

# Procedural Progression Model for Smash Time

João Catarino  
INESC-ID & Instituto Superior Técnico  
University of Lisbon, Portugal  
jcbpc@tecnico.pt

Carlos Martinho  
INESC-ID & Instituto Superior Técnico  
University of Lisbon, Portugal  
carlos.martinho@tecnico.pt

**Abstract**—This paper addresses the problem of improving the player experience in single player endless games and encouraging the player to be engaged with procedurally generated content for longer periods of time. We present a model in which the procedural content generation process takes into consideration the dynamics of two dimensions of the player experience: the performance of the player when overcoming the challenges created by the game, and the variety of challenges presented to the player over time. We discuss the implementation of the model in the endless mode of the mobile game *Smash Time*, and describe how its evaluation supports that the model was able to increase both the number and duration of play sessions as well as having a better game experience reported by the participants, when compared to the original game. These results suggest that this approach could improve replayability and, as a consequence, the lifetime of a digital game.

**Index Terms**—Player Skill, Content Variety, Progression Modelling, Procedural Content Generation, Dynamic Difficulty Adjustment, Player Modelling, Digital Games.

## I. INTRODUCTION

Most endless single player games generate content based on a difficulty setting whose parameters are tuned based on pre-launch playtesting or post-launch on-going analytics and such parameters are typically the same for all the players playing the game at a certain point in time. To improve each play experience, we should provide means for the game to adapt to the individual characteristics of each player, and keep them in Csikszentmihalyi’s flow channel [1] at all time, ensuring that the players never get bored because they have to perform a task well below their skill, or get anxious as they have to deal with a task too difficult for their skill level.

To address this issue, we propose a progression model that procedurally creates content not only based on the inherent difficulty of each challenge but follows a player-centric approach that takes both the player’s skill as well as the player’s familiarity with each type of challenge into account. Overall, this is achieved by tagging the challenges when they are created and tracking player performance while interacting with the challenges associated with these tags. By choosing challenges according to the desired skill and variety progression curves associated with their associated tags, we hypothesize this will create a more engaging play experience and promote more frequent and longer play sessions with the game. To validate our approach, we integrated our model in the endless mode of the mobile game *Smash Time*, a smasher game with more than

250K downloads on the iOS, Android and Windows Phone platforms.

## II. RELATED WORK

Procedural Content Generation (PCG) is the process in which computer software algorithmically generates, on the fly, game content with limited or indirect user input [2] [3]. The algorithmically generated game content can be anything from dungeons and levels [4], game rules [5], 2D textures and 3D models, characters and items, vegetation [6], weapons [7], music, side quests and story [8], etc. Focusing on the player experience to modulate the generation of effective and meaningful content, Yannakakis and Togelius [9] proposed a framework for PCG driven by computational models of user experience based on the personalization of user experience through affective and cognitive modelling combined with real time adjustment of the content according to user needs and preferences.

The player is the main element when we talk about the experience created by a game. In order to develop good games with mechanisms that boost the experience, we need to have a good understanding of the player: motivations such as needs, preferences, interests, expectations, values, fears and dreams; limitations; capabilities; knowledge; and the context in which they play a game, e.g. with whom, where and when they play a game. Gathering this information, we are able to create a player profile that will be useful to create better game experiences [12]. As Chen [13] points out, each player is different and experiences the same games in different ways, due to their personality, skills and expectations when playing a game. To satisfy different types of players, the game should be able to adapt itself to the preferences of the player.

Cook [14] describes the player as an “(...) entity that is driven, consciously or subconsciously, to learn new skills high in perceived value”. In this context, a skill is a behavior that a person uses to manipulate the world. Cook states that, when players learn something new and can use that knowledge to successfully manipulate the environment for the better, they experience joy and gain pleasure for that achievement. Furthermore, to create enjoyable play experiences, the game should demand full concentration from the player because when a person needs most of his/her skills to deal with a challenging situation, his/her attention is completely absorbed by the activity in question leaving no excess attention and focus to process anything else besides that activity [1].

Shaker et. al [15] [16] demonstrated how to collect players' data to accurately model player experience and tailor game content generation according to the player behavior. They propose the use of Active Learning for player experience modelling. With this technique, the learning algorithm is allowed to select the data to learn from, significantly reducing the amount of data needed for training the player model. A data set from hundreds of players of Infinite Mario Bros<sup>1</sup>, related to content features, gameplay features and reported player experience, was used to learn models of player experience through an active learning approach. Even though our work is focused in modeling the player's skill progression and the goal of this work is to create models of player's experience, using concepts like challenge and frustration, both use online content generation with the data being collected as needed in real time, with the goal of providing a better gameplay experience to the players. Bakkes et. al [10] and Blom et. al [11] also used the Infinite Mario Bros video game to personalize the play space to tailor the experienced challenge and the affective game experience of the individual user respectively. Both models focus on player experience while our proposed model focus on player skill.

