Skill-based Progression Model for Smash Time

João Catarino
Instituto Superior Técnico
Lisboa, Portugal
jcbpc@tecnico.ulisboa.pt

ABSTRACT

Nowadays an increasing number of games use procedurally generated content to provide immersive and engaging gaming experiences. Most of the endless single player games generate content just based on the difficulty of the challenges combined with the duration of the current game session. In this dissertation, we address the problem of keeping the players engaged in a game for longer periods of time generating game content that is modeled according to the player's performance, to improve the gameplay experience. In order to increase the overall gameplay experience of a game, we should increase the feeling of progression. To solve this problem, we propose a progression model that creates content based on the player skill and performance to overcome those challenges. We also propose to control the variety of the challenges to create a better gameplay experience. We theorize that, by providing a progression that is adapted to the player based on the player skill, keeping the variety of the challenges will lead to more engaging and fun gameplay experiences, increasing the replayability in single player games. We propose a progression model that uses player skill and challenges' variety, in the endless level of the mobile game Smash Time. With this work, we believe to have created a skill-based progression model that is robust and dynamic enough to be used in different types of games and that excludes the need of using preset difficulty settings and/or adaptation rules. The results from playtests with users suggest that the developed skill-based progression model is able to increase the number and duration of the game sessions. The results also suggest that the progression model has the potential to increase player immersion, and consequently to create more engaging gameplay experiences. With the potential to create longer and more engaging gameplay experiences as well as to increase their frequency, comes the possibility to, ultimately, increase the overall lifetime of the game itself.

Keywords

player skill; content variety; progression; procedural content generation; dynamic difficulty adjustment; player modelling; video games

INTRODUCTION

Most of the endless single player games generate content just based on the difficulty of the challenges and usually the difficulty setting is achieved through the adaptation of the challenges to a sample of the players population. In order to increase the overall gameplay experience of a game, we should increase the feeling of progression (flow). Csikszentmihalyi[1] concluded that a person's skill and the difficulty of a challenge can create different emotional states on that person. He found that people become bored when dealing with an easy task if their skill level is high and, alternatively, they become anxious when dealing with a difficult task if their skill level is too low. To solve this problem, we propose a progression model that creates content based, not just on the difficulty of the challenges, but also on the player skill to overcome those challenges. As an extra element to create a better experience, we also propose to control the variety of the challenges. The variety of a challenge represents its novelty, i.e., whether the challenge characteristics have been previously presented to the player. This is achieved with the classification of the challenges' pace, number of obstacles, game designer challenge description and type of obstacles that compose the challenge. We theorize that, by providing a progression that is adapted to the player based on his/her skill, keeping the variety of the challenges, may lead to more engaging gameplay experiences, creating a stronger connection between the player and the game, and possibly resulting in longer gameplay experiences. We propose a progression model that uses player skill and content variety, in the endless level of the mobile game Smash Time, a smasher game with more than 250.000 downloads on the iOS, Android and Windows Phone platforms.

RELATED WORK

This chapter starts with a brief description of what is Procedural Content Generation (PCG) and some of its goals followed by a survey on previous work on PCG and its use in the gaming industry. We finish the current chapter with an overall discussion about the related work bearing in mind the focus of this 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 parts of a level or a map to complete levels and maps, game rules, 2D textures and 3D models, characters and items, music, stories and side quests, etc. In this section we will make a quick survey through some of the most iconic and important games which have paved the way PCG can be used in games and

some interesting games that were groundbreaking on the way they used PCG to generate its content.

User Experience

The player is the main element when we talk about the experience created by the game. In order to develop good games with mechanisms that boost the experience we need to have a good understanding about the player, like their: motivations such as needs, preferences, interests, expectations, values, fears and dreams; limitations; capabilities; knowledge; and the context in which they play a game like, with who, 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 [4]. As Chen [5] pointed out, each player is different and experiences the same games in different ways, due to their personality, skills and expectations when playing a game. This suggests that to satisfy different types of players, the game should be able to adapt itself to the preferences of the player. Focusing in the player experience and in order to generate effective and meaningful content, Yannakakis and Togelius [6] proposed a framework for PCG driven by computational models of user experience based on the personalization of user experience through affective and cognitive modeling combined with real time adjustment of the content according to user needs and preferences.

User Skill

Focusing in the player, Cook [7] described the player model as "The player is 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. Some skills are physical, such as making a sculpture while others are on the conceptual domain, such as observing a map and planning the best path to go from one point to another. 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 gameplay 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 [8].

