How can we help? Towards a design framework for performance-accommodation mechanisms for users struggling with input

Rossau, Ingeborg Goll; Bugge Skammelsen , Rasmus ; Czapla, Jedrzej; Hougaard, Bastian Ilsø; Knoche, Hendrik; Jochumsen, Mads Rovsing

Published in:
CHI PLAY 2021 - Extended Abstracts of the 2021 Annual Symposium on Computer-Human Interaction in Play

DOI (link to publication from Publisher):
10.1145/3450337.3483497

Publication date:
2021

Document Version
Accepted author manuscript, peer reviewed version

How can we help? Towards a design framework for performance-accommodation mechanisms for users struggling with input

INGEBORG GOLL ROSSAU, Aalborg University, Denmark
RASMUS BUGGE SKAMMELSEN, Aalborg University, Denmark
JĘDRZEJ JACEK CZAPLA, Aalborg University, Denmark
BASTIAN ILSØ HOUGAARD, Aalborg University, Denmark
HENDRIK KNOCHE, Aalborg University, Denmark
MADS JOCHUMSEN, Aalborg University, Denmark

Maintaining balance between challenge and skills in games is critical for enjoyment, and can be applied to accommodate player performance as well as system performance for low accuracy input devices. Previous work has explored different performance-accommodation mechanisms (PAMs) for balancing, but studies have focused mainly on variables not directly related to how the PAMs lower challenge level. This paper identifies different levels of PAMs, then offers a first attempt at a coherent framework of action-level PAMs based on how they modify the mapping from player input to output.

CCS Concepts: • **Human-centered computing → HCI theory, concepts and models**.

Additional Key Words and Phrases: Video games; game design; game balancing; performance-accommodation mechanisms; player experience

ACM Reference Format:
Ingeborg Goll Rossau, Rasmus Bugge Skammelsen, Jędrzej Jacek Czapla, Bastian Ilsø Hougaard, Hendrik Knoche, and Mads Jochumsen. 2021. How can we help? Towards a design framework for performance-accommodation mechanisms for users struggling with input. 1, 1 (September 2021), 9 pages. https://doi.org/10.1145/3450337.3483497

1 INTRODUCTION

One of the main reasons players engage with video games is for the sake of enjoyment. When the challenge of a game matches the players' skill, they can enter a state of flow which makes the experience intrinsically rewarding and increases their engagement and immersion [9]. Flow majorly contributes to the enjoyment in video games [8, 22]. Games can keep players in a state of flow by applying mechanisms that adjust the level of challenge to the player's level of skill. Previous studies have referred to these mechanisms as game balancing [31], difficulty adjustment [1, 4], skill assistance [10], and skill-accommodation mechanisms [5]. In the literature, these mechanisms are often applied to provide an equal challenge in the presence of differing skills of opposing players in multiplayer games [4, 5, 7, 10, 14, 20, 31], but are also used to adjust challenge level in single-player games to keep players in flow [18, 37].

Authors' addresses: Ingeborg Goll Rossau, Aalborg University, , Aalborg, Denmark; Rasmus Bugge Skammelsen, Aalborg University, , Aalborg, Denmark; Jędrzej Jacek Czapla, Aalborg University, , Aalborg, Denmark; Bastian Ilsø Hougaard, Aalborg University, , Aalborg, Denmark; Hendrik Knoche, Aalborg University, , Aalborg, Denmark; Mads Jochumsen, Aalborg University, , Aalborg, Denmark.

2021. Manuscript submitted to ACM

The latter is perhaps especially applicable to players with impairments, which add an extra layer of challenge to the game [14]. Similarly, for novel input devices such as motion sensors (e.g. Microsoft's Kinect), brain-computer interfaces, etc., the difficulty is increased due to the large amount and range of sensor inputs and the required mappings to discrete or continuous outcomes in comparison to more constrained input methods such as joysticks, gamepads, keyboards, and mouse [36]. To address challenges from either poor sensor inputs or low ability of users to issue desirable inputs, we refer to this type of mechanism as 'Performance-Accommodation Mechanisms' (PAMs) defined as: *A game mechanism to increase the player's enjoyment by lowering the game's challenge level to accommodate for poor performance of the player, input device, or system.*

We focus only on mechanisms intended to decrease challenge rather than increase it, as these apply to all cases described above. For this concept we further exclude mechanisms benefiting players who perform well or extend the same benefits to all players in a multi-player context, as we consider these 'standard' game mechanics. An open question for designers is how to best reduce the challenge without reducing struggling players' gaming experience. Previous work investigating how to aid struggling players has explored different ways of implementing PAMs, but mainly focused on variables not directly related to challenge level, such as players' level of awareness and agency over the PAM [3–5, 10, 14, 31, 32]. While some studies explored different methods of lowering the challenge level [10, 31, 37], they did not articulate the difference between the PAMs.