Pereira [17] developed a progression modelling tool for an endless runner game allowing a game designer to specify the conditions that will unlock new challenges and game mechanics according to the player's mastery over other challenges and mechanics in the game. Skill mastery is represented as a set of categories (e.g. uninitiated, partially mastered, mastered, among others) inspired by the work of Cook, and tracked while the player uses the mechanics to overcome the different challenges procedurally generated by the game. The progression is guided by preset adaptation rules created by the game designer as a progression graph that specifies how challenges and mechanics should be considered in the PCG process according to the player's skill evolution.

Bicho and Martinho [18] proposed a progression model that takes player skill progression as its core feature. Players have a set of actions, called mechanics, available to overcome the presented challenges, and player performance is defined in terms of mechanics and associated challenges. This means one player might have different success rates overcoming the same challenge while using different mechanics. According to player choices on which mechanics to use when facing a challenge, the model will record skill progression using pairs  $\langle challenge, mechanics \rangle$ . Every time a challenge is presented to the player, the model saves if the player succeeded or failed at the challenge with the chosen mechanics. By measuring the player performance on these dimensions, it is possible to increase game difficulty on the dimensions the player is better at, while at the same time pushing the players to improve their skill on dimensions less explored. Bicho and Martinho show that players tend to stick to certain approaches

<sup>1</sup>Markus Persson, "Infinite Mario Bros." Super Mario Bros open-source clone, 2008. Available at <https://openhtml5games.github.io/games-mirror/dist/mariohtml5/main.html>

when overcoming a challenge even when using another would make the challenge much easier.

Zook et al. [19] proposed a model for skill-based mission generation that tries to solve challenge tailoring, i.e. "the problem of matching the difficulty of skill-based challenges over the course of a game to match player abilities"; and challenge contextualization, i.e. the fact that the game should provide appropriate motivating story context for the skill-based challenges. To deal with the challenge tailoring problem, the model must find a sequence of challenges that produce a given progression of predicted player performance. To achieve this, the game designer specifies a performance curve that determines the wanted progression of the player's performance over the course of a mission.

In this work, we will consider each player individually, with their own capabilities and limitations, and aim at providing a distinct and personal play experience in each run. This will be achieved through the creation and maintenance of a player model that will keep track of how the player is gaining mastery over the different dimensions of play. Rather than using a dependency graph preset by the game designer, which may become difficult to maintain as the amount of content increases throughout development, all content will be unlocked from the start and categorized using tags. The selection of the content available by the PCG process at a certain point in the game will be based on how the player mastery over these categories of play identified by the tags match the desired performance at this point in the game, as specified by the game designer through performance curves. This model will be updated over the course of a game as well as between game sessions. With this model, we expect each run to have a distinct feel while being enjoyable for the player going back through the game.

### III. SMASH TIME

Smash Time (see Fig. 1) has fast gameplay mechanics, that result from the combination of elements from classic games like Whac-A-Mole<sup>2</sup> and Space Invaders<sup>3</sup>, mixed with puzzle mechanics. Smash Time characters are enemies, animals and heroes, that coexist in the same world. Enemies enter the screen from the top and both sides and try to attack both the hero that is at the bottom of the screen, helping the player, as well as the animals that are trying to escape. The player goal is to smash the enemies and clear all the incoming waves. To smash one enemy and receive points for it, the player must tap on the enemy with a finger.

The progression model developed in this work was implemented using the Unity game engine on top of the Arena game

<sup>2</sup>The Whac-A-Mole arcade game (Creative Engineering Inc, 1976) consists of a large cabinet with five holes in its top and a large black mallet. Once the game starts, the moles begin to pop up from their holes at random. The goal of the game is to force the moles back into their holes by hitting them on the head with the mallet. The more quickly this is done the higher the final score will be.

<sup>3</sup>Space Invaders (Taito, 1978) is a shooter in which the player controls a laser cannon by moving it horizontally across the bottom of the screen and firing at descending aliens. As more aliens are defeated, their movement speeds up. Defeating the aliens brings another wave that is more difficult, a loop which can continue without end.



Fig. 1. Snapshot from Smash Time, Arena mode (endless game mode)

mode of Smash Time. In the Arena mode, players try to play for as long as possible and reach their highest score possible, on an infinite level with a timer. According to the player's performance at killing enemies in special sequences, a special enemy may appear that will extend the timer if smashed.

#### IV. PROGRESSION MODEL

Our approach is anchored on three main concepts – Challenges, Obstacles, and Tags – used by the three main components of the progression model: the Player Performance Model, the Content Variety Model and the Content Generation Model.