Pereira [9] developed a progression modeling tool to understand the player skill evolution in order to provide appropriate challenges in a 2d platformer game. This tool uses a library of challenges combined with a skill-based player model that is based on the player mastery level (e.g. uninitiated, partially mastered, mastered) for specific challenges (e.g. wall, hole) and game mechanics (e.g. jump, slide, double jump) to know when to unlock new and more difficult challenges and mechanics. In this progression model, a challenge become mastered after several challenges of that type are successfully overcome and a mechanic becomes mastered when its used several times to overcome a challenge. The progression is guided by preset adaptation rules created by the game designer, that specify how the progression of the game should evolve according to the player's skill evolution. The game designer creates a progression graph that specify the dependencies and constraints between all the challenges the game can generate and the mechanics that can be used by the player. The game starts with few or just one mechanic and challenge unlocked to the player and, as the player skill evolves the game starts unlocking new and more difficult challenges and mechanics, according to the progression graph defined by the game designer.

Zook et al. [10] proposed a model for skill-based mission generation that tries to solve 2 problems: challenge tailoring, or in other words "the problem of matching the difficulty of skill-based challenges over the course of a game to match player abilities"; and challenge contextualization, related to the fact that the game should provide appropriate motivating story context for the skill-based challenges and thus resulting in the creation of story content that motivates game play in between 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. As a result of this study, they propose a five criteria data-driven player model: predictive power; accuracy; efficiency; generative sufficiency and temporality to guide skill-based mission generation combined with story events.

Discussion Overview

Inspired on the work of Chen et al. [5] about the importance of the player in the development of video games, we will try to value each player individually, with his/her own capabilities and limitations, in order to provide a different and personal gameplay experience to each one of them trying to satisfy different types of players with the same game. Furthermore, we will create a player model, that will be updated over the course of time and play sessions, trying to give each player a different and appropriate experience every time he/she plays the game according to the evolution of his/her skills.

Like in the work developed by Pereira [9], we will try to understand how the player skill evolves, also using the concept of game content being presented in the way of challenges, to provide a challenging and engaging game progression to the players. In contrast with that progression model, that uses a progression graph of dependencies and constraints between challenges, preset by the game designer to unlock new content, all of the content in our progression model is unlocked at the start of the game. The approach in the presented progression model, is that is going to be the player's skill evolution to show the game that the player is ready for more challenging content, according to the defined

performance curve defined by the game designer. With this approach we want to avoid, as the game dimension and complexity increases, the task of the game designer to create new challenges not to become too difficult and time-consuming, due to the fact that the challenges do not need to be interrelated, as it happens with a progression graph and all its dependencies and constraints between the challenges. Another reason for the exclusion of the progression graph is that, without it, we also remove the game designer's intuition factor, about what would be the difficulty level of each challenge and the decision of all the possible connections and dependencies between the challenges. Instead, the game designer has a simpler task, that is to define a performance curve that will allow the game itself to choose which content to provide to the player. The proposed progression model also uses a library of challenges in which each challenge can be created in a modular way without dealing with its relation with all the other challenges already existing in the challenges' library. This leads to a less time-consuming and easier challenges' creation task for the game designer. Finally, with the exclusion of the progression graph, the game is able to provide different gameplay experiences each time a player plays the game, instead of providing always the same progression evolution for all the players in all game sessions.

TESTBED GAME

Smash Time has fast gameplay mechanics, that result from the combination of elements from classic games like Whac-a-Mole and Space Invaders, 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 and the animals that are trying to escape from them to survive. The player goal is to smash the enemies and clear all the incoming waves. To smash one enemy and receive one or more score points for it, the player must tap on it with a finger.

Arena Mode

In this game mode, players try to play as much time as possible, to reach the highest score possible, on an infinite level with a timer that can be extended to the maximum of 99 seconds. According to the player's performance against killing the enemies of the enemies' sequences, that are generated by the game, they are rewarded with the possibility to extend the timer, by tapping and smashing a special type of enemy that appears more often the larger the number of enemies' sequences completed by the player.

