Dungeon Wipe: Exploring Dynamic Difficulty Adjustment with Power-Up Mechanics

Tiago Félix[1], Fausto Mourato[1,2] [0000-0003-4935-7206] and João Morais[1] [0009-0006-5067-7185]

[1] ESTSetúbal, Inst. Politécnico Setúbal. Campus IPS, Estefanilha, 2914-508, Setúbal, Portugal
[2] Centro ALGORITMI, University of Minho, Campus Azurém, 4800-058, Guimarães, Portugal
202100209@estudantes.ips.pt;
{fausto.mourato, joao.morais}@estsetubal.ips.pt

Abstract. Dynamic Difficulty Adjustment (DDA) is a mechanism in video games that automatically adapts the game's difficulty based on the player's skills or performance, aiming to provide a tailored and engaging experience for players of varying abilities. However, such approaches have drawbacks, including feelings of unfairness and potential integrity compromise of the gaming experience. In this paper, we present an action game prototype to test an approach that aims to mitigate these drawbacks by applying DDA indirectly to the gameplay and core game content. Our method consists of keeping the core gameplay and challenges intact, such as the number of enemies and their features, while adjusting the distribution of power-ups during the level. The idea is to distribute power-ups, like healing potions, to guide the gaming experience through a state of flow, providing moments of alternate tension/relaxation to promote a more immersive experience and affective learning steps. Our initial experiments show that this approach provides some control over the player's life value through gameplay without obvious interference, suggesting further research.

Keywords: Power-ups, DDA, Player Experience, Flow.

1 Introduction

Game development requires balance between entertainment and engagement. Central to this challenge is captivating players to immerse them in a state of optimal engagement, commonly referred to as flow [1], a concept adapted from psychology to video games as game flow [2]. Flow is characterized by a balance between the challenge and the player's skills, leading to a deeply focused engagement where time seems to slip away. Flow implicitly implies the concept of difficulty, which can be linked to the subjective concept of player's satisfaction or to more concrete measurements based on probabilities of success and failure [3, 4].

Understanding difficulty in the game design process spans different motivations. Whereas it is important from the definition of the experience itself, considering the notion of flow, it can be applied differently. One context in which it is important is Procedural Content Generation (PCG), the creation of any type of content that is part of a game using programmatic instructions in opposition to use tools for the purpose.

It is thus “the algorithmic creation of game content with limited or indirect user input” [5], which can be a difficult task because “the generator has to create the content, satisfy constraints imposed by the artist, and return interesting instances for gamers” [6]. This includes providing an output with an adequate challenge in which “a minor change can affect [the level] with indefinite proportions” [7]. Game wise, the roots of PCG can be found in the game Rogue, an influential title that origins the term rogue-like for games that include levels with rooms and corridors established with random factors, providing replayability through an accountable set of possible levels [8].

Besides PCG, an interesting research topic that encompasses the concept of difficulty in entitled Dynamic Difficulty Adjustment (DDA). The concept can be traced back to the work of Hunicke [9] with the purpose of “modulating in-game systems to respond to a particular player’s abilities over the course of a game session”.

We address DDA in action games, presenting a rogue-like dungeon-crawler that applies DDA with a specific approach to control difficulty indirectly, without changing the main elements (number of opponents, damage factors, etc.) and thus focusing complementary elements, mainly power-ups (healing potions, power boosts, etc.).

2 Related Work

Regarding game design, numerous approaches have been proposed to conceptualize the notion of difficulty, from quantitative metrics to qualitative assessments, ranging from heuristic models to complex algorithms. Setting up a level implies applying patterns in which difficulty is involved, from defining set back penalties and establishing rules to force players to lose [10]. Tangible metrics and assessments require quantitative approaches as difficulty must be measurable and comparable [3, 11].

Moreover, different genres require different approaches. In racing games, one can aim to create fast paced segments of high speed in contrast with sharp turns on specific locations, while considering aesthetic notions [12, 13]. Multiple strategy games focus on leveraging game progress [14]. Platformers focus the level physical features to depict challenges [15] but can also consider the player’s affective state [16].

