in various methods to be compelling for the player. This is also extremely dependant on which game type it is. The most common difficulty setting is usually in regard to how much damage the player takes in the game. If the game is set to the Easy setting, then the player takes minimal damage per hit and the Game AI has a slower reaction time, makes more mistakes and has less precision, however if it is set to the Hard setting, then the player takes maximised damage per hit and the Game AI has a much faster reaction time, rarely makes mistakes and has high precision. Dynamic game difficulty balancing will disable the Easy/Normal/Hard settings we have all grown used to and instead opt for an option less setting that will adjust to how you are performing with the use of player profiling. The adaptable difficulty can make the game world feel more dynamic as it will change to easier or harder depending on the player's skills so the game can never feel like the game is too easy or too hard. The main goal for dynamic game difficulty balancing is to avoid dullness and keep the players interested in the game for a longer time. Dynamic game difficulty balancing (DGDB) can manage various elements of the game such as: Speed of enemies, Health of enemies, Frequency of enemies, Frequency of powerups, Power of player, Power of enemies, Duration of gameplay experience and more. In order to achieve this ability in your games, one must use player profiling in order to track the player's performance throughout the game. This is where various algorithms can be inserted in the game. <sup>[5]</sup>An example of an algorithm would be where it would calculate the rate of successful shots before finishing the round. The game's difficulty can be scaled in such a manner with this formula:

$$A. \text{ successfulshots} = \text{totalshots} - \text{missedshots}$$

This formula would then be able to trigger other events depending on how many successful shots were performed in a round. If the 'successfulshots' threshold reached a certain high number, then the difficulty would increase or decrease if they reached a bad score. The 'successfulshots' threshold

would be scaled on how many average shots it takes to kill all enemies in a round, therefore being able to adjust the difficulty accordingly on how well the player had played the previous round. Similar algorithms can be used as well to add difficulty scaling during a match and not relying on ending rounds to increase or decrease difficulty. Instead having the ability to balance some difficulty aspects while the game is being played.<sup>[9]</sup> Some problems in content quality may be encountered when taking this approach. <sup>[6]</sup>While the game is running, the game may increase or decrease difficulty in an unrealistic or chaotic manner, therefore rendering the game unenjoyable or even unplayable in some occasions. There is another approach to dynamic game difficulty balancing found in some games. One notable game using niche ideas to make it easy or harder for the player would be considered to be Left 4 Dead 2. Even though this game has the options to do a campaign in either Easy, Normal, Expert and Realism modes that handle the general amount of health that zombies have and also how much damage they deal to the player, however an interesting example where dynamic difficulty is being used is where the player can be punished by a special zombie secretly in the safehouse that immobilises and kills the player if they are not helped by another player. The main reason being for leaving their teammates behind and rushing to the safehouses by themselves. Another way to challenge the player is that a horde of zombies will spawn and rush into the players if they are just not moving around the map enough. It generally gives a sense of direction and a sense of danger for the players who prefer to stay safely in a room instead of progressing to the checkpoints. The general consensus of how player profiling data will be gathered during play will be depending on how many different algorithms are operating in the game. The more algorithm aspects, the better the chance to add more difficulty aspects, such as increased or decreased values of health, damage, speed, or the number of zombies. Simple action vs reaction formulas would be very easy to implement in such games as either 'first person shooter' or 'button masher action games'. Using simple formulas, such as: Requiring a number of kills or missed shots in a level to apply an easier or harder difficulty, can be implemented in many game genres and such an aspect will be easy for developers to implement in their games. However, it is always important for proper game balancing, such as "Not too easy, not too hard, just right!". Unfortunately getting this perfect balance is not such an easy task because every player is skilled differently, and maybe some game features were not implemented yet, therefore the game would not be able to be tested accordingly. In online games, <sup>[6]</sup>player profiling can be further collected by using machine learning techniques since every action by a player is recorded and can be translated to extremely rich datasets. Different outcomes can be reached by different players depending on their total lifetime of playtime, in-game progression, and session playtime in one sitting. This is mostly viable in an online service game since their every action is recorded live in a player log that can be accessed by the developers by usage data. Using graph tables to calculate