PROGRESSION MODEL

In this dissertation thesis, we propose a skill-based progression model for endless single player videogames and the present chapter will serve to explain our proposal in detail. The progression model developed in this dissertation thesis was implemented, using Unity engine, on top of the Arena game mode of Smash Time. The main goal of our progression model is to provide immersive and engaging gameplay experiences that will make players want to play more often and for longer periods. To achieve this, we kept in mind these principles:

1. the game should allow and support player skill development;

2. the game should be constantly challenging, with care to avoid not becoming too difficult, and therefore stressful for the players, nor too easy, and consequently boring for the players;

3. the game should match the player's skill level at all times throughout a game session, increasing the level of challenge as the player moves forward through the game and increases their skill level;

4. the game should provide different levels of challenge for different players;

5. the game should vary its content and provide new challenges at an appropriate pace.

Following these principles, we propose a skill-based progression model that is composed by the following components:

- Concepts:
  - Challenges;
  - Obstacles;
  - Tags.
- Models:
  - Player Performance Model;
  - Content Variety Model;
  - Content Generation Model.

In this chapter, we will analyze the progression model concepts and components. More specifically, how the Challenges, Obstacles and Tags are used by the three main components that compose the core of the developed progression model: the Player Performance Model; the Content Variety Model; and the Content Generation Model.

Challenges

Challenges are formations created by the game designer with one or more paths (or wave paths) that guide the movement of the obstacles. 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 later in real time. A challenge is composed by a set of waves that are going to spawn 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. The

progression model has the task to generate new challenges as the game progresses, while the player has the task to overcome those challenges.

Obstacles

An obstacle can be something that requires the player to use his/her skills to be overcome. In the context of the testbed game, the obstacles of the Arena game mode can be: a Red Enemy; a Green Enemy; a Blue Enemy; and a Purple Enemy. To overcome each obstacle, the player must tap on it. When the player taps on an enemy, it smashes and kills that enemy. Some enemies generate another enemy after being killed by the player. The Figure 1 illustrates the arena obstacles' backward evolution phases when the player smashes one enemy with a tap. Even though each obstacle is a different enemy and all of them have one life, meaning this that they die with a single tap from the player, as some enemies generate other enemies, we consider that an obstacle is completely overcome just when the Purple and last enemy in the de-evolution order is smashed and killed. On the basis of this premise, we can distinguish an enemy that was spawned inside one challenge wave and an enemy that was spawned from another enemy that was already alive.

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

Figure 1. Arena obstacles' backward evolution phases.

Tags

The proposed progression model uses a set of tags to give context to the game content, more precisely to give context to the challenges and obstacles of the game. The context of a challenge and its obstacles describes and classifies the challenge, using a set of tags that are assigned, both by the game designer and by the progression model. The tags used by the progression model to describe each challenge, have the purpose to provide a way to assign, each given challenge to the player, a performance value and a variety value, and are organized into four separated groups

Game Cycle

This section presents the game cycle with the progression model integrated. To better understand the game cycle, the progression model architecture diagram is visible in the Figure 2. This diagram represents the game cycle of the game with the progression model which is composed by the following steps, that are repeated in a loop:

1. Generate a new challenge (game content) to present to the player, using:

• the Player Performance Predictive System from the Player Performance Model;

• the Content Variety Data from the Content Variety Model;

• the Challenge Library.

2. Register the player response 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. Register the player performance data in the Player Performance Model;

5. Register the challenge variety data in the Content Variety Model;

6. Go to step 1.

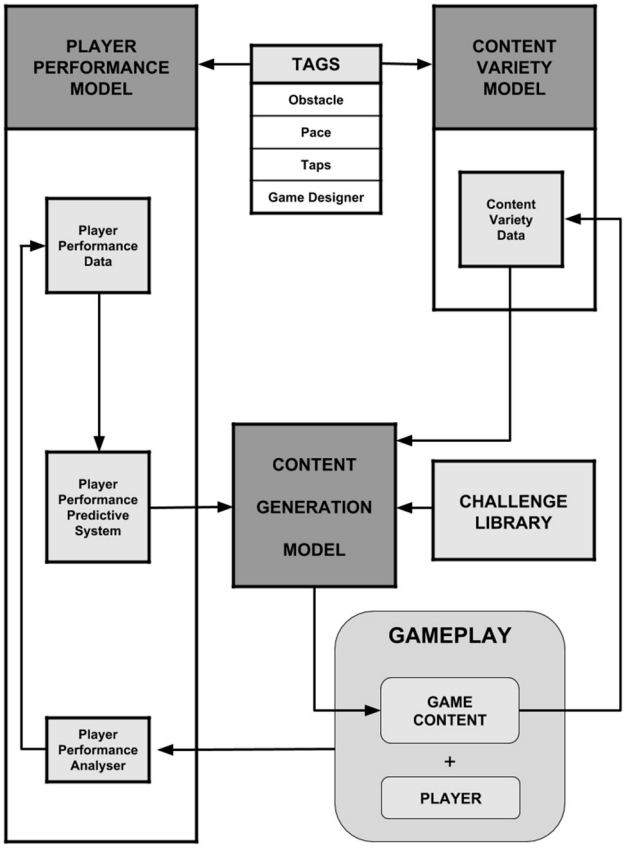

Figure 2. Game cycle with the Progression Model Architecture.