Game balancing is a wide field with inconsistent terminology, hampering building a cohesive framework of approaches. This paper provides a first attempt at classification of PAMs based on a literature review. We identified two scopes of PAMs based on which level of game challenge they affect - the action-level (e.g. performing a jump), or high-level challenges (e.g. reaching the end of the level). Focusing on the action-level, we identified five different PAMs and formalized their differences in a model of input-output mapping.

2 PREVIOUS WORK

Baldwin et al. [3] presented a framework for classifying multiplayer dynamic difficulty adjustment (mDDA) from a review of 180 commercial multiplayer games. They identified seven dimensions to consider when designing PAMs: whether it is automatically applied by the system or user activated (automation), applied to an individual or a whole team (recipient), requiring the user to perform an action (user action), activated through an action requiring skill (skill dependency), single-/multi-use or used over time (duration), based on choices made before or during game play (determination), and how aware the assisted player and other players are (visibility). Studies on player experience of PAMs relied on similar variables, mainly focusing on *activation*, *awareness*, and *dynamism*.

Regarding *activation*, players preferred skill-dependent over skill-independent PAM activation [31] and user- over system-activated PAMs in two studies [31, 32] as system-activation reduced their autonomy [32]. However, system-activated PAMs did not reduce player enjoyment [31, 32]. Having *awareness* of PAMs reduced sense of autonomy of both assisted and unassisted players' [4]. While different PAMs balanced the gameplay between physically impaired players assisted and non-impaired players, both players experienced lower self-esteem when aware of help from PAMs [14]. In contrast, Depping et al. found no detrimental impact from awareness of PAMs to either assisted or unassisted players [10]. Regardless of their awareness, assisted players were likely to attribute successes to themselves, thus rendering adverse effects from disclosure not a concern. It is possible that the relatively subtle implementation of PAM in Depping et al.'s study (an indicator in the upper-left corner) caused it to not have the same negative effect as in Gerling et al's [14] study, where the PAM (reduction of incoming arrows for assisted playersin a DDR-style rhythm game) was much more obvious. *Dynamism* that automatically adjusts the amount of assistance players receive based on

their performance, often referred to as dynamic difficulty adjustment (DDA) [1, 3, 18], is common to implementations of PAMs. Static/non-dynamic PAMs can be found in multiplayer settings where less skilled player receive a pre-determined level of assistance [14, 31] similar to difficulty settings in commercial games, where an easier setting e.g. lowers enemies attack power or increases the player's health across the whole game. Player experience for both assisted and unassisted target assistance did not differ despite static accommodation allowing the assisted player to win more often, while adaptive accommodation resulted in more balanced outcomes [5].

While these dimensions are relevant to the design of PAMs, they do not describe how an instance of PAM affects the players' ability to complete challenges. There are studies which investigated players' experience with PAMs [5, 14, 37] which affected challenge level differently, but so far none have formalized the differences between the PAMs such that designers and researchers can leverage them in other contexts.

3 PERFORMANCE-ACCOMMODATION MECHANISMS

We can think of game challenges on different 'levels' in terms of what goal or outcome they contribute to. For example, in a platformer players may face the high-level challenge of completing a level or, on an even higher level, completing the game. On a moment-to-moment basis, players meet the per-action-level challenge of e.g. jumping across a gap, avoiding an enemy, etc.. The success of each action ultimately depends on a single input from a player. Succeeding in the action-level challenge will often bring players closer to completing their higher-level challenge.