these variables will indicate better who your audience is and in turn will show how difficulty should be managed by the developers. Having such data will enable them to cater their difficulty to their main audience. However, in doing so it can also make the other minorities of players feel left out if they are hardcore gamers or casual gamers and vice versa. An example can be the popular franchise of Dark Souls which is clearly more suited for veteran players as its difficulty is a staple that many games try to compete with. This birthed the idea of 'Souls-like' games and in this approach to video games it can also have a negative impact on the general consensus of video games. This is due to the fact that many game reviewers are considered casual gamers, therefore deeply intricate role playing, and difficult games can suffer a negative review and end up with an average score that is not acceptable by their publishers and the developers can end up losing all their funding and become bankrupt. This is a huge problem looming in the game development industry as catering for both audiences is a very difficult task to do. Optional dynamic difficulty may be able to fix this glaring issue in modern games so they can appeal to casual gamers as well. The results in taking this approach will most definitely have a positive or negative outcome in regard to general player immersion and gameplay. While the player is learning how to play the game, the algorithms may see that the player is performing well, and the difficulty increase will throw the player off balance, therefore it may not always be the best case to use adaptive difficulty in some games. I will take what I have learnt from my research to implement in my methodology.

### III. RESEARCH METHODOLOGY

The main methodology for this study will require a properly functioning zombie first person shooter game prototype with elements of game AI in conjunction to new built algorithms to collect player data in order to handle difficulty balancing accordingly. Know-how is required in how algorithm variables can be handled globally between scripts and kept as simple as possible to avoid creating new errors in code. Object Oriented Programming knowledge is a crucial benefit for game development. The coding language that will be used to construct the game is C Sharp. The main software required for this study will be Unity 4.2 to serve as the game engine for this prototype. Visual Studio for scripting purposes (Game development) and later on implementing the algorithms in question. Unity assets for game production. The game in question was already developed in a three-man group effort for our Game Level Design subject we had done earlier this year. The version I will be using for this study will be a modified version of the game, which only features one level and will repeat when the level is finished. The main hardware components required for this study will be a gaming PC capable of running Unity and the game prototype. The algorithms will handle how much damage the player's pistol deals and less damage intake from the enemies depending on the player's performance. If the player hits 5 headshots, then the game will reduce the damage the player deals. If

the player misses 5 shots, then the player will increase the damage the player deals. If the player dies 3 times, then the enemy damage is reduced, however if the player manages to finish the level, the damage the enemy deals will reset back to normal. The level in the game prototype requires the player to shoot zombies and find three bombs in the process to escape the asylum. After finding these bombs, the player must then attach these bombs to the moss wall in the main hall. This is when the player is prompted to detonate the bombs and therefore needs to find a detonator. If they manage to find the detonator then they will win the level and the game resets. After resetting, if the player died three times in the process and reduced enemy damage to the player is active, then the enemy damage gets reset back to the original damage numbers.

#### A. Algorithms

Code snippets to how the difficulty balancing will work is found below:

When the player dies three times, it decreases the enemy damage to 5 points instead of 13 points.

---

```
if (PlayerHealth.PRdeathCounter >= 3)
{
    Debug.Log("Player has died more than 5
        times... Decreasing enemy damage");
    damageAmount = 5f;
}
```

---

When the player manages to finish the level, it increases the damage back to the normal 13 points and resets the level.

---

```
if (Input.GetKeyDown(KeyCode.E))
{
    Debug.Log("Player has finished the level...
        Increasing enemy damage");
    enemyAI.damageAmount = 13f;
    PlayerHealth.PRdeathCounter = 0;
    SceneManager.LoadScene("Lv11");
}
```

---

When the player dies, there is a death counter being incremented +1.

---

```
if(currentHealth <= 0 )
{
    PRdeathCounter++;
    Deaths++;
    Application.LoadLevel(Application.loadedLevel);
}
```

---

When the player scores 5 headshots, the damage to body hits is decreased to 10 points from 20 points originally. It requires that the variable PRhitHeadshot needs to be 5 or more.

---

```
if (PRhitHeadshot >= 5)
{
    Debug.Log("Hit Headshot = 5 or more...
        Decreasing damage");
    damageEnemy = 10f;
}
```

---

This piece of code handles how the gun operates normally when it hits an enemy.

