Using Dynamic Difficulty Adjustment to Improve the Experience and Train FPS Gamers

Johann Knorr
GILT-Instituto Superior de Engenharia do Porto
1160996@isep.ipp.pt

Carlos Vaz de Carvalho
GILT-Instituto Superior de Engenharia do Porto
cmc@isep.ipp.pt

ABSTRACT

The concept of Dynamic Difficulty Adjustment (DDA) refers to software techniques that make a game adjust itself in real time to make the game better fit the player's abilities and therefore to improve his/her gaming experience. But DDA can also be used to improve and render more effective the training of e-sports players by creating even more challenging environments that always push the gamer to his/her limits. This article presents a first step towards that objective: in this study, the First-Person Shooter (FPS) game Half-Life was modified to allow several game mechanics (and, accordingly, the game's difficulty) to be adjusted in real-time creating what was expected to be an improved and more immersive experience to the player. The DDA algorithm makes these adjustments according to the previous actions of the player and towards a predefined performance goal. Preliminary results with a group of e-sports players showed positive results in terms of their perceived experience.

CCS CONCEPTS

• **Human computer interaction**; • **Heuristic Evaluations**; • **Empirical studies in interaction design**;

KEYWORDS

Dynamic Difficulty Adjustment, Serious Games, Game-Based Training, First-Person Shooter

ACM Reference Format:
Johann Knorr and Carlos Vaz de Carvalho. 2021. Using Dynamic Difficulty Adjustment to Improve the Experience and Train FPS Gamers. In *Ninth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'21) (TEEM'21), October 26–29, 2021, Barcelona, Spain.* ACM, New York, NY, USA, **??** pages. https://doi.org/10.1145/3486011.3486447

1 INTRODUCTION

Video-game players are motivated to play games by different reasons (which justifies the existence of widely different game genres) and several taxonomies and frameworks have been proposed to describe and justify those differences focusing mostly on the psychological traits of the users. Probably the most well-known taxonomy is the one proposed by Richard Bartle that, although initially thought for Multiuser Dungeons (MUD) players, has since been extended to all video game players [1]. Bartle classified players according to their psychological characteristics as Achievers (want to get points, badges, levels, status), Explorers (want to see new things and discover secrets), Killers (want to get points and status by competing and beating opponents) and Socializers (have fun by interacting with other players).

Figure 1: Bartle's taxonomy of player types [1]

The taxonomy is represented in a quadrant model (Figure 1) with the horizontal opposition of players that prefer to relate with the other players vs. players that prefer to explore the game worlds and, vertically, the opposition between players that prefer to play alone vs. the ones that prefer multiplayer settings. So, Socializers are naturally in the quadrant of players that like to interact with other players in multiplayer settings while Achievers are exactly on the opposing quadrant, preferring to play alone and explore the world. Normally, a player profile is not restricted to a single quadrant but it is rather a combination of the different characteristics with different weights.

Bateman and Boon initially proposed a somehow similar taxonomy with the player styles of Conqueror, Manager, Wanderer and Participant but later expanded it to the BrainHex model which includes six different types: Achievers (goal-oriented and motivated by completion), Conquerors (enjoy struggling against strong opponents), Daredevils (motivated by excitement and risk taking), masterminds (enjoy solving puzzles and devising strategies), Seekers (motivated to explore the game world), Socializers (enjoy interacting with other people) and Survivors (enjoy frightening experiences in games) [2][3].

Yee developed a gamer motivation profile with six clusters: Action (destruction and excitement), Social (competition and community), Mastery (challenge and strategy), Achievement (competition and power), Immersion (fantasy and story) and Creativity (design and discovery) [4]. Lazzaro tried to relate emotions with the motivation for playing games and found 4 different ways or keys that make players experience that: the Internal Experience Key (Emotions are generated with Perception, Thought, Behavior, and Other People), Hard Fun (Emotions are generated from Meaningful Challenges,