Content Generation Model

The Content Generation Model uses both the Player Performance Model and the Content Variety Model to be able to generate engaging and challenging game content throughout a game session. Figure 3 shows the game content generation process, focusing in the connection between the Challenge Library and the Content Generation Model, and type of content that is used and generated by the Content Generation Model.

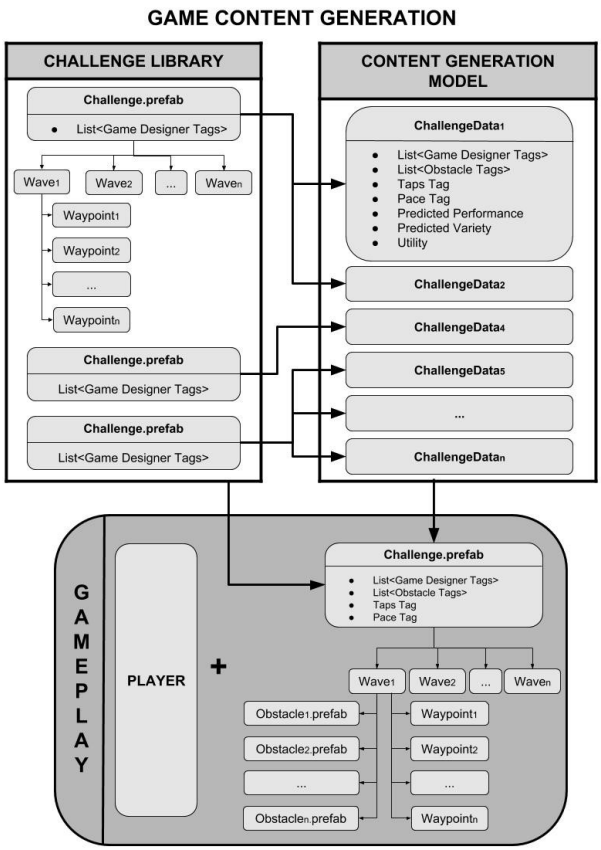

Figure 3. Game Content Generation (Challenge Library + Content Generation Model).

The Content Generation Model is used to generate a new challenge in the beginning of new a run and, from that moment, every time there is only 1 obstacle left from the last generated challenge. This model is composed by the following phases:

1. Generate a new population of 50 random challenges: this population of new challenges is, in fact, a population of metadata challenges that will contain only the data needed to generate a new challenge. This data includes the list of the various tags that will be assigned to the challenge, as well as the predicted player performance value and the predicted content variety value that are both used to calculate the overall challenge utility value. Instead of generating a population of Challenge prefabs, that would take a big amount of time and memory to be instantiated, especially given the fact that the testbed game runs on mobile devices, this model generates a population of metadata challenges. The class Challenge Data was created for this purpose, as it serves as a container to store the data of a possible challenge, allowing the creation of lots of possible challenges and thus, optimizing the efficiency and performance of the progression model. After the generation of the metadata challenges' population, each Challenge Data is connected to a, randomly selected, challenge prefab in the challenges' library;

2. Populate the new population of metadata challenges: each Challenge Data is populated with random content for each challenge's wave: the wave obstacles' quantity; the wave obstacles' types; and the wave obstacles' order. After setting the obstacles that will be spawned, the corresponding Obstacle Tags are assigned to each Challenge Data. The next step is to count the number of taps needed to overcome all the spawned obstacles and the obstacles that will appear from the originally spawned obstacles of the Challenge and assign the correspondent Challenge Taps Tag to the Challenge Data. A random pace is set for the challenge and the corresponding Challenge Pace Tag is assigned to the Challenge Data. The challenge pace is then assigned to each wave of the challenge, to be later set to each obstacle when it is spawned. Finally, the challenge Game Designer Tags are copied from the original challenge from the challenges' library to the Challenge Data, as well as to each wave of the challenge to be later set on each obstacle when spawned;