##### A. Challenges

Challenges are formations created by the game designer with one or more paths (wave paths) that guide the movement of the obstacles that will compose the challenge. The game designer creates a library of challenges that are stored as Unity prefabs (game objects with components and properties that can be used as templates to create new object instances in real time) to be used by the progression model during gameplay. A challenge is composed by a set of waves that spawns a set of obstacles that will move with a certain speed (challenge pace) in a specific wave path. A wave path is defined by a set of waypoints (intermediate points that are connected to form the path that will be followed by the obstacles of the wave) that are specified by the game designer. The quantity and type of each wave's obstacles, as well as the challenge pace are defined by the progression model in real time.

##### B. Obstacles

An obstacle is something that requires players to use their skill to overcome. In Smash Time Arena, the obstacles are enemies: Red, Green, Blue and Purple. When the player taps on an enemy, it smashes it and removes it from the game,

but may spawn another enemy as a result of being smashed. Fig. 2 illustrates this process. We consider that an obstacle is completely overcome when the Purple and last enemy in the chain is smashed and removed from the game.









Obstacle	Mechanic	Obstacle	Mechanic	Obstacle	Mechanic	Obstacle	Mechanic	Obstacle
Red Enemy	→ tap	Green Enemy	→ tap	Blue Enemy	→ tap	Purple Enemy	→ tap	✓
								✓

Fig. 2. Overcoming obstacles in Smash Time Arena: when Red is smashed, it spawns Green before being removed from the game. Similarly Green will spawn Blue that will spawn Purple.

##### C. Tags

Our progression model uses a set of tags to categorize the challenges and obstacles of the game. The different tags are freely created by the game designer and inform the progression model to keep track of these dimensions as they will be important for the modulation of the game experience. When a player interacts with content in the game, each tag associated with the content will be updated to reflect how successful the player has been at overcoming content with this tag and how often the player has encountered content with such tag. In Smash Time Arena, tags are themselves organized in four groups: Obstacle (e.g. *Red*, *Green*), Pace (e.g. *Moderate* or *Fast*), Taps (e.g., *Taps1-3*, *Taps4-6*) and Game Designer tags (e.g. *TargetHero*, *Gate*).

Fig. 1 depicts an example of a procedurally generated challenge. This challenge is composed by 6 different initial obstacles, 4 purple enemies and 2 blue enemies, moving at a slow pace from the top of the screen towards the hero at the bottom. The tags of this challenge are: Obstacle = {*Purple*, *Blue*}; Pace = {*Slow*}; Taps = {*Taps7-9*}; and Game Designer = {*TargetHero*, *DownLine*}.

##### D. Progression Model Overview

Fig. 3 depicts the architecture of the progression model and how it is integrated into the game loop of Smash Time Arena, which repeats the following steps:

- 1) Generate a challenge to present to the player, based on:
  - the Challenge Library;
  - the prediction of player performance based on the Player Performance Model;
  - how recently the player interacted with content based on the Content Variety Model.
- 2) Register the player response when dealing with the obstacles that compose the generated challenge;
- 3) Analyze the player performance through the recorded player actions relative to the generated challenge;
- 4) Record the player performance data in the Player Performance Model;
- 5) Record the challenge variety data in the Content Variety Model;
- 6) Repeat from step 1.