TEEM'21, October 26–29, 2021, Barcelona, Spain

ACM ISBN 978-1-4503-9066-8/21/10.
https://doi.org/10.1145/3486011.3486447

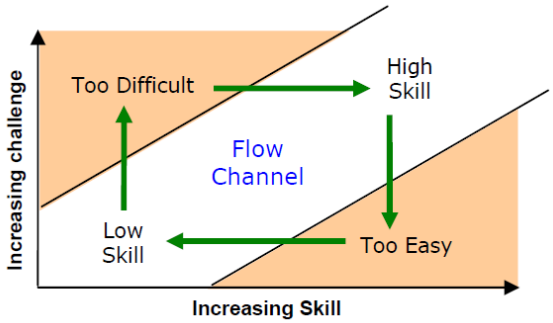

Figure 2: - Game Flow State [8]

Strategies, and Puzzles), Easy Fun (Emotions are generated from Ambiguity, Incompleteness, and Detail) and the Social Experience (Emotions are generated from Player Competition, Cooperation, Performance, and Spectacle) [5].

To maximize the player fun, game experiences must be created that adhere to his/her preferences but also that push the player's skills to the maximum so that they enter a state of mind that stimulates their creativity, enjoyment, and learning [6]. This is colloquially known as being in "the zone" or being "in flow". Flow, as proposed by Csikszentmihalyi, is a state in which a person voluntarily invests his/her attention in a certain task, becoming totally absorbed in the experience and ignoring external stimuli [7]. This optimal state results from experiences in which one is focused on goals that can realistically be achieved and one's skills are sufficient to partake in the available opportunities. As a person partakes in such an experience, his/her skill level increases, which in turn means that the necessary challenge must increase at a similar rate, avoiding situations of distress (challenge too hard) or boredom (challenge too easy), to guarantee that the Flow state is maintained (Figure 2). The appeal for this optimal experience results from its intrinsic rewarding nature and the corresponding activities are thus undertaken for their own sake.

1.1 Dynamic Difficulty Adjustment

The concept of Dynamic Difficulty Adjustment (DDA), also known as Dynamic Game Balancing (DGB), relates to the use of techniques that make a game adjust itself in response to player actions to better fit the current player's skills and to avoid boredom or anxiety [9] while keeping the game balance and feedback [8]. One well-known simple example of DDA is "rubber banding", used primarily in racing games, where the vehicles that are placed behind the player receive speed boosts, while the vehicles placed ahead are slowed down [10]. Unfortunately, with simple DDA techniques like this one there is the risk that the players become aware and start using it as another game mechanic which disrupts the intended game design and eventually the game balance (for instance, players realize quickly that they are not able to distance themselves too much from the followers and that they will also not lose much ground to the leaders, so it is better to have a relaxed game attitude and just try to win on the final lap).

Once there is a multitude of game genres, with different dynamics and mechanics that cater for completely different individual player preferences and motivations, game development studios and researchers have experimented with different ways to apply DDA. For instance, the sixth major instalment in the Resident Evil series, a survival horror, third-person shooter, developed by Capcom in 2005 displays some DDA techniques: the player selects a difficulty at the start of the game; however, the difficulty can then increase or decrease within certain thresholds, according to the player's actions. The game alters the reaction time of the enemies, their health and strength, as well as the items the player receives. If the player is performing well, the enemies will do more damage, have increased health and decreased reaction time. The player will receive fewer items, and, in some cases, more enemies will be present. Conversely, should the player be performing poorly, the opposite actions will be taken [11]. The Naughty Dog Studio introduced some DDA by controlling game objects (dynamically changing the number of checkpoints, the number of power-ups, or the speed of obstacles, for instance) according to the performance of the player ". . .help weaker players without changing the game for better players" [12]. The EA game company has actually just been granted a patent for DDA that is meant to ". . .review historical user activity [. . .] to generate a game retention prediction model to [. . .] automatically adjust the difficulty level of the video-game" [13].

On the academic side there have been also several interesting attempts: the Hamlet system, developed for Half-Life, is a set of libraries embedded in the game's engine that monitors several metrics to estimate how well the player will adjust to a future situation in the game [8]. If the system determines that the player is on a path that will lead them to fail repeatedly, it will adjust the game accordingly. The assessment of the player's situation is made by observing the state of their inventory, tracing how the player has used their items so far, watching for possible shortfalls, and by the damage, the player has been taking in previous encounters. If an unpleasant situation is detected, Hamlet may either take proactive or reactive actions. Reactive actions will adjust elements that are currently present, like enemies that are aware of the player's presence, by modifying the accuracy, or the damage of the enemy's attacks or their health. Proactive actions modify elements that are not yet visible to the player, which might entail changing the type and number of enemies that appear as well as their accuracy and damage (before they appear).