---

```
if (Physics.Raycast(shootPoint.position,
    shootPoint.forward, out hit, weaponRange))
{
    print(hit);
    if (hit.transform.tag == "Enemy")
    {
        Debug.Log("Zombie hit");
        EnemyHealth enemyHealthScript =
            hit.transform.GetComponent<EnemyHealth>();
        enemyHealthScript.DeductHealth(damageEnemy);
    }
}
```

---

If the player misses a shot then, the PRhitElse variable will increment by 1.

---

```
else
{
    PRhitElse++;
    Debug.Log("Hit Something Else");
}
```

---

This piece of code shows how the headshot works normally if the player hits the enemy head. PRhitHeadshot variable will increment by 1 and PRhitElse will decrement by 1.

---

```
if (hit.transform.tag == "Headshot")
{
    PistolHeadShotTracker.AddHeadshotPistol();

    PRhitHeadshot++;
    PRhitElse--;
    Debug.Log("Headshot");
    EnemyHealth enemyHealthScript =
        hit.transform.GetComponentInParent<EnemyHealth>();
    enemyHealthScript.DeductHealth(damageHeadshot);
}
else
{
    Debug.Log("Not headshot");
}
```

---

#### B. Main Difficulties

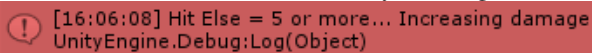
Developing the game took about 2 to 3 months to complete in a group effort of three people as making an FPS game requires a lot of various knowledge and it took countless hours to create and test. A lot of difficulties were encountered while designing the game and implementing the shooter elements and objective based functions. The biggest issues we had was with pathfinding for the enemy AI and making guns function properly with animations playing at the right time. After rigorously developing the game I could finally start implementing the Dynamic Game Difficulty Balancing algorithms and game modifications to run and test these algorithms. Fortunately, my study was about implementing different algorithms to handle dynamic difficulty balancing and luckily this was the easiest part in the whole production of the game as they gave me the least amount of trouble and time to create compared to

other aspects of game development/design. This means that many game developers can easily include such features in their games.

### C. Testing

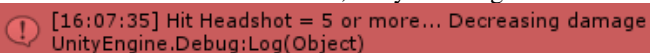
After implementing the algorithms, they worked pretty well in the testing phase. When I missed 5 shots, the player damage increased, when I hit 5 headshots, the player damage decreased. When I died 3 times, the enemy damage was reduced significantly and when I finished the level the enemy damage increased back to its original number.

After I missed 5 shots, my damage is increased



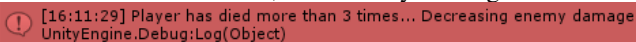
I was able to kill zombies in two shots with this change

After I hit 5 headshots, my damage is reduced



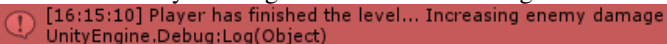
This change made the enemies a much bigger threat as they were much more impervious than before.

After I died 3 times, the enemy damage was reduced



This change made my health decrease a lot slower than before after taking hits.

After gathering the three bombs, attaching the bombs, and detonating the bombs, the level was reset, and the enemy damage was reset to original values.



This change made the game reset the damage taken by the player to be reset back to normal.

### IV. EVALUATION

After gathering the data from the tests I have performed, the tests resulted as a success and algorithms worked as intended, however I do question their viability in such a game. I fear that it would make the game too easy and probably eliminate the challenge and genuine fun or scary factor it brings. However, this sort of difficulty balancing would be very useful in the game testing phase when a game is still in production as it would be frustrating for the developer to test if they have not yet implemented cheats in their game. I want to see how another types of game would perform with Dynamic Game Difficulty Balancing. A shooter game with DGDB would probably not work as intended therefore it is to be avoided if not necessary, however it is not always the case. If more balancing algorithms are added in the game and maybe ones that are not too game-changing, then they would be a welcome feature. Inherently, many more algorithms could have been developed to create an even more dynamic game experience,

however I wanted to not include too many algorithms or changes as it would render the experience to become chaotic instead of feeling like difficulty balance. I feel like dynamic game difficulty balancing would be a very welcome addition to mobile games as it is usually more targeted towards casual gamers. Another addition to the balancing that would have been beneficial would have been a prompt asking the player if they want to reduce the difficulty of the game, instead of automatically reducing the game difficulty. Another addition can be the idea to reduce rewards if the player chooses to reduce the difficulty so the player can still be determined to try to overcome the challenge of the game in its original difficulty.