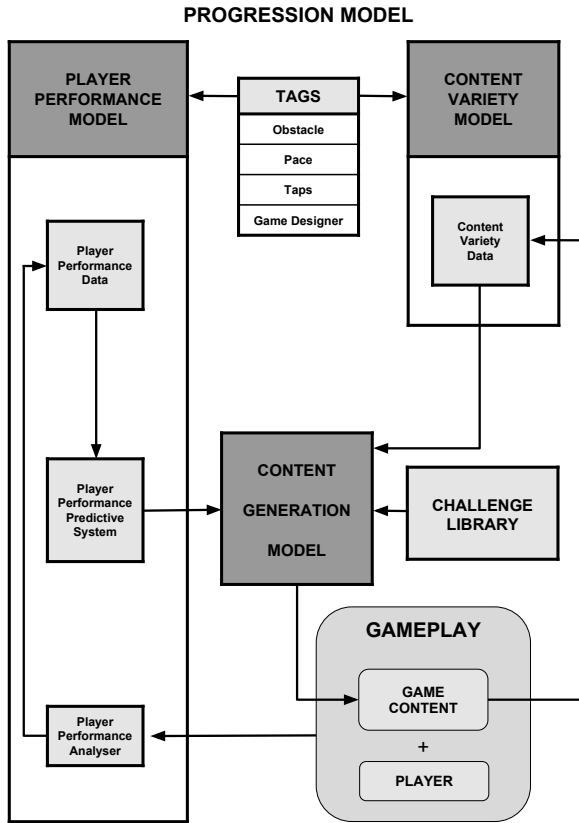


Fig. 3. Progression Model Overview

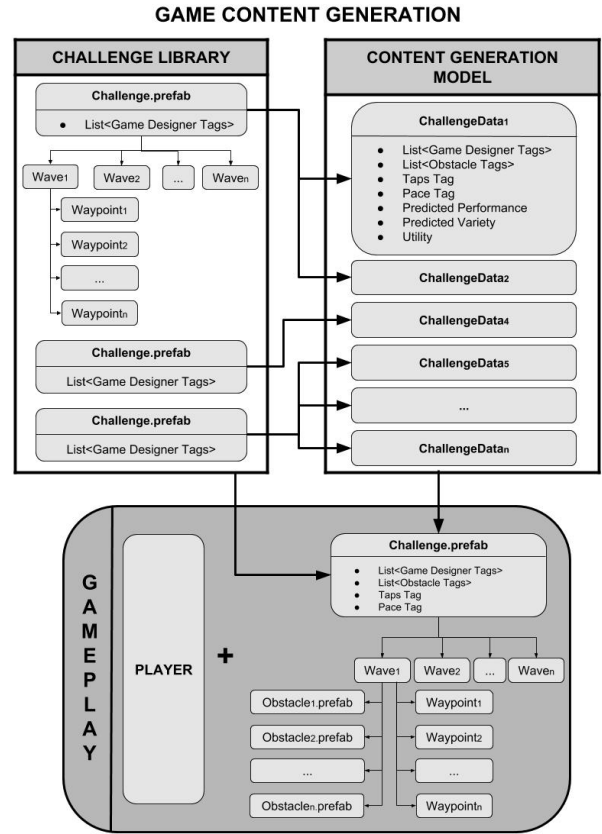


Fig. 4. Content Generation Overview.

### E. Content Generation

The Content Generation Model uses both the Player Performance Model and the Content Variety Model to generate engaging and challenging content throughout a game session. Fig. 4 shows the game content generation process.

The Content Generation Model is used to generate a new challenge at the beginning of a new run. From then on, a new challenge is generated every time there is only one obstacle left from the last generated challenge.

Content generation is divided into the following steps:

- 1) Generate a new population of 50 random meta-challenges. For performance reasons, the challenges themselves are not actually instanced. Only the parameters (meta-data) that will control the future creation of the actual challenges from the prefabs are manipulated at this stage (hence the designation meta-challenge). This meta-data includes the list of tags assigned with the challenge, the predicted player performance and how novel the content is according to interaction history, and is used to compute the challenge utility value. The meta-data created is also linked to the respective selected prefab in the challenge library.
- 2) Refine the new population of meta-challenges. Each meta-challenge is attributed random content for each wave: the number of wave obstacles; the type of each

obstacle, and; the order of the obstacles in the wave. The tags associated with the type of obstacle are then associated to the meta-challenge. The next step is to count the number of taps required to overcome all the obstacles in the wave, and assign the respective tag to the meta-challenge (e.g. if 2 taps are required, then the *Taps1-3* Tag is attributed). A random pace is then attributed to the meta-challenge and the respective tag associated with it. The challenge pace is then assigned to all obstacles of each wave of the challenge. Finally, the Game Designer Tags associated with the challenge are copied from the original challenge in the challenge library to the meta-challenge, as well as to each wave of the challenge to be later set on each obstacle when spawned.

- 3) Select the next best meta-challenge: calculate the utility of all meta-challenges using an heuristic evaluation function based on desired content variety and desired player performance and select the meta-challenge with the highest utility to be instanced and presented to the player by the Content Generation Model. The heuristic evaluation compares: the current point on the performance curve defined by the game designer with the predicted performance of the player based on the Tags associated with the meta-challenge, and; the current

point on the variety curve defined by the game designer with the history of interaction of the player with the Tags associated with the meta-challenge. These will be detailed in next sections.

- 4) Generate and activate a new challenge based on the meta-challenge selected for its highest utility. The Content Generation Model copies all the data from the chosen meta-challenge to the corresponding challenge prefab from the library and activates the new challenge.
- 5) Clean all data from the population of meta-challenges and reuse them in the next iteration.

#### F. Modelling Player Performance

The Player Performance Model was designed to evaluate the player's skill, support the generation of game content that matches the player's skill, and keep the game constantly challenging for the player. It uses in-game collected data to continuously adapt the challenges and their skill level to each individual player.

To that end, the Performance Model only requires the game designer to specify a performance curve, that will shape the progression of the player's performance over the game session. Fig. 5 shows the performance curve used for the evaluation of our model in Smash Time Arena Mode.

$$Performance(C) = \frac{\sin(C/2) - 0.01C + 3}{4}$$

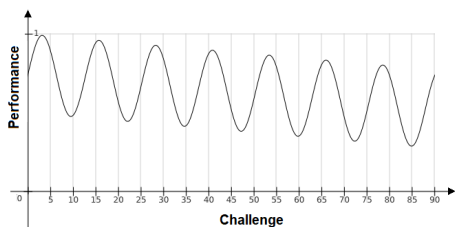


Fig. 5. Player performance curve. The curve defines what is the expected performance (between 0 and 1) of the player over time.