3. Select the next possible best challenge to be generated: run through all the populated metadata challenges and calculate their utility values using an heuristic evaluation function that combines the predicted content variety value of a challenge with its predicted player performance value. In this end of this phase, the Content Generation Model defines which one is the most useful challenge to be next generated and presented to the player. To get the best challenge candidate, the heuristic evaluation compares: the next desired player performance value, from the performance curve defined by the game designer, with the predicted player performance value of each metadata challenge; and the next desired content variety value, from the variety curve also defined by the game designer, with the predicted content variety value of each metadata challenge;

4. Generate and activate a new challenge: after choosing the next best possible challenge, from the metadata challenges' population, the Content Generation Model copies all the data from the chosen Challenge Data to the corresponding challenge prefab from the library and activates the new challenge;

5. Clean all the data from the last generated and populated metadata challenges' population and reuse again this population of metadata challenges on the step 1.

Player Performance Model

The following Player Performance Model was designed to evaluate the player's skill level, in order to allow and support the game to generate content that matches the player's skill level and to keep constantly challenging the player, as the player progress through the game. Since this Player Performance Model is adapted with data collected every time the player plays the game, it also allows the game to adapt itself and provide different levels of challenge for different players.

To achieve this, the game designer specifies a performance curve, that will shape the progression of the player's

performance as it determines the difficulty flow of a game session. Figure 4 shows the used performance curve in this progression model as an example of a possible performance curve defined by the game designer.

Player Performance Analyzer

The performance of a challenge represents the player's dexterity to overcome the obstacles that compose the challenge. The value of the player performance relative to a generated and presented challenge is calculated using the combination of all its tags (Challenge Game Designer Tags, Challenge Pace Tags, Challenge Taps Tags, Obstacle Tags) performance values considering their categories' performance weights.

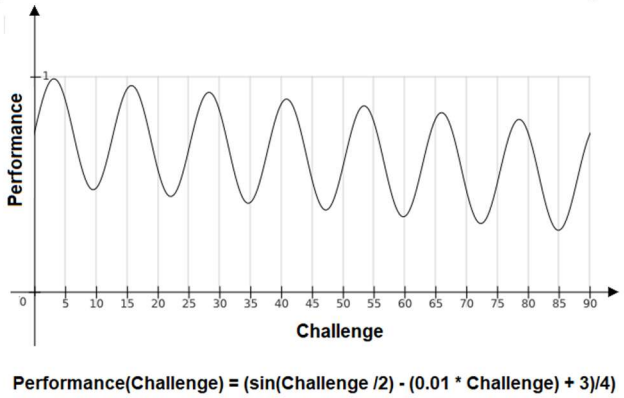

Figure 4. Player Performance Curve.

The Player Performance Model stores performance data relative to each tag of the progression model and relative to every challenge generated and presented to the player. The tags have a performance history of the last 10 performance values and the challenges' performances are recorded since the beginning of each game run. When the game is installed by a new user, each tag starts with a bootstrap performance value, that was obtained during play testing sessions with both new users to the game and experienced users. With each arena run session this bootstrap performance values will be replaced by real player performance values obtained through the gameplay data. While the Obstacle Tags' performances are analyzed individually to get more granular and accurate data, the Challenge Tags' (Pace, Taps, Game Designer) performances are calculated in a macro point of view, with all the challenge obstacles contributing to those tags' performance, using a performance metric called Challenge Taps Score, that is calculated from the counting of all the taps needed to overcome all the obstacles that will be spawned in a challenge and the taps successfully done on obstacles, calculated according to the following formula:

$$ChallengeTapsScore = TapsDone \div TapsNeeded$$

The performance of an obstacle reflects if the player was able to overpass it (enemy smashed) or not (enemy escaped or attacked the hero). After one obstacle is overcome or not by the player, a performance value is assigned to that obstacle, and recorded in the performance history of the correspondent Obstacle Tag assigned to the challenge to which the obstacle belongs to, with one of the following values:

$$Performance(ObstacleOvercome) = 1$$

$$Performance(ObstacleNotOvercome) = 0$$

An obstacle is considered to be overcome when all the obstacles that are spawned from it, if there are any, are overcome. The performance assigned to the challenge is calculated using the following formula:

$$Performance(Challenge) =$$

$$Performance(GameDesignerTags) * GameDesignerWeight$$

$$+ Performance(PaceTag) * PaceWeight$$

$$+ Performance(TapsTags) * TapsWeight$$

$$+ Performance(ObstaclesTags) * ObstaclesWeight$$

Every time a challenge is deactivated, its tags' performances are calculated and registered on the Player Performance Model and used to assign an overall performance value to the challenge. The Challenge Taps Score value is calculated and assigned to the Pace Tag, Taps Tag and Game Designer Tags performance values to be afterwards registered in each tag performance history in the player performance data. 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.