Different Artificial Intelligence techniques are also being used for DDA like artificial neural networks, genetic algorithms, fuzzy neural networks, machine learning, reinforcement learning and others as shown by the works of Demasi and Cruz [14], Spronk [15], Yannakakis and Hallam [16] and Andrade et al [17].

2 IMPLEMENTING DDA IN A FIRST-PERSON SHOOTER

This study aims to answer the question on how can the difficulty of a game be adjusted dynamically to improve the player's gaming experience and/or to foster his/her skills in a game (or even in a game genre)? For this purpose, it was decided to use a First-Person Shooter as a case study considering the large number of e-sports players dedicated to this game genre. In particular, the existing

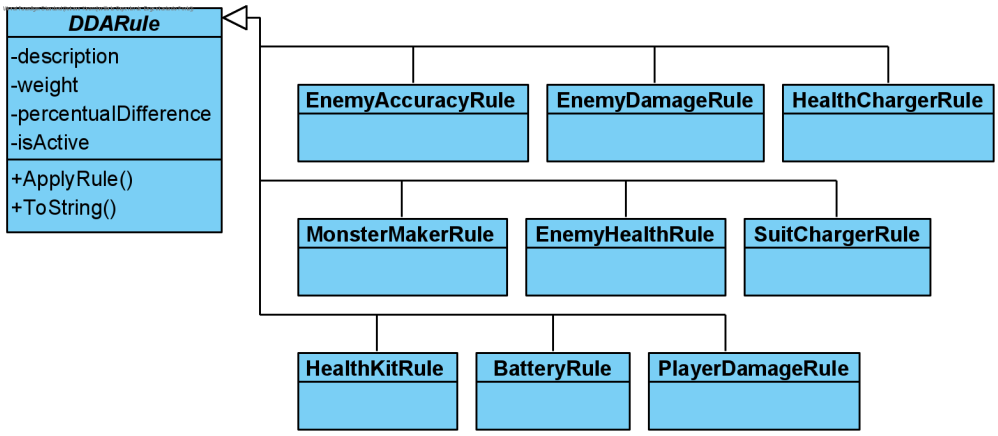

Figure 3: - Dynamic difficulty rules diagram

video-game Half-Life was chosen because it is a well-known game with a lot of players, it was possible to access the game engine code and the second release of the game, Half-Life 2, already included some DDA features that could potentially be used to compare the results of this study.

Half-Life is a First-Person Shooter (FPS) originally developed by the company Valve and published by Sierra Studios. The player takes control of a single character, Gordon Freeman, a scientist who must escape a research facility after it has been invaded by aliens. The character is equipped with weapons and armor which should be used to keep the player alive. The player is hindered in the progression of the game by multiple encounters with enemy non-playable characters (NPC) who will attack the player on sight.

The implementation of DDA techniques in the modified Half-Life focused on two aspects:

- The first part is responsible for changing combat actions (damage output, enemy hit-points, accuracy, etc.), by creating difficulty modifying rules selected according to the player's performance. These rules have a weight property which increases on successes and decreases on failures and it is used to perform a weighted random selection of the rules.
- The second part analyses the player's inventory status and determines which items are most beneficial to the player at any given time and modify item drops accordingly.

The developed DDA Rule System consists of a list of 155 rules of 9 different types (Figure 3). At any given time, one rule of each type is active, allowing to change the difficulty through different means: enemy accuracy, enemy damage, enemy health, player damage, enemy spawning, and item effectiveness.

The rules are stored in a text file and can be modified at any time. This allows to rapidly deploy a different set of modifiers or change the number of rules. The selection of the active rules are active and their weight can be triggered in three different ways: repeated player death, check-point completion or level completion.

As the player progresses through the game, local and global tracking of his/her performance facing challenges is done through an integration with the GameAnalytics API which allows to gather global statistics on the player [18]. Depending on the player's statistics it is determined if the attempt was successful, which results in an increase of the active rules weight, or unsuccessful, where the active rules are punished while the inactive rules receive a weight increase depending on if the difficulty should be increased or lowered. For this solution, checkpoints were considered when the game performs the auto-save function. Determining the success of the run is done by analysing how often the player has died since the last rule change, how much damage the player has received and the time between player deaths. A player who rarely dies is considered to have an adequate challenge, while a player who dies repeatedly or does not die at all for an extended period, is considered to have an unmatching challenge. Finally, there is also the option to delegate the rule change to the next trigger in case nothing relevant happened since the last rule change.