### V. CONCLUSION

I have used a formula of a first-person shooter game to see whether dynamic game difficulty balancing was a viable option for optimising game difficulty. There is not a fully clear definition whether this approach has been a success or not because this is determined with different players. Every player is different and may prefer or dislike this option in games. There is also a percentage of people who enjoy a challenge when playing games and are usually solely interested in getting challenged and overcoming these challenges, therefore if the game keeps getting easier when they lose, they may lose their purpose of playing the game. At the end, this option is better included as an optional feature for certain games. Future work includes exploring different ideas for including other algorithms and further improvements for the game itself to handle these balancing formulas better. Further exploring other methods in including these formulas, such as time based or a more conditional approach to algorithms. Further study into how dynamic game difficulty balancing would affect other game genres such as strategy, fighting games or mobile games in general.

### VI. ACKNOWLEDGEMENTS

I would like to thank Chris Anthony Borg and Liam Laus for being part of the game development team. I also wish to thank Ivan Briffa for mentoring me on the paper.

### REFERENCES

- [1]Adams, R., 2020. 10 Powerful Examples Of Artificial Intelligence In Use Today. [online] Forbes. Available at: <https://www.forbes.com/sites/robertadams/2017/01/10/10-powerful-examples-of-artificial-intelligence-in-use-today/>; [Accessed 4 June 2020].
- [2]Baldwin, A., Johnson, D., Wyeth, P. and Sweetser, P., 2020. A Framework Of Dynamic Difficulty Adjustment In Competitive Multiplayer Video Games. Undergraduate. Queensland University of Technology.
- [3]Del Rio, Ana Fernandez, Pei Pei Chen, and Africa Perianez. "Profiling Players with Engagement Predictions." 2019 IEEE Conference on Games (CoG) (2019): n. pag. Crossref. Web.

<sup>[4]</sup>Hunicke, R., LeBlanc, M. and Zubek, R., 2020. MDA: A Formal Approach To Game Design And Game Research. Undergraduate. northwestern.

<sup>[5]</sup>Guillermo Gomez-Hicks and David Kauchak. 2011. Dynamic game difficulty balancing for backgammon. In Proceedings of the 49th Annual Southeast Regional Conference (ACM-SE '11). Association for Computing Machinery, New York, NY, USA, 295–299. DOI:<https://doi.org/10.1145/2016039.2016115> <https://arxiv.org/pdf/1907.03870.pdf>

<sup>[6]</sup>Jenkins, H., 2020. Game Level as Narrative Architecture. FIRSTPERSON, 1(1), pp.118-128.

<sup>[7]</sup>Muliawan, F., 2020. Enemy Speed Control On Shoot Em' Up Game With Fuzzy Takagi Sugeno Method. Undergraduate. Faculty Computer Science, Sriwijaya University.

<sup>[8]</sup>Rummell, P., 2011. Adaptive AI to Play Tower Defense Game. CGAMES, 16(1), pp.1-3.

<sup>[9]</sup>Shi, P. and Chen, K., 2018. Learning Constructive Primitives for Real-Time Dynamic Difficulty Adjustment in Super Mario Bros. IEEE TRANSACTIONS ON GAMES, [online] 10(2), pp.155-156. [Accessed 4 June 2020].

<sup>[10]</sup>Silva, M., do Nascimento, V. and Chaimowicz, L., 2015. Dynamic Difficulty Adjustment Through an Adaptive AI. Brazilian Symposium on Computer Games and Digital Entertainment, 14(1), pp.1-10.

<sup>[11]</sup>Spronck, Pieter Sprinkhuizen-Kuyper, Ida Postma, Eric. (2004). Difficulty scaling of game AI. Proceedings of the 5th International Conference on Intelligent Games and Simulation.