Player Performance Predictive System

When the Player Performance Model needs to estimate what would be the player performance against a new challenge, it looks at the estimated performance of all the tags that are assigned to that challenge, according to each tag category weight, like when the player performance is analyzed and calculated as explained above. The estimated performance of each tag is calculated by the average value of the last 10 player performance values recorded

Content Variety Model

The variety of a challenge represents its novelty and is defined by the variety of its tags. Challenges have variety values between 0 and 1. The variety 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 variety value is and on the opposite side the less a tag was used, the closer its variety value is to 1. Hence, when a player starts a new game run, all the tags start with a variety 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 the gameplay in terms of variation of the game content that is generated by the Content Generation Model. Figure 5 shows the used variety curve in this progression model as an example of a possible variety curve defined by the game designer. The variety of a tag is calculated using the counting of the used tags in the already

generated challenges, through the data stored in the Content Variety Model as shown if the following formula:

$$Variety(Tag) = TagUsageCount \div TotalTagsUsageCount$$

Using each tag's individual variety value, the Content Variety Model is able to assign a variety value to a challenge using the following formula:

$$Variety(Challenge) =$$
$$Variety(GameDesignerTags) * GameDesignerWeight$$
$$+ Variety(PaceTag) * PaceWeight$$
$$+ Variety(TapsTags) * TapsWeight$$
$$+ Variety(ObstaclesT ags) * ObstaclesWeight$$

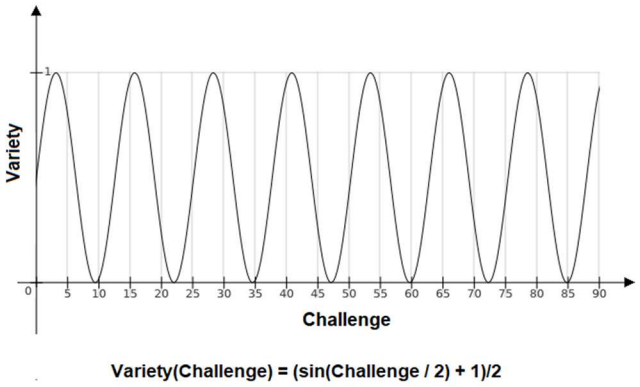

Figure 5. Content Variety Curve.

EVALUATION

In this chapter, we report the testing made on the progression model presented in the previous chapter, presenting the procedures, results, changes and insights for each moment of the evaluation. In order to validate if the generated player-adapted content improves the player experience we evaluated the quality and potential of the implemented progression model with playtesters, in several phases during the development of the proposed solution. The first part of the evaluation with real users, novice and expert players of the testbed game, was made to adjust the parameters of the progression model and to verify its potential. Next, we will describe in detail the qualitative and quantitative evaluations made to validate of our approach. To conclude this chapter, we will present the insights gained from the different evaluations that were made.

Preliminary Evaluations

Preliminary evaluations were made several times through the development of this progression model. These evaluations were made with real users, both novice and expert players of the testbed game. These 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 skill level;

• Test players reactions obtained from the gameplay experience with the progression model;

• Get the bootstrap values of the player performance to be used in the final version of the Player Performance Model;

• Refine the performance curve that defines the wanted player skill evolution curve;

• Refine the variety curve that represent the wanted content variety progression;

• Fine tune every parameter of the progression model components.

Preliminary Evaluations - Procedure

These evaluations were made without any formal playtest guideline, it was based on informal playtesting the new arena followed by an unstructured interview to the participants to collect all type of feedback without telling the participants what they were testing in order to not bias the gameplay experience and the participants feedback. In the end of each playtest session the values recorded by the Player Performance Model, accessible on the mobile, were collected and gathered on the progression model being developed.