Similarly, different types of PAMs can assist players on these levels. While all PAMs affect the challenge level, high-level PAMs do not directly affect the action-level challenge, instead manipulating continuous game parameters. This can e.g. be the number of enemies or characteristics such as spawn time, speed, etc. [28, 30]. It may also affect the game environment, such as changing the map layout in a platformer [34]. The amount of ammunition found in the environment in *The Last of Us* [24] is an example of this in a commercial games. It is inversely proportional to the amount of ammunition the player has, accommodating for players who are struggling to aim properly. In *Celeste's* [12] assist mode, one of the variables that can be manipulated is the game speed, which allows players who struggle with the fast-paced gameplay to slow down the movement of both the player avatar and the environment by as much as 50%. All these examples affect the overall challenge level but have at most an indirect impact on the action-level challenges. Whatever input the player makes (or doesn't make), the output of the action will be the same as the 'normal' output, i.e. without any assistance. While the assist mode in *Celeste* does mean movement (e.g. a dash or a jump) triggered by the player's input will be slower, it ultimately has the same result. It does not change whether the input results in a successful outcome, but instead allows players more time to figure out what the input should be in order to succeed. Action-level PAMs directly affect how input from the player is interpreted and translated into output. Thus this type of PAM has an effect on the difficulty of the action-level challenge. The following subsection presents a framework of action-level PAMs based on how the input-output mapping is manipulated.

3.1 Framework of action-level PAMs

We conducted a literature review based on keywords such as 'game balancing', 'difficulty adjustment', 'skill assistance' and 'video games'. Articles not describing concretely one or more action-level PAM(s) were excluded. This resulted in 13 articles describing 29 different cases of action-level PAMs, compiled in Table 1. We classified these into five distinct types of PAM, modelled in Figure 1 based on how they affect the mapping between input and output in terms of successes and failures in action-level challenges. To provide additional context to these PAMs, we provide examples of each in commercial video games, but did not attempt to systematically review that space.

Study	Augmented success	Mitigated failure	Rule change	Input override	Shared control
Baldwin et al. [4]		extra shield			
Bateman et al. [5]			sticky targets, cursor area (+)		target gravity
Cechanowicz [7]	speed (+), acceleration (+)				steering (virtual car)
Depping et al. [10]	outgoing damage (+)	incoming damage (-)	bullet magnetism		
Gerling et al. [14]	score (+) on hit		perfect score timing		
van Huysduynen [16]				autopilot (driving sim)	
Hunicke [18]		extra health when low			
Jensen, Grønbæk [20]	score (+) on hit		perfect score timing, target size (+)		
Mulder et al. [23]					steering (driving sim)
Rakita et al. [29]					steering (robotic arm)
Rogers et al. [31]	extra points for goals	no points for enemy		free points (no player input)	
Smeddinck et al. [4]		slower fall speed, smaller balance loss on hit			
Vicencio-Moreira [37]			area cursor (+), sticky targets, bullet magnetism	target lock	target gravity

Table 1. Overview of employed action-level PAMs, increases are marked (+), decreases (-)

Augmented successes create outcomes that can be considered better than what players could have achieved with successful inputs. In this way, it can be said that they expand the output space. This is often implemented as power ups or boosts in e.g. driving or shooting games. Depping et al. [10] implemented a damage modifier in a multiplayer shooting game, where the damage given and taken depended on the score difference between assisted and unassisted players; the bigger the score difference, the more damage the 'worse' player would deal. Their study showed that the PAM increased perceived competence, enjoyment, suspense, and more. Another example by Rogers et al. [31] explored how various PAMs affect the enjoyment and experience of a multiplayer augmented reality (AR) table football game. Their augmented success changed how many points a goal from assisted players would grant depending on which half of the goal it would hit, or which half of the field the ball was shot from. These examples show that augmented successes amplify or enhance successful inputs to have a bigger impact than normal (see Figure 1). In commercial games, we can find this PAM in e.g. the mushroom boosts in *Mario Kart 8* [26] . Other examples award permanent bonuses through lower difficulty settings, such as bonuses to attack power in games such as *Dragon Age: Origins* [6] or *Horizon Zero Dawn* [15] or to the effects of healing items in *Dishonored* [2].

Mitigated failures like augmented successes result in outputs often not attainable without the PAM, but are activated upon failed inputs. The outputs in this PAM lie between failures and successes, lowering the penalties of failures but not resulting in successes either. Rogers et al. [31] implemented mitigated failure in their AR table football game. When activated, the unassisted players would score a goal against the assisted players but would not get any points.