Should the system determine that an action must be taken to modify the weights of our difficulty rules, then three scenarios are possible: in the first scenario, the rules are considered to be adequate so the weight of the active rule and of the two rules that surround it is increased by a flat value of 10. In the second scenario, the selected rules created a challenge that the player could not overcome so the following two expressions are used to increase the weight of easier rules and decrease the weight of more difficult rules.

$$W = \frac{W' + 10}{|d| \times c}, \ \forall\, d \in \mathbb{R}_{<0}$$

$$W = \frac{W' - 10}{d} \times c, \ \ \forall\, d \in \mathbb{R}_{>0}$$

Where W is the new rule weight, W' represents the old weight, d represents the distance between the current and active rules and c represents the value obtained from *ShouldChangeDifficulty* function, which in this case is a value between 0 and 1.

Lastly, in case the offered difficulty was not sufficient to challenge the player, two expressions similar to the previous ones are used to decrease the weight of easier rules and increase the weight of more difficult ones, based on their distance to the active rule.

$$W = \frac{W' + 10}{|d| \ \times c}, \forall\, d \in \mathbb{R}_{<0}$$

$$W = \frac{W' - 10}{d} \times c, \forall\, d \in \mathbb{R}_{>0}$$

Where d, again, represents how far the rule is in the ordered rule list and c represents the value obtained from our *ShouldChangeDifficulty* function, which in this case can be any value greater than 1.

The second system, responsible for determining and modifying item drops, works by analysing the player's health, armour status and inventory to ascertain what items would be most beneficial to the player in a certain moment. Items are dropped to the player upon the death of certain NPC and when the player breaks predetermined crates. Using the information on the player's inventory, each item is given a weight and the selection of the item to be dropped is done using a weighted random selection. If the player's health and armour levels are high, healing items are restricted from spawning and, likewise, only ammunition suitable for weapons that the player already possesses and for which they do not already carry the maximum allowed amount is spawned. It was also determined that alien items or any new weapons would be included in the list of items as giving these items to the player would interfere with the game's narrative and progression.

The analytics system, created to gather data that allows to make inferences about the behaviour of the players and how to further enhance the difficulty modifying system, makes use of the service offered by GameAnalytics. As the events that are being tracked occur in-game, a string representation of that event is then pushed to the GameAnalytics server. The GameAnalytics SDK, responsible for the communication between the local machine and the remote server gathers these events and submits them every 8 seconds. Should the server be unreachable, then the events are stored locally so they can be pushed again at a later date.

Each event string is composed of two or more fields. The first field represents the type of event that is being recorded, while the later fields record additional information of that event. Additionally, events can also have a numerical value associated with them. For example, the event that tracks rule selection is composed of three fields: event_selected_rule:ruleName:rule_modifier. This allows to track which rule modifiers are selected most often for each type of rule and make inferences based on that.

2.1 Results

The evaluation of this project's effect on the player's experience was performed by pilot testing with a group of players (15 participants), selected from volunteers that match our target audience: men or women, between the ages of 18 and 30 years of age, with prior extended experience with FPS games. Participants were then subject to two one-hour play sessions of the game: in one session, players were presented with the original version of the game and in the other session players used the DDA version of the game. Users were not told which version of the game they were playing in each session and the order they played each version of the game was randomized to reduce bias in their perception. After each session, players answered two surveys, the System Usability Scale (SUS) and the Game Experience Questionnaire (GEQ) [19] [20]. By comparing the results from each session, it is then possible to verify the performance of the algorithm in improving the player's experience and compare the usability of the game.

The GEQ tool assesses the gaming experience through eleven variables: competence, immersion, flow, annoyance, challenge, negative affect, positive affect, positive experience, negative experience, tiredness and returning to reality. Some of these variables translate a positive game experience and some relate to a negative game experience. The GEQ survey was composed of two modules, with questions C1 to C33 belonging to the core module of the GEQ and questions P1 to P17 belonging to the post-game module. The score for each variable is obtained by calculating the average of the related questions for each participant and then calculating the average of those values for all the participants (Figure 4).