Preliminary Evaluations - Results and Changes

One of the goals of this preliminary evaluations was to get the bootstrap values of player performance of the several tags that are stored in the Player Performance Model. This process was done through several iterations, each of them with changes on the values collected from the users' playtests, in order to tune those bootstrap values to the point that they represent a good starting point to new users. After having tuned the bootstrap player performance values that compose the Player Performance Model of the progression model, we were ready to do qualitative and quantitative evaluations of the implemented solution. As these experiments were repeated several times through the development of the progression model, the number of recorded player performance entries for each tag changed several times. In the first version used on these preliminary evaluations, the Player Performance Model had saved data related to all challenges and obstacles presented to the player since the game was installed. This first experiments revealed that this approach was not appropriate because on the one hand, the player data become too large and the game started to slow down a little bit after some game sessions and on the other hand, as all the times a new challenge is generated by the Content Generation Model, the Player Performance Model takes into account everything the player did since the beginning, which means that we were not giving more relevance to the most recent actions of the player. To solve both of these problems the recorded player actions window was updated to keep track of the last 20 obstacles presented to the player. As a result of later experiments and in order to decrease the reaction time of the progression model to the player skill evolution, this value was later changed to the last 10 player performance records for each tag. With a smaller

performance history window, with the most recent player performance data, the progression model is able to adapt more easily to the player skill progression not being tied up to what happened a long time ago and that is no longer relevant to the current player skill level. One of the changes that were made after these experiments, was to change the Challenge Taps Tags to have each tag representing a group of number of taps (e.g. Taps1-3 = [1,3] taps; Taps4-6 = [4,6] taps, Taps7-9 = [7,9] taps; ...) instead of each tag matching a specific number of tags (e.g. Taps1 = 1 tap; Taps2 = 2 taps; Taps3 = 3 taps; ...). With this change we hope that in most arena runs, the player performance data is updated, with recent data, for the largest possible number of the existing Challenge Taps Tags in the Player Performance Model. If the tags only represented one value of taps, there was the possibility that several arena runs would pass, until one or more tags' performance data history would be updated, due to no challenges being generated with some values of taps needed, what would lead the progression model to keep old player performance data that, once again, could no longer be relevant to the current player skill level. After having the bootstrap values of the progression model and all parameters adjusted, we were ready to qualitatively and quantitatively test the implemented solution.

Final Evaluation

The qualitative and quantitative evaluations made to validate the developed progression model are described below.

Final Evaluation - Procedure

For this study, we looked to find people from all types:

• gamers and non-gamers;

• with and without experience in games of the same genre;

• with and without previous experience in the used testbed game.

The objective of this evaluation was to test if the progression model is able to adapt itself to any type of player to provide enjoyable and challenging gameplay experiences every time a player starts a new game session. To validate this hypothesis, we compared the developed progression model in this study with the previous progression model already being used in the testbed game. Hence, we tested two different versions of a progression model on the arena endless level of Smash Time:

• Smash Time O - Old progression model that is on the game on the stores at this moment, that takes into account the difficulty of the challenges combined with the duration of the game session;

• Smash Time N - New progression model created during this dissertation that analyses the performance of the player facing the challenge combined with the control of the variety of the generated content.

The playtest started with a brief presentation of the testbed game, without mentioning what going to be tested, after it was said to the participants they had a questionnaire to answer in the end, again without mentioning what was going to be asked, to not influence the gameplay experience and the answers collected in the questionnaires. After the presentation, the participants played the first three campaign levels that serve as game tutorials to present the game and its content and to teach the players how the gameplay mechanics work, during this time the participants were able to ask any questions if they were not understanding something. Following this initial contact with the game, through the tutorial levels, we made a short demonstration of the arena level, focusing on the time component, on how it works and could the players extend the time to achieve better scores. After the arena game mode demonstration, the participants had the opportunity to ask any remaining questions before starting the real playtest. There was no minimum nor maximum number of arena runs for the playtest, so that the participants could play as much as they wanted and during the time they wanted. During this phase of the playtest there was no intervention by the observer, except when requested by the participants. When the participants decided to stop playing, a Game Experience Questionnaire was given to them, followed by an unstructured interview in order to gather some additional feedback from the participants. After finishing the participants part in the playtest, the observer then had to collect the game data that was automatically tracked and registered by the game, relative to: the total arena gameplay duration; the number of arena starts; the number of times the Hero was attacked by an enemy; the number of times the session ended with time out; and the number of times the user quit an arena run and fill the Game Data Collected Questionnaire. The Game Experience Questionnaire will try to answer if the new progression model changed the gameplay experience and the Game Data Collected Questionnaire will try to show if the players spent more time playing the game or not, and if they play more times than with the previous progression model. The user tests group was composed by 32 people, 16 for each arena version with the respective progression models

Quantitative Evaluation Results

Regarding the Mann-Whitney U statistical test on the above mentioned variables, there was statistical significance on the following variables: *Play Test Duration*; *Arena Start Count*, and *Time Out Count*, in all cases with $p < 0.001$, which means there is a clear difference in both versions, whereas the new arena version with the developed progression model on this dissertation got better results that the old arena version.