**Measuring Performance.** The performance value associated with a challenge represents the player's skill at overcoming the obstacles that compose the challenge. This value is computed as the weighted sum of the performance value of all the Tags associated with the challenge (Game Designer, Pace, Taps, Obstacle). The Player Performance Model stores performance data on each Tag associated with every challenge generated and presented to the player. Each Tag stores how the player performed when interacting with the last  $N$  ( $= 10$  in our implementation) challenges having this Tag associated. Each Tag starts with an initial performance value measured during playtest with both new users to the game and experienced players, which will be progressively replaced by the real player data obtained during gameplay.

While the performance of Obstacle Tags is analyzed for each obstacle individually, the performance of the other Tags (Pace, Taps, Game Designer) is based on the overall performance with all the obstacles in the challenge, using a

performance metric called TapsScore ( $TS_c$ ), that is the ratio between the taps successfully performed on all obstacles in a challenge  $c$ , and all the taps needed to overcome all the obstacles of said challenge:

$$TS_c = TapsDone_c / TapsNeeded_c$$

The value of  $TS_c$  is then assigned to the Pace Tag ( $P_{pac}$ ), Taps Tag ( $P_{tap}$ ) and Game Designer Tags ( $P_{des}$ ) performance values and recorded in the performance history of each tag.

The performance  $P_o$  of the player relative to an obstacle  $o$  reflects if the player was able to surpass it (enemy smashed,  $P_o = 1$ ) or not (enemy escaped or attacked the hero,  $P_o = 0$ ). After one obstacle is overcome or not by the player,  $P_o$  is assigned to that obstacle, and recorded in the history of the correspondent Obstacle Tag assigned to the challenge to which the obstacle belongs to.

An obstacle is considered to be overcome when all the obstacles that are spawned from it<sup>4</sup>, if there are any, are overcome. The final performance value of each Obstacle Tag of the challenge is calculated with the average of all the recorded obstacle performances with the same tag and recorded in the player performance data. The final performance value of all the Obstacles Tags is equal to the average of the individual performance of each Obstacle Tag.

The performance  $P_c$  assigned to the challenge  $c$  is calculated using the following formula, where  $(WP)_T$  is the weight of the category of tag  $T$  when computing performance (in our final implementation, we used 0.25 for all weights):

$$P_{des} = P_{pac} = P_{tap} = TS_c, P_{obs} = \frac{1}{N} \sum_{i=1}^N P_{o_i}$$

$$P_c = P_{des} * (WP)_{des} + P_{pac} * (WP)_{pac} + P_{tap} * (WP)_{tap} + P_{obs} * (WP)_{obs}$$

Every time a challenge is deactivated, its tags' performances are calculated and registered in the Player Performance Model and used to calculate and assign, with the previous formula, an overall performance value  $P_c$  to the challenge.

**Predicting Performance.** When the Player Performance Model needs to estimate the player performance against a new challenge to evaluate its adequacy in terms of the performance curve, we use the same formula that is used to compute the performance of the player in a challenge, but use the estimated performance of each tag, calculated as the average of the last  $N = 10$  recorded player performance with that tag.

#### G. Modelling Content Variety

The rarity of a challenge represents its novelty and is defined by the rarity of its associated Tags. The rarity of one specific Tag is calculated by the frequency of its appearance in the game compared to the total count of already used Tags. This means the more often a tag was used, the closer to 0 its rarity

<sup>4</sup>Remember that smashing an enemy can spawn another different enemy.

value is and, on the opposite side, the less a tag was used, the closer its rarity value is to 1. Hence, when a player starts a new game run, all the tags start with a rarity value of 1, because they were never presented to the player.

In the same way the game designer has the task to define a performance curve, he/she also has the task to define a variety curve, that will shape and guide the progression of gameplay in terms of variation of the game content that is generated by the Content Generation Model. Fig. 6 shows the variety curve used in our implementation of the progression model as an example of a possible variety curve defined by the game designer.

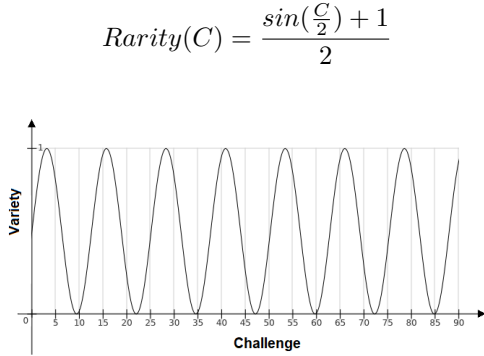


Fig. 6. Content variety curve. The curve defines how familiar or novel (between 0 and 1) the content is expected to be to the player over time.