Even though there are multiple positive aspects to point regarding DDA, there are also some drawbacks that can be pointed and that are sustained in literature, namely:

- **Player frustration**, especially if the adjustments are not finely tuned or if they fluctuate too frequently, such as the “rubber-band” effect in racing games [9].
- **Misuse of the adjustment mechanism**, in which the players notice that rules change somehow depending on their performance, and then feel tempted to exploit those changes in their favor. The example of the previous point is also applicable.
- **Impact on skill development**, as adapting to the player's abilities can prevent them from overcoming obstacles that are essential for skill improvement.
- **Reduced immersion**, because abrupt changes in difficulty levels can break the flow and remind players that they are in a constructed environment [17].

Our work focuses those drawbacks, trying to promote a way to keep the original challenge, but setting an adequate timing to intervene, as we will see in the next section.

3 Indirect DDA Adjustments

3.1 Approach Foundations

The main premise of our approach for indirect adjustments is straightforward: for players struggling to accomplish to overcome a level, power-ups should be placed in advance, preventing premature failure, and adding an additional buffer to the learning curve. For players exceeding the amount of challenge, power-ups should come when they are less useful (for instance, when the players' life is maxed out), slightly increasing difficulty without removing their empowerment. Still, the number of power-ups that are spawned for a level should be the same regardless of the players skills.

For this preliminary study on the subject, we are focusing the difficulty adjustments on applying those principles with healing potions, a life regeneration mechanism that in normally present in action videogames [18], even though a more structured and complex approach can have the same foundations with other power-ups.

3.2 Application Setup

To test our approach, we are in the development process of an action rogue-like dungeon-crawler entitled Dungeon Wipe, which will be described with more detail in Section 4. In Dungeon Wipe, by design, enemies' appearance in the level are predetermined timed events, meaning that enemy spawners have a deterministic behavior. Under these conditions, a level represents implicitly a function that depicts the damage that the player will sustain during the level. This function has higher values for sections of higher intensity and lower values for section of calmer gameplay.

Such function can be established in different manners: recording damage logs for one specific game session, a set of playing sessions (for a certain player or group of players) or play sessions of virtual AI players. Having a damage function allows predicting the evolution of the player's life value on the game. Fig. 1 shows an example of such function, considering one single session of a certain player. It is possible to observe that the level starts with an intense short period, followed by an also short period of rest. This is followed by a more intense confrontation period that provides another resting stage before the final wave. Fig. 2 shows the evolution of player's life during that session considering different cases, each one considering a different approach to provide health recovery items to the player. In case 1 the healing potions are spawned constantly, at fixed intervals. It is possible to observe that this leaves the player constantly in average values, avoiding more intensive emotions. Case 2 has a similar approach but uses two healing items, one restoring a smaller life value and the other restoring a higher life value. During the level, the spawn mechanism alternates between spawning small and large healing positions. This approach provides a higher oscillation of life values because the consecutive items tend to apply a two-stage effect: death avoidance followed by empowerment. Still, this does not guide the player through more intensive emotions. In opposition to cases 1 and 2, in case 3 the spawning mechanism reacts to the player performance, instead of being predetermined. The approach consists of allowing the players to reach lower life values and then reacting by providing enough items for them to reach an empowered state of high life value.

This is the behavior to which we aimed and is the goal of the proposed mechanism that we will cover next.

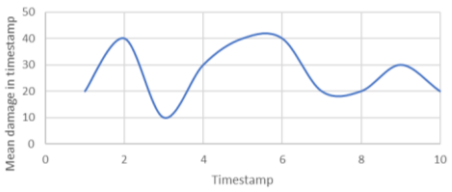

Fig. 1 – Example damage function

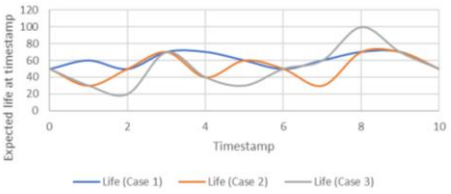

Fig. 2 – Impact of health items distribution

3.3 Proposed mechanism

The proposed mechanism is described considering that a level as a predetermined value to be given as regeneration to the player. This value can be set by the designer for manually created levels or defined by the PCG algorithm. The whole generation should be divided into healing potions that will be spawned in the level. For this process, the following elements should be considered:

- T: Total healing value that will be provided with healing points.
- $H = \{h_1\ h_2, \ldots, h_n\}$: Set of possible healing potions (our prototype has two types of potions, small (h_{30}) and large (h_{60}), to recover 30 or 60 points of life.
- t_i: Timestamps that are defined from t_1 to t_n.
- A: Anxiety threshold (a life value under A means the player is in anxiety).
- E: Empowerment threshold (a life value over E means the player is empowered).