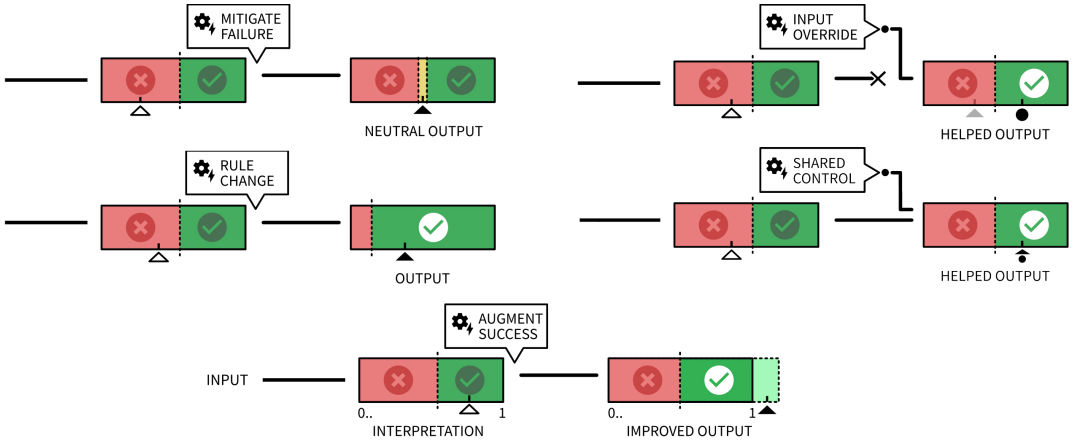

Fig. 1. Five action-level PAMs illustrated in terms of how they alter player outcomes: 1) Mitigated Failure, 2) Rule Change, 3) Input Override, 4) Shared Control and 5) Augmented Success. An input sends a signal to the system (black line), which makes an interpretation of the signal (white triangle). It then applies a PAM to its interpretation (white box), which alters the signal, creating a new interpretation (black triangle). In two cases, the signal is partially provided by the system itself (circular marker). The altered interpretation becomes the system's output.

It disregarded the assisted players' failure to stop the ball without rewarding them by e.g. giving them points. In a shooting game by Depping et al. [10] the assisted player took less damage when hit, making their failure less penalized than usual. Baldwin et al. [4] gave low-performing players in a shooting game a shield. When failing to avoid a hit, their failure was not punished by them losing health or dying. From these examples, we can see that when the output can be described as binary (e.g. stopping the ball from hitting the goal vs. not stopping it), the failure gets nullified (e.g. by removing the otherwise ensuing penalties). When the output is continuous (e.g. how much damage you take when getting hit), it can nullify or reduce the consequences of failure, e.g. no damage taken due to a shield or taking less damage. Either way, the output will fall in-between a normal success and failure, as seen in the model in Figure 1, modifying the output space just as augmented success. Similarly to augmented success, some commercial examples of mitigated failure are activated permanently via difficulty levels, such as lower impact of taking damage via defense bonuses (*Dragon Age: Origins* [6], *Kingdom Hearts 3* [33]). In *Hades* [35], the 'god mode' setting also increases resistance to incoming damage, and the resistance is further increased upon each successive death. In a dynamic multiplayer example, *Towerfall* [21] awards struggling players with a shield at the beginning of a round when they have fallen behind the leading player by a certain amount.

Rule changes apply to sub-optimal inputs, but translate them into successful outputs by changing the output space. They change the rules such that what would previously have been failures become successes, by changing the threshold for what can be considered a success. The output can therefore still be considered a result of the user's input. This PAM can be found in several contexts: Vicencio-Moreira et al. [37] implemented two aim assistance mechanisms relying on the user missing the target (a sub-optimal input), but changing the rules so it translates to a hit (a successful output). One is called 'area cursor' which increases the size of the area that is hit, and the other recalculates the bullet's travel vector to hit the target, called bullet magnetism. The user's initial input is still what triggers the output, but it no longer needs to be as close to the target in order to result in a hit. Another rule change PAM (sticky targets) lowers the CD gain when the cursor traverses targets. This increases the range of inputs that result in successful outcomes, i.e. lowering

the threshold for success. Gerling et al. [14] implemented a rule change in a multiplayer rhythm game, in which the timing for creating the optimal input was extended for assisted players, making otherwise sub-optimal inputs result in the same outcome as optimal input. For a continuous task, lowering the threshold is simply a matter of changing a range of what would otherwise be considered failures into successes, such as in the targeting assistance techniques described above. This is not an option for a binary task, unless the task has a timing aspect, such as in Gerling et al.'s dancing game. This can be seen in Figure 1, where an input is (under normal circumstances) interpreted as a failure, but the output is a success due to the lowered threshold for succeeding. Commercial games have used this PAM similarly to the examples implemented by Vicencio-Moreira et al., such as the aim assist in *Call of Duty: Warzone* [19] which uses the sticky targets mechanism. *Super Mario Odyssey* [27] and *The Last of Us 2* [25] contain similar versions of this PAM, both of which can be turned on via assist/accessibility modes, where the player cannot drown due to being underwater for too long. Here, the threshold for success (i.e. not drowning) gets lowered to a degree that renders failure impossible.