The rarity  $R_T$  of a tag  $T$  is calculated by counting the appearance of the tag in the challenges (most recently) met, using the data stored in the Content Variety Model as shown if the following formula:

$$R_T = 1 - (RecentUsageCount_T / HistorySize)$$

Using each tag's individual rarity value, the Content Variety Model is able to assign a rarity value  $R_c$  to a challenge  $c$  using the following formula, where  $(WR)_T$  is the weight of the category of tag  $T$  when computing rarity (in our final implementation, we used 0.25 for all weights):

$$R_{obs} = \frac{1}{N} \sum_{i=1}^N R_{o_i}$$

$$R_c = R_{des} * (WR)_{des} + R_{pac} * (WR)_{pac} + R_{tap} * (WR)_{tap} + R_{obs} * (WR)_{obs}$$

## V. EVALUATION

This section describes the evaluation process used to validate our progression model. We present the procedures, results, changes and insights for each moment of the evaluation. We start by describing the preliminary evaluations that took place during development, then we detail the qualitative and quantitative evaluations that were made to evaluate the final version of the progression model.

### A. Preliminary Evaluations

We performed evaluations with real users, both novice and expert players at several points in the development process; more specifically, we had 7 people involved over a process of 5 iterations that took 5 weeks. This initial evaluations had the following objectives:

- Test the performance of the progression model;
- Test the reaction time of the progression model to the evolution of the player's skill level;
- Test players reactions obtained from the gameplay experience with the progression model;
- Get the initial values of the player performance to be used in the final version of the Player Performance Model;
- Tune the performance curve that defines the desired player skill evolution;
- Tune the variety curve that represent the desired content variety progression;
- Tune all parameters of the components from the progression model.

1) *Procedure*: The experiments started with an informal playtest of the revised Arena Mode on a mobile phone, followed by an unstructured interview with the participants to collect all type of qualitative feedback. The participants were not told that the game would be trying to adapt to them. At the end of each playtest session, all recorded values were collected and gathered to inform the progression model being developed.

2) *Results and Changes*: The experiment was repeated until we reached a point where the performance values assigned to each Tag in the Player Performance Model, when starting the game, represented a good starting point for new players.

The experiment also allowed us to tune the amount of data that would be recorded in the Player Performance Model. Although we started initially by recording all the data from the game (which is reasonable for a single play session) we ended up settling on a history window of the 10 most recent challenges. The model would take into account only the most recent experience of the player (both in terms of performance and content variety) and ignore what happened earlier, which we verified was no longer relevant for assessing player skill or their perception of novelty. This would additionally insure memory usage would only increase proportionally to the number of Tags and data would never become too large to slow the game down on the targeted mobile phone.

Finally, we decided to group all Tags associated with the number of Taps required to overcome a certain challenge in sets of 3 rather than individually (e.g. *Taps1-3*, *Taps4-6*). This was a good compromise between update frequency (insuring all data is recent and reflect player skill) and challenge differentiation in the Player Performance Model.

### B. Final Evaluation

After having adjusted all the parameters of our progression model, we proceeded to qualitatively and quantitatively validate our approach. The objective of this evaluation was to

test if the progression model is able to adapt to all players and provide enjoyable and challenging play experiences every time a player starts a new game, resulting in longer play sessions.

1) *Procedure*: To validate our approach, we compared the developed Player Performance Model to the progression used in the original game and, as such, tested two different versions of the Arena Endless Mode of Smash Time:

- Smash Time “O”: original progression model in the game on the stores at the time of the evaluation, in which the difficulty of the challenges was tuned to the duration of the game session;
- Smash Time “N”: version including the Player Progression Model described in this paper.

32 participants took part in this study: 87.5% gamers (participants that play games casually when the opportunity presents itself or reserve time in their schedule to play games), 65.6% with experience in games of the same genre, and 46.9% with previous experience with the original Smash Time game. 16 participants were randomly attributed to each condition.

Playtest started with a brief presentation of Smash Time, as participants were informed they would have to play the game and answer to a short questionnaire afterwards. Participants were never told what was actually being tested. After the presentation, the participants played the first three campaign levels, that serve as the game tutorial. During this time, the participants were encouraged to ask any questions. We then made a short demonstration of the Arena Mode, focusing on the time mechanics and how players could extend time and achieve better scores. Once again, the participants were encouraged to ask any remaining questions before starting the actual playtest.