The algorithm works as follows:

- Calculate expected life value at that timestamp, l_i.
- If $l_i<A$ (player is in the anxiety zone):
  - Determine the required healing value, $h_r=A-l_i$.
  - Spawn the smallest healing potion $h_j \in H$ such that $h_j \geq h_r$.
- Update the player's expected life value: $l_i \leftarrow l_i+h_j$.
- Continue spawning healing potions for subsequent timestamps until l_i exceeds the empowerment threshold E.

This approach waits for the player to struggle and then provides recovery stages.

4 Implemented Prototype and Test Framework

4.1 Dungeon Wipe

To apply and test the metrics proposed in the previous section, we have been implementing Dungeon Wipe, a dungeon-crawler game, applying the aforementioned principles. In Fig. 3 we present screenshots of the game, showcasing the main gameplay on the left and a healing potion on the right.

Fig. 3 – Screenshots of Dungeon Wipe (gameplay on the left, health potion on the right).

The game is being developed with the Unity engine and following the principles of Doppler Interactive GDLC [19]. Now it is in the iterative cycling between production and testing aiming for an alpha version, considering different approaches for DDA while establishing the final game mechanics of the game. The initial design followed the MDA principles [20], aiming for challenge aesthetics driven through PCG for replayability. Additionally, the game aims for players under the profile achievers (according to Bartle's taxonomy [21]), setting challenges as completing levels under a certain time and gathering as much treasures as possible within the level duration.

The current version of the game uses levels that have been created but are read from external JSON files that can be either stored on the hard drive or on a web server. Having level externally allows a straightforward use of PCG, as the service that provides the files can obtain them from a generation algorithm.

4.2 Preliminary Results

Playtesting sessions have been used to analyze gameplay, user experience and, specifically, players' perception regarding DDA with the proposed approach. Some experiments are still under development, but we have already run some tests having the players to play levels with and without potion distribution adjustments.

Two main hypotheses have been tested:

- H_1: Indirect DDA is not evident to the players.
- H_2: The indirect DDA approach still allows controlling difficulty.

To test H_1, a set of players from a convenience sample comprised by 38 students from two different computer science degrees were asked to play the game having access to an unspecified option that they could toggle. This option consisted in enabling or disabling DDA. They were faced with the challenge of identifying what the options does. 83% of them were not able to perceive changes. The remaining players understood that the option controlled somehow the appearance of potions, even though no one identified exactly how. After their individual sessions, the players could debate, and it was interesting to observe some disagreements. Some claimed that the option would make more potions to appear, while others claimed it was the opposite. Naturally, this was caused by their performance.

H_2 was tested theoretically using an approach based on the Monte Carlo method. We ran multiple simulations, sampling possible evolution of players life considering different player profiles and different health potion approaches (fixed intervals, ran-

dom intervals and with the indirect DDA method) and counting the probability of success/failure for the whole level. Fig. 4 shows the defined damage function for a test level and the random damage values that were obtained for a single run. Fig. 5 shows the evolution of the player's life distributing the healing potions uniformly, randomly and with our algorithm. After testing with different damage functions (5 different levels were considered) and applying 1000 runs for each one, we verified that our algorithm provides control over difficulty, in fact making the levels much easier. To provide de overall picture, for our model of standard player, the probability of failing the level (average of the 5 levels) was 40%, 56% and 8%, using uniformly distributed, randomly distributed, and distributed with our algorithm, respectively. This shows us that H_2 is plausible theoretically. We consider that at this moment it is important to compare these values with effective probabilities of success measured in the game, as well as perceiving if the indirect change of the probabilities have a positive impact in the player experience.