Figure 4 presents the results for all the GEQ variables. For positive variables a positive difference between the normal Half-Life game and the DDA Half-Life game is good while for negative variables (Annoyance, Negative Affect, Negative Experience, Tiredness) negative differences are good. It can be seen that the DDA implementation fared better than the original version in 8 out 11 metrics. The most striking result of the implementation corresponds to the Challenge variable which has been considerably increased. Accordingly, the competence metric has been somehow reduced as players felt less in control of the game and that resulted in Annoyance and Tiredness. The other metrics were quite positive, particularly in terms of Flow and Positive Experience so we can consider the overall result to be very good and according to the original expectations.

As mentioned before, the usability of the game was assessed through the System Usability Scale, a validated tool consisting of a 10-item questionnaire using a 5-point Liker scale, from Strongly agree to Strongly disagree. Besides the individual evaluation of each item answers, SUS provides an aggregate score ranging from 0 to 4 with 4 being the most positive. To get this score, individual item scores are normalized and the scores from the negative statements are inverted. Finally, the score is converted in a 0-100 scale, with the following cut-off: above 80.3: excellent; between 68 and 80.3: good; 68: ok; between 51 and 67: poor; below 51: awful.

In this study, more than really looking for a more "usable" game it was more a matter of trying to understand if the introduction of the DDA features would have somehow introduced some issues which could have given the users the feeling or perception of a less "usable" game. Figure 5 shows the scores reported for each question as well as the final result which actually shows a decrease in the usability of about 3.67 points, from 84.50 to 80.83, which can still be considered a good result as the relative change is less than 5%. Therefore the DDA implementation introduced no major changes in the perception of the players about the game.

3 CONCLUSIONS

Video-game players play different genre of games because their motivations and preferences are very different. To maximize the player game experience and fun, the game elements should adhere to his/her preferences but should also push the players' skills to the maximum so that they enter and remain in "the flow". Dynamic Difficulty Adjustment techniques make a game adjust itself in response to player actions to better fit the current player's skills and to avoid boredom or frustration. But is can also be used to create challenges that increase the performance of the player either through diversity or difficulty.

Metric	No DDA	DDA	Difference	Questions
Competence	2.95	2.6	-0.35	C2, C10, C15, C17, C21
Immersion	1.90	1.96	0.06	C3, C12, C18, C19, C27, C30
Flow	2.00	2.57	0.57	C5, C13, C25, C28, C31
Annoyance	0.56	0.98	0.42	C22, C24, C29
Challenge	0.96	1.84	0.88	C11, C23, C26, C32, C33
Negative Affect	0.97	0.87	-0.1	C7, C8, C9, C16
Positive Affect	2.71	2.87	0.26	C1, C4, C6, C14, C20
Positive Experience	1.67	2.02	0.35	P1, P5, P7, P8, P12, P16
Negative Experience	0.57	0.50	-0.07	P2, P4, P6, P11, P14, P15
Tiredness	0.47	0.70	0.23	P10, P13
Returning to Reality	0.76	0.93	0.17	P3, P9, P17

Figure 4: Average values for GEQ variables

Question	Result with DDA	Result without DDA
I think that I would like to use this system frequently.	2.33	2.53
I found the system unnecessarily complex.	3.33	3.53
I thought the system was easy to use.	3.27	3.47
I think that I would need the support of a technical person to be able to use this system.	3.40	3.93
I found the various functions in this system were well integrated.	2.93	2.73
I thought there was too much inconsistency in this system.	3.47	3.60
I would imagine that most people would learn to use this system very quickly.	3.40	3.33
I found the system very cumbersome to use.	3.53	3.53
I felt very confident using the system.	3.07	3.47
I needed to learn a lot of things before I could get going with this system.	3.60	3.66
Result	80.83	84.50

Figure 5: SUS Scores for the original version and the DDA implementation

This study is a first step in that direction. The DDA implementation used the FPS Half-Life as a case study, with two subsystems: in the first one, nine different types of difficulty modifying rules were created and selected based on the user success, each with multiple modifiers to ensure that multiple permutations were possible and increase the adaptability of the system; the second subsystem modifies item dispensing to guarantee that the player always has the items that he/she needs at a certain point.