All participants used the same tablet device during the tests and could play the game for as much or as little as they wanted, without any intervention from the researcher (except if explicitly requested). After a participant would stop playing, they would be asked to fill the Game Experience Questionnaire (GEQ) [20]. Finally, an unstructured interview would take place to gather some additional feedback. The researcher would then gather the game data automatically tracked by the game and accessible directly on the device through the SRDebugger Tool<sup>5</sup>, relative to: total playtime; number of games started; number of times a hero was hit; number of times a game ended with a timeout, and; number of times the user quit the game, and fill an online form with the Collected Game Data. Both the Collected Game Data and the data from the Game Experience Questionnaire were exported to a CSV file and imported into IBM SPSS Statistics 24 for analysis. Our hypothesis was that the experience reported by the GEQ and both playtime and number of games played would be significantly statistically different in the two conditions.

2) *Results*:

a) *Collected Game Data*: A Mann-Whitney U statistical test performed on the previously mentioned variables revealed

statistical significance on: playtime ( $U = 39, p < 0.001$ ), number of games started ( $U = 28, p < 0.001$ ) and, number of times a game ended with timeout ( $U = 29, p < 0.001$ ). The Mean Ranks in both conditions suggests Smash Time “N” got better results than Smash Time “O” on these dimensions. The median values of the four variables of the collected game data are presented in Table I.

	Version “O”	Version “N”
<b>Game Data</b>		
Playtime (seconds)	381.5	910.0
Games started count	2.0	4.0
Hero attacked game ended count	0.5	1.0
Timeout game ended count	1.0	3.5
<b>GEQ Components</b>		
Competence	4.0	3.7
Sensory and Imaginative Immersion	3.17	3.67
Flow	3.3	3.8
Tension / Annoyance	1.0	1.3
Challenge	2.7	3.3
Negative Affect	1.0	1.5
Positive Affect	4.0	4.4
<b>GEQ Selected Items</b>		
I found it tiresome	0.0	1.0
I found it impressive	2.0	3.0

TABLE I  
MEDIAN VALUES OF THE COLLECTED GAME DATA VARIABLES AND GAME EXPERIENCE QUESTIONNAIRE (GEQ)

b) *Game Experience Questionnaire*: We compared the dimensions of experience rated by the seven components of the GEQ “Core Module” [20]. None of these components presented statistical significance on a Mann-Whitney U test. Two items, however, were found statistically significantly different: “I found it tiresome” ( $U = 70.5, p < 0.05$ ) and “I found it impressive” ( $U = 74, p < 0.05$ ). These results suggest the participants felt Smash Time “N” was more impressive but also more tiring in comparison with Smash Time “O”. The median values of the components and two items are presented in Table I.

3) *Summary*: The evaluation performed suggests that the Player Progression Model described in this paper could be a good approach to be used in future updates to the original game, as well as in other endless single player digital games. Players did play the game more times and for longer periods of time, which were the two main objectives of our work. The feedback received through the GEQ also suggests players found this approach more impressive and challenging than the original game, which progression model increased the difficulty of the game based on the duration of a game session, through the use of preset rules. Therefore, we believe to have developed a robust and dynamic progression model that takes into account player skill and content variety to create an adapted play experience that does not disrupt from the original game and may well improve it, while removing the burden of specifying the exact parameters of content progression from the game or level designer.

<sup>5</sup>Stompy Robot, SRDebugger - Unity Console & Tools On-Device, 2015. Available at <https://www.stompyrobot.uk/tools/srdebugger>

## VI. CONCLUSIONS

In this paper, we presented a skill-based progression model for endless single player digital games based on player performance and content variety, that creates more challenging and engaging play experiences for the player, leading to an increase in the duration of play sessions. We presented a case study of the model in “Smash Time”, a mobile smasher video game commercially available, to support the adequacy of the approach.

Our approach consists in creating a player model that explicitly tracks both the player skill progression as well as the variety of the content the player experiences through play. This information is used to control the pace of the challenges building the experience based on player performance, alternating between the tension of new and known but harder versions of previously visited challenges and the relaxing interaction with challenges that have already been overcome but are still interesting for the player.

To support our approach, we presented an experiment with both novice and experienced players, that played a version of “Smash Time” with the initial parameters of our progression model tuned after a short exploratory study. The goal of the experiment was to test how enjoyable the play experience created by our progression model is for players and whether play sessions using our approach are longer. Both quantitative and qualitative data was gathered: players filled a game experience questionnaire and in-game data was automatically tracked. The results suggest we were able to successfully increase both the duration of each game session and the number of times a game is started by the player, by quickly bringing the players into a flow channel adequate to their skill, and keeping them inside that channel with content that is neither boring or frustrating while varied. By providing an alternate progression model to traditional games, one that is always different but adapted and consistent to the player performance, we also increased replayability. We also believe (although this still has to be experimentally verified) that the approach could “ease-in” players returning to the game after some time away from it.