Input overrides replace failed inputs with completely system-generated success inputs. It is similar to rule change in that it turns a failure into a success, but rather than changing the output space to accommodate for the user's failure, it simply takes over and 'converts' their failure to a success. Unlike in rule change, the user has no control over the moment when it is employed, as illustrated in Figure 1. This PAM can therefore also be applied to instances where no input is provided, with the system providing the input instead. When input override is used, the outcome is identical to the outcome of the input it replicates. However, the feedback may be different. By this we mean that the end result will be the same even if how you get to that end result is not the same as you usually would. For example, Vicencio-Moreira et al. [37] implemented input override for a targeting task, where the system would instantly lock onto the nearest target at the press of a button. The outcome of this was the same as for normal targeting - placing the crosshairs over the target - but the movement of the crosshair was (most likely, depending on the user's skill level) both faster and more even than when performed by the user. Note that for a continuous input such as in a pointing/targeting task, this PAM essentially reduces the continuous output to a binary and a single 'correct' success output (i.e. moving the crosshairs in a direct line to the target). Similarly Van Huysduynen et al.'s self-driving car simulation replaced true negative inputs with system input, reducing the possible options of 'correct' steering to a single 'correct' path [16]. For some challenges the difference between input override and rule change can be subtle, and in some cases the two may look identical from a user's perspective. However, the two still differ in a fundamental way due to how one incorporates player input and one doesn't. Because of this, there is always temporal contiguity between player input and the output in rule change. For input override, as the player's input is discarded, output may trigger at a different time that can reduce player agency [17]. *Mario Kart 8* [26] employs this PAM in a similar fashion as Van Huysduynen et al. via the 'bullet bill' power-up, which when activated gives the system complete control over steering the player's car. *The Last of Us 2* [25] and *Shadow of the Tomb Raider* [11] both have auto-targeting options which allow the player to lock onto enemies with one press of a button, similar to the mechanism implemented by Vicencio-Moreira et al. In a binary example, a setting in *Kingdom Hearts 3* [33] allows players to turn 'auto-block' on, which automatically blocks enemy attacks without user input, though only when the player is not in the middle of an attack.

Shared control PAMs bear much resemblance to automation in driving cars. The Society of Automotive Engineers defined six levels of automation and human intervention and the attentiveness required in the driving task, ranging from fully manual (level 0) to fully automatic (level 5, the same as input override) [16]. Between these are various levels of split ownership between the driver and the system. This split ownership over control has been explained using the horse-metaphor (H-metaphor): a rider can use loose rein and provide only fine-tuning, otherwise letting their horse move autonomously, and then using tight rein to gain more control. The H-metaphor serves both as explanation and

narrative justification of how shared control works. Flemisch et al. [13] suggested that the concept of "*[t]ight rein and loose rein may be the extremes of a continuum rather than two exclusive states of operation.*" In the context of PAMs we refer to shared control when system input mixes with player input to lead to more desirable output than would have resulted from player input alone. In the literature, we found this PAM in the contexts of steering in driving games and in shooters for targeting. Bateman et al. [5] examined the effects of improving pointing through various target assistance schemes on player performance and experience. One of the target assistance schemes was target gravity, where the system pulled the reticle towards a target that the user was simultaneously moving towards, resulting in a mix of system and user input. In terms of fun, target gravity was deemed inferior to other target assistance schemes due to its interference with the users' movement, creating a conflict. This PAM can also be found in non-game contexts, e.g. in the novel system by Rakita et al. [29]. It used a shared control robotic arm to subtly improve the user's movements over time, showing and teaching them the movements to ideally perform a task. They compared the users' movements to a pre-defined ideal curve and created an average movement, similar to the mechanism implemented by Bateman et al. These cases all make use of continuous input and provide continuous output. Shared control cannot exist for a binary input/output system, as there is no 'room' for a blended input when there are only two options. The user also needs to perform sub-optimally, i.e. their input needs to fall within the range of 'failure', for the shared control to have an effect. If the user performed the movements perfectly in Rakita et al.'s experiment, then nothing would change from the input to the output. This is modelled in Figure 1, where we can see that shared control 'moves' a continuous failure input to a success input by combining system and user input. In contrast to input override, the output when shared control is applied is still influenced by the user's input. Just as in the literature, commercial games often apply shared control to steering tasks, such as the 'auto-steering' option that can be applied in *Mario Kart 8* [26]. This also works in the absence of any input and could then be considered Input Override instead, as the system takes complete control.