The solution was tested by surveying a group of 15 volunteers for their overall experience and the usability of the system. The players' responses indicated that the system had a positive effect in stimulating player flow, immersion and positive emotion and that players felt challenged when playing.

REFERENCES

[1] R. Bartle, "Hearts, clubs, diamonds, spades: Players who suit MUDs," Journal of MUD research, vol. 1, no. 1, p. 19, 1996.
[2] C. Bateman and R. Boon, 21st Century Game Design, Charles River Media, 2006.
[3] L. Nacke, C. Bateman and R. Mandryk, "BrainHex: A Neurobiological Gamer Typology," Entertainment Computing, vol. 5, no. 1, pp. 55-62, 2014.
[4] N. Yee, "Gamer Motivation Model Overview and Descriptions," Quantic Foundry, 2015.
[5] N. Lazzaro, "Why We Play Games:Four Keys to More Emotion Without Story," Game Dev Conf, 2004.
[6] R. Batista, A. Coelho and C. Vaz de Carvalho, "Relationship between game categories and skills development: Contributions for serious game design," in Proceedings of the European Conference on Game Based Learning, 2015.
[7] M. Csikszentmihalyi, Flow: The Psychology of Optimal Experience, Harper, 1990.
[8] R. Hunicke and V. Chapman, "AI for dynamic difficulty adjustment in games," Challenges in game artificial intelligence AAAI workshop, vol. 2, 2004.
[9] C. Vaz de Carvalho, "Dynamic Serious Games Balancing," in International Conference on Serious Games Interaction and Simulation, 2015.
[10] A. Rietveld, S. Bakkes and D. Roijers, "Circuit-Adaptive Challenge Balancing in Racing Games," Conference Proceedings - 2014 IEEE Games, Media, Entertainment Conference, IEEE GEM 2014, 2014.
[11] Capcom, Resident Evil 4: The Official Strategy Guide, Future Press Verlag und Marketing GmbH, 2005.
[12] A. Gavin, "Making Crash Bandicoot – part 6," 2007. [Online]. Available: https://all-things-andy-gavin.com/2011/02/07/making-crash-bandicoot-part-6/. [Accessed 30 August 2021].
[13] N. Aghdaie, J. Kolen, M. M. Mattar, M. Sardari, S. Xue, K. Atif-Uz, K. Zaman and A. Moss, "Patent Application Publication: Dynamic Difficulty Adjustment," 25 March 2021. [Online]. Available: https://pdfaiw.uspto.gov/.aiw?PageNum=0&docid=20210086083&IDKey=34B510DB4F77&HomeUrl=http%3A%2F%

2Fappft.uspto.gov%2Fnetacgi%2Fnph-Parser%3FSect1%3DPTO1%2526Sect2%3DHITOFF%2526d%3DPG01%2526p%3D1%2526u%3D%2Fnetahtml%2FPTO%2Fsrchnum.html%2526r%3D1%2526f%3DG%. [Accessed 30 August 2021].

[14] P. Demasi and A. Cruz, "Online Coevolution for Action Games," in Proceedings of the 3rd International Conference on Intelligent Games and Simulation, 2002.

[15] P. Spronck, I. Sprinkhuizen-Kuyper and E. Postma, "Difficulty Scaling of Game AI," in Proceedings of the 5th International Conference on Intelligent Games and Simulation, 2004.

[16] G. N. Yannakakis and J. Hallam, "Evolving Opponents for Interesting Interactive Computer Games," in Proceedings of the 8th International Conference on the Simulation of Adaptive Behavior, 2004.

[17] G. Andrade, G. Ramalho, H. Santana and V. Corruble, "Challenge-Sensitive Action Selection: an Application to Game Balancing," in Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2005.

[18] GameAnalytics, "GameAnalytics," 2021. [Online]. Available: https://gameanalytics.com/. [Accessed 30 August 2021].

[19] J. Brooke, "SUS: A quick and dirty usability scale," Usability Eval. Ind., vol. 189, 1995.

[20] W. IJsselsteijn, Y. de Kort and K. Poels, The Game Experience Questionnaire, Eindhoven: Technische Universiteit Eindhoven, 2013.