This approach also has a strong impact on the game development process: it reduces the time game designers have to spend fine-tuning the parameters of their games and the rules controlling content presentation through the game, e.g. the probability of certain challenges appearing at certain levels, as was the case in the original version of the test-bed game. According to the informal interviews with the game designers that worked on the original version of the progression of Smash Time, it took around 6 months of playtest and parameter tweaking to fine tune the progression model, when compared to the 5 weeks we took to implement the proposed progression model. Finally, we believe the approach could be applied to other games of the same genre and even ported to other genres by carefully redefining the core concepts of Challenge, Obstacle and Tags in the design space of the new genre, while maintaining the same approach.

## ACKNOWLEDGMENTS

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2019.

## REFERENCES

- [1] M. Csikszentmihalyi, *Flow: The Psychology of Optimal Experience*. Harper & Row, 1990.
- [2] J. Togelius, E. Kastbjerg, D. Schedl, and G. N. Yannakakis, “What is procedural content generation?: Mario on the borderline” in *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games*, p. 3, ACM, 2011.
- [3] J. Togelius, A. J. Champandard, P. L. Lanzi, M. Mateas, A. Paiva, M. Preuss, and K. O. Stanley, “Procedural content generation: goals, challenges and actionable steps” in *Dagstuhl Follow-Ups*, vol. 6: *Artificial and Computational Intelligence in Games*, pp. 61–75, 2013.
- [4] N. Shaker, J. Togelius, A. Liapis, R. Lopes and R. Bidarra, “Constructive generation methods for dungeons and levels” in *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*, Springer, Chapter 3, pp. 31-55, 2016.
- [5] M. J. Nelson, J. Togelius, C. Browne, and M. Cook, “Rules and Mechanics” in *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*, Springer, Chapter 6, pp. 99-121, 2016.
- [6] N. Shaker, J. Togelius, J. Dormans, “Grammars and L-systems with applications to vegetation and levels” in *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*, Springer, Chapter 5, pp. 73-98, 2016.
- [7] W. Yin-Poole, “How many weapons are in Borderlands 2?”, <https://www.eurogamer.net/articles/2012-07-16-how-many-weapons-are-in-borderlands-2>, 2012.
- [8] Y. Cheong, M. O. Riedl, B. Bae and M. J. Nelson, “Planning with applications to quests and story” in *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*, Springer, Chapter 7, pp. 123-141, 2016.
- [9] G. N. Yannakakis and J. Togelius, “Experience driven procedural content generation” in *IEEE Transactions on Affective Computing*, vol. 2, Issue. 3, pp. 147–161, IEEE, 2011.
- [10] S. Bakkes, S. Whiteson, G. Li, G. V. Vişniuc, E. Charitos, N. Heijne and A. Swellengrebel, “Challenge balancing for personalised game spaces”, in *IEEE Games Media Entertainment*, 1-8, 2014
- [11] P. M. Blom, S. Bakkes, C. T. Tan, S. Whiteson, D. Roijers, R. Valenti and T. Gevers, “Towards Personalised Gaming via Facial Expression Recognition”, in *Proceedings of the Tenth Annual AAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2014
- [12] C. Martinho, P. Santos, and R. Prada, “Design e Desenvolvimento de Jogos”, ch. 4. Lisboa, Portugal: FCA, 2014.
- [13] J. Chen, “Flow in games (and everything else)” in *Communications of the ACM*, vol. 50, No. 4, pp. 31–34, ACM, 2007.
- [14] D. Cook, “The chemistry of game design.” [https://www.gamasutra.com/view/feature/129948/the\\_chemistry\\_of\\_game\\_design.php](https://www.gamasutra.com/view/feature/129948/the_chemistry_of_game_design.php), 2007.
- [15] N. Shaker, G. N. Yannakakis, and J. Togelius, “Towards player-driven procedural content generation,” in *Proceedings of the 9th conference on Computing Frontiers*, pp. 237–240, ACM, 2012.
- [16] N. Shaker, M. Abou-Zleikha, and M. Shaker. 2015. “Active Learning for Player Modeling.” in *Proceedings of the 10th International Conference on the Foundations of Digital Games (FDG 2015)*, ACM, 2015.
- [17] P. Pereira, “Modelling progression in video games.” MSc Thesis, Instituto Superior Técnico, University of Lisbon, 2016.
- [18] F. Bicho, C. Martinho, Multi-dimensional Player Skill Progression Modelling for Procedural Content Generation in *Proceedings of the 13th International Conference on the Foundations of Digital Games (FDG 2018)*, Article 1, pp. 1-10, ACM, 2018.
- [19] A. Zook, S. Lee-Urban, M. R. Drinkwater, and M. O. Riedl, “Skill-based Mission Generation: A Data-driven Temporal Player Modeling Approach,” in *Proceedings of the The third workshop on Procedural Content Generation in Games*, ACM, 2012.
- [20] K. Poels, Y. A. W. de Kort and W. A. IJsselstein, “D3.3 : Game Experience Questionnaire: development of a self-report measure to assess the psychological impact of digital games.”, Technische Universiteit Eindhoven, 2007.