4 DISCUSSION AND CONCLUSION

We have presented a classification of PAMs into high- and action-level, both of which work to decrease challenge for struggling players. This paper makes no attempt at judging which PAMs result in a better player experience, as the literature does not provide any definitive answers. How players perceive a PAM can differ wildly, depending on how designs communicate it. For example, if implemented in a fast-paced shooting game, a rule change 'bullet magnetism' PAM ([10, 37] and a shared control 'target gravity' PAM ([5, 37]) may look the same to the player, even if the latter actively moves the player-controlled reticle and the former does not. Previous work also indicated that how a PAM is implemented with respect to dimensions such as activation, awareness and dynamism can affect player experience (see Section 2). In regards to flow [9], the design of the PAM could lead players to feel that their skill level has increased, rather than the challenge level having decreased. E.g. hidden PAMs could increase players' self-esteem, making them feel their skill is higher, while obvious PAMs can decrease it when players perceive them as the reduced challenges [14]. The narrative surrounding the PAM, i.e. how/whether it is 'explained' in-game, could also affect player perception. Augmented success that simply makes the damage amount increase (e.g. [6, 15]) could have a different effect from your car gaining a visible burst speed as flames shoot out of it (e.g. [26]). To our knowledge, no studies have explored how different PAM narratives affect player experience. The effect of PAMs on player experience may also vary depending on the game context and the skills and capabilities of the player.

Our framework describes action-level PAMs as separate, but many implementations include more than one at the same time, such as difficulty levels in games [6, 15, 33], in which damage output (augmented success), defense (mitigated failure) and effects of healing items (augmented success) were all manipulated concurrently. The implementations of

multiple PAMs simultaneously could hinder understanding their contribution to user experience, but it is also relevant to investigate how different PAMs may complement or contrast each other. Assisted players in the study by Gerling et al. [14] experienced increased self-esteem from both rule change and augmented success PAMs despite the fact that the PAMs did not lead to more wins (rule change) or cause landslide wins (augmented success). While a follow-up study combined the two PAMs and resulted in closer outcome their results did provide further insights. The differences between the two PAMs in the first study could also have been due to the way they were implemented in this specific study, as results indicated the amount of assistance provided was too high for augmented success and too low for rule change, and therefore not something that can be applied in general. Future work should address how PAMs interact with each other, both on an action- and high-level.

We identified a range of PAMs based on how they changed the mapping from input to output focusing on binary outcomes (success and failure). However, modelling PAMs based on different criteria might bear additional insights. Reviews of high-level PAMs and PAM implementations in commercial games would be equally relevant to expand the field of game balancing. While our descriptive model for action-level PAMs is fairly simple and might not be able to express all possible methods of assisting players, these design patterns provide a first stepping stone for a comprehensive framework that will assist the research community in comparing studies and building a consistent terminology. We hope it will be useful for game designers as a lens when choosing how to assist players.

ACKNOWLEDGMENTS

This research was partially funded by VELUX FONDEN (project number 22357).

REFERENCES

[1] Dennis Ang and Alex Mitchell. 2017. Comparing Effects of Dynamic Difficulty Adjustment Systems on Video Game Experience. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play (CHI PLAY '17)*. Association for Computing Machinery, Amsterdam, The Netherlands, 317–327. https://doi.org/10.1145/3116595.3116623

[2] Arkane Studios. 2012. Dishonored. Game [PC]. Bethesda Softworks, Rockville, Maryland, United States.