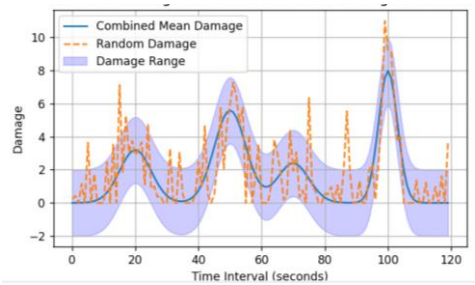

Fig. 4 – Example of randomized damage

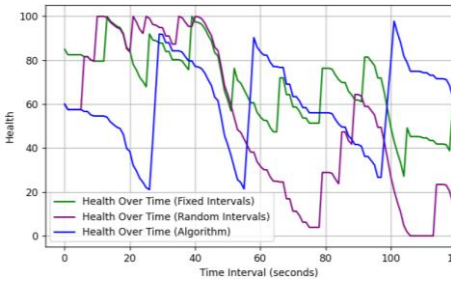

Fig. 5 – Example of a test run

5 Conclusions and Further Developments

We have presented an approach for indirect DDA using as a test case a game prototype. By adjusting difficulty indirectly through the strategic placement of healing potions, we aimed to enhance player experience without changing the core challenge. This preliminary work shows potential, as it allows changing the probability of success and failure without having the players to evidently notice the changes.

Our algorithm controls how health potions are spawned in the levels, but the proposed approach can be used with other types of power-ups, such as movement boosts, armor, damage multipliers, game modifiers, and weapons, among others. All those types of power-ups are more indirect to the notion of life, but still imply on the probability of success in the level. Further research will focus on understanding these differences and how the parametrization of such power-ups can also affect difficulty.

Still, there is room for further research regarding the distribution of power-ups. Our algorithm uses an overall value to be distributed within a level considering power-ups that are spawned at certain timestamps. We also aim to research the alternative of adjusting the probability of dropping power-ups when killing enemies, which might depend on the current level time, current live value, and the probability deviation.

References

1. Csikszentmihalyi M (1991) Flow: The Psychology of Optimal Experience. Harper & Row
2. Chen J (2007) Flow in Games (and Everything Else). Commun ACM 50:31. https://doi.org/10.1145/1232743.1232769
3. Aponte M-V, Levieux G, Natkin S (2011) Difficulty in Videogames: An Experimental Validation of a Formal Definition. In: Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology. ACM, New York, NY, USA, pp 49:1--49:8
4. Mourato F, Birra F, Dos Santos MP (2014) Difficulty in action based challenges: Success prediction, players' strategies and profiling. In: ACM International Conference Proceeding Series
5. Togelius J, Kastbjerg E, Schedl D, Yannakakis GN (2011) What is Procedural Content Generation?: Mario on the Borderline. In: Proceedings of the 2Nd International Workshop on Procedural Content Generation in Games. ACM, New York, NY, USA, pp 3:1--3:6
6. Hendrikx M, Meijer S, Van Der Velden J, Iosup A (2013) Procedural Content Generation for Games. ACM Transactions on Multimedia Computing, Communications, and Applications 9:1–22. https://doi.org/10.1145/2422956.2422957
7. Mourato F, Próspero dos Santos M (2010) Measuring Difficulty in Platform Videogames. In: 4ª Conferência Nacional Interacção humano-computador. Aveiro, Portugal, pp 173–180
8. de Castro BP, da Mota RR, Fantini EPC (2017) Level Design on Rogue-like Games: An Analysis of Crypt of the Necrodancer and Shattered Planet
9. Hunicke R (2005) The Case for Dynamic Difficulty Adjustment in Games. In: ACE '05 Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology. pp 429–433
10. Juul J, Norton M (2009) Easy to Use and Incredibly Difficult: On the Mythical Border Between Interface and Gameplay. In: Proceedings of the 4th International Conference on Foundations of Digital Games. ACM, New York, NY, USA, pp 107–112
11. Aponte M-V, Levieux G, Natkin S (2011) Measuring the Level of Difficulty in Single Player Video Games. Entertainment Computing 2:205–213. https://doi.org/10.1016/j.entcom.2011.04.001
12. Togelius J, De Nardi R, Lucas S (2006) Making Racing Fun through Player Modeling and Track Evolution. In: Proceedings of the SAB'06 Workshop on Adaptive Approaches for Optimizing Player Satisfaction in Computer and Physical Games
13. Togelius J, De Nardi R, Lucas SM (2007) Towards Automatic Personalised Content Creation for Racing Games. 2007 IEEE Symposium on Computational Intelligence and Games 252–259. https://doi.org/10.1109/CIG.2007.368106
14. Pereira G, Santos P, Prada R (2009) Self-adapting Dynamically Denerated Maps for Turn-based Strategic Multiplayer Browser Games. In: ACE '09 Proceedings of the International Conference on Advances in Computer Enterntainment Technology. pp 353–356