[3] Alexander Baldwin, Daniel Johnson, Peta Wyeth, and Penny Sweetser. 2013. A framework of dynamic difficulty adjustment in competitive multiplayer video games. In *2013 IEEE international games innovation conference (IGIC)*. IEEE, Chicago, IL, USA, 16–19. https://doi.org/10.1109/IGIC.2013.6659150

[4] Alexander Baldwin, Daniel Johnson, and Peta A. Wyeth. 2014. The effect of multiplayer dynamic difficulty adjustment on the player experience of video games. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems (CHI EA '14)*. Association for Computing Machinery, Toronto, Ontario, Canada, 1489–1494. https://doi.org/10.1145/2559206.2581285

[5] Scott Bateman, Regan L. Mandryk, Tadeusz Stach, and Carl Gutwin. 2011. Target assistance for subtly balancing competitive play. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. Association for Computing Machinery, Vancouver, BC, Canada, 2355–2364. https://doi.org/10.1145/1978942.1979287

[6] Bioware. 2009. Dragon Age: Origins. Game [PC]. Electronic Arts, Redwood City, California, United States.

[7] Jared E. Cechanowicz, Carl Gutwin, Scott Bateman, Regan Mandryk, and Ian Stavness. 2014. Improving player balancing in racing games. In *Proceedings of the first ACM SIGCHI annual symposium on Computer-human interaction in play (CHI PLAY '14)*. Association for Computing Machinery, Toronto, Ontario, Canada, 47–56. https://doi.org/10.1145/2658537.2658701

[8] Ben Cowley, Darryl Charles, Michaela Black, and Ray Hickey. 2008. Toward an understanding of flow in video games. *Computers in Entertainment (CIE)* 6, 2 (2008), 1–27. https://doi.org/10.1145/1371216.1371223

[9] Mihaly Csikszentmihalyi. 1990. Flow: The Psychology of Optimal Experience. Harper & Row New York, New York, NY, USA.

[10] Ansgar E. Depping, Regan L. Mandryk, Chengzhao Li, Carl Gutwin, and Rodrigo Vicencio-Moreira. 2016. How Disclosing Skill Assistance Affects Play Experience in a Multiplayer First-Person Shooter Game. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery, San Jose, California, USA, 3462–3472. https://doi.org/10.1145/2858036.2858156

[11] Eidos-Montréal. 2018. Shadow of the Tomb Raider. Game [PS4]. Square Enix, Tokyo, Japan.

[12] Extremely OK Games. 2018. Celeste. Game [Nintendo Switch]. Extremely OK Games, Vancouver, Canada.

[13] Frank O. Adams Flemisch. 2003. *The H-Metaphor as a Guideline for Vehicle Automation and Interaction.* Technical Report. NASA. https://ntrs.nasa.gov/search.jsp?R=20040031835

[14] Kathrin Maria Gerling, Matthew Miller, Regan L. Mandryk, Max Valentin Birk, and Jan David Smeddinck. 2014. Effects of balancing for physical abilities on player performance, experience and self-esteem in exergames. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. Association for Computing Machinery, Toronto, Ontario, Canada, 2201–2210. https://doi.org/10.1145/2556288.2556963

[15] Guerrilla Games. 2017. Horizon Zero Dawn. Game [PS4]. Sony Interactive Entertainment, San Mateo, California, United States.

[16] Hanneke Hooft van Huysduynen, Jacques Terken, and Berry Eggen. 2018. Why Disable the Autopilot?. In *Proceedings of the 10th International Conference on Automotive User Interfaces and Interactive Vehicular Applications* (Toronto, ON, Canada) *(AutomotiveUI '18)*. Association for Computing Machinery, New York, NY, USA, 247–257. https://doi.org/10.1145/3239060.3239063

[17] Bastian Ilsø Hougaard, Ingeborg Goll Rossau, Jedrzej Jacek Czapla, Mozes Adorjan Miko, Rasmus Bugge Skammelsen, Hendrik Knoche, and Mads Jochumsen. 2021. Who Willed It? Decreasing Frustration by Manipulating Perceived Control through Fabricated Input for Stroke Rehabilitation BCI Games. *Proceedings of the Annual Symposium on Computer-Human Interaction in Play* 5 (Sept. 2021), 235:1–235:19.

[18] Robin Hunicke. 2005. The Case for Dynamic Difficulty Adjustment in Games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology* (Valencia, Spain) *(ACE '05)*. Association for Computing Machinery, New York, NY, USA, 429–433. https://doi.org/10.1145/1178477.1178573

[19] Infinity Ward and Raven Software. 2020. Call of Duty: Warzone. Game [PC]. Activision, Santa Monica, California, United States.

[20] Mads Møller Jensen and Kaj Grønbæk. 2016. Design Strategies for Balancing Exertion Games: A Study of Three Approaches. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems* (Brisbane, QLD, Australia) *(DIS '16)*. Association for Computing Machinery, New York, NY, USA, 936–946. https://doi.org/10.1145/2901790.2901843

[21] Matt Makes Games. 2013. Towerfall. Game [Nintendo Switch]. Matt Makes Games, Vancouver, Canada.

[22] Lazaros Michailidis, Emili Balaguer-Ballester, and Xun He. 2018. Flow and immersion in video games: The aftermath of a conceptual challenge. *Frontiers in Psychology* 9 (2018), 1682. https://doi.org/10.3389/fpsyg.2018.01682

[23] Mark Mulder, David A. Abbink, and Erwin R. Boer. 2012. Sharing Control With Haptics: Seamless Driver Support From Manual to Automatic Control. *Human Factors* 54, 5 (Oct. 2012), 786–798. https://doi.org/10.1177/0018720812443984

[24] Naughty Dog. 2013. The Last of Us. Game [PS4]. Sony Interactive Entertainment, San Mateo, California, United States.

[25] Naughty Dog. 2020. The Last of Us 2. Game [PS4]. Sony Interactive Entertainment, San Mateo, California, United States.

[26] Nintendo EAD. 2017. Mario Kart 8 Deluxe. Game [Nintendo Switch]. Nintendo, Kyoto, Japan.

[27] Nintendo EPD. 2017. Super Mario Odyssey. Game [Nintendo Switch]. Nintendo, Kyoto, Japan.

[28] Johannes Pfau, Jan David Smeddinck, and Rainer Malaka. 2020. *Enemy Within: Long-Term Motivation Effects of Deep Player Behavior Models for Dynamic Difficulty Adjustment.* Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/3313831.3376423

[29] Daniel Rakita, Bilge Mutlu, Michael Gleicher, and Laura M. Hiatt. 2018. Shared Dynamic Curves: A Shared-Control Telemanipulation Method for Motor Task Training. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction (HRI '18)*. Association for Computing Machinery, Chicago, IL, USA, 23–31. https://doi.org/10.1145/3171221.3171278

[30] William Rao Fernandes and Guillaume Levieux. 2019. *delta-logit : Dynamic Difficulty Adjustment Using Few Data Points.* Springer International Publishing, Cham, 158–171. https://doi.org/10.1007/978-3-030-34644-7_13

[31] Katja Rogers, Mark Colley, David Lehr, Julian Frommel, Marcel Walch, Lennart E. Nacke, and Michael Weber. 2018. KickAR: Exploring Game Balancing Through Boosts and Handicaps in Augmented Reality Table Football. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, Montreal QC, Canada, 1–12. https://doi.org/10.1145/3173574.3173740

[32] Jan D. Smeddinck, Regan L. Mandryk, Max V. Birk, Kathrin M. Gerling, Dietrich Barsilowski, and Rainer Malaka. 2016. How to Present Game Difficulty Choices? Exploring the Impact on Player Experience. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery, San Jose, California, USA, 5595–5607. https://doi.org/10.1145/2858036.2858574

[33] Square Enix. 2019. Kingdom Hearts 3. Game [PS4]. Square Enix, Tokyo, Japan.

[34] David Stammer, Tobias Günther, and Mike Preuss. 2015. Player-adaptive Spelunky level generation. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, Chicago, IL, USA, 130–137. https://doi.org/10.1109/CIG.2015.7317948

[35] Supergiant Games. 2020. Hades. Game [Nintendo Switch]. Supergiant Games, San Francisco, California, United States.

[36] Bram van de Laar, Danny Plass-Oude Bos, Boris Reuderink, Mannes Poel, and Anton Nijholt. 2013. How Much Control Is Enough? Influence of Unreliable Input on User Experience. *IEEE Transactions on Cybernetics* 43, 6 (Dec. 2013), 1584–1592. https://doi.org/10.1109/TCYB.2013.2282279

[37] Rodrigo Vicencio-Moreira, Regan Mandryk, Carl Gutwin, and Scott Bateman. 2014. The effectiveness (or lack thereof) of aim-assist techniques in first-person shooter games. *Conference on Human Factors in Computing Systems - Proceedings* (April 2014). https://doi.org/10.1145/2556288.2557308