Quantitative Evaluation Results

Firstly, we will do a macro evaluation composed by 7 components, each representing a group of variables from the original Game Experience Questionnaire. This variables and respective analysis were based on the scoring guidelines for the Game Experience Questionnaire Core Module [11]. From these 7 aggregating variables, the variables *Sensory And Imaginative Immersion* and *Flow* follow normal

distributions on the Shapiro-Wilk normality test. Also, on the Mann-Whitney U test, none of these variables have shown statistical significance. Except in the case of the variables *Felt Could Explore* (Item: I felt that I could explore things.) and *Put Lot Effort* (Item: I had to put a lot of effort into it), none of the other variables passed the Shapiro-Wilk normality test. Regarding the non-parametric Mann-Whitney U tests, the variables *Was Tiresome* (Item: I found it tiresome.) and *Was Impressive* (Item: I found it impressive.) have statistical significance with $p < 0.05$. These results suggest the participants felt the new arena version was more impressive but also more tiring in comparison with the old arena version.

Final Evaluation - Summary

According to the evaluations done to the developed progression model and the results that we got from those evaluations, we believe that the approach to model the player skill progression was a good approach and has a great potential to be used in a future update to the testbed game, as well as to be applied to other endless single player video games. Players did play the game more times, effectively, and for longer periods of time, which was the two main objectives of our work. The feedback received through the questionnaires, answered after the playtest sessions, also suggest that players found this approach more impressive and challenging than the previous progression model version that just increased the difficult of the game, through preset rules, based on the game designer intuition combined with the game session duration. Therefore, and to complete the analysis of the evaluation of the potential and success of our approach, we believe that the approach to develop a progression model that takes into account the real player skill and the content variety, was successfully implemented into a robust and dynamic skill-based progression model.

CONCLUSION

In this dissertation, we presented a skill-based progression model for endless single player video games based on two main concepts: player performance and content variety. While implementing this progression model, we focused on addressing the main research objective of this dissertation, more specifically: creating more challenging and engaging gameplay experiences, increasing the duration of the game sessions on endless single player video games. To achieve this goal, we have used Smash Time, a mobile smasher videogame, as a testbed game for our experiments. Given that the main reason videogames are created is to give enjoyable and challenging gameplay experiences to the players, we focused in the player role while playing a game since the beginning of this study. To that extent, we decided to create a player model to understand better the player, more specifically, a player model that would assess the player skill and its evolution as he/she progresses in the game. The other focus of attention was the variety of the generated game content, that should present new and varied content at an appropriate pace, letting the player recognize some patterns while at the same time, be constantly presenting new and mixed content. Starting from these premises, we used the concepts of Challenges, Obstacles and Tags to develop the three main components of the developed progression model: the Player Performance Model; the Content Variety Model; and the Content Generation Model. An experiment, with both novice and experienced players to the testbed game, was conducted to find the best bootstrap player performance values to start the progression model to adapt the game content. After finding a good starting point to the Player Performance Model, both quantitative and qualitative experiments were done through playtest sessions with real players. In the end of the playtest session, each player answered a game experience questionnaire. The data collected from the questionnaires was later combined with game data automatically tracked and collected by the game while the playtesters were trying the testbed game with the developed progression model. The main objectives of this evaluation were to test if the progression model is able to adapt itself to any type of player providing enjoyable and challenging gameplay experiences every time a player starts a new game session and to confirm if the progression model is able to increase the number of times the game is played as well as to increase the duration of each game session. As the results suggest, we were able to successfully increase, both the duration of each game session and the number of starts of a new game run. We believe to have managed to take the players faster to their gameplay flow channel, in order to make the game interesting, fun and challenging for longer periods of time, by preventing the players to get through a phase of an easy difficulty level in the beginning of every game run, that eventually will make the game boring in the beginning. Besides that, by allowing the players to stay in their gameplay flow channel for longer periods, we also believe to have prevented the game to become unsustainably difficult after a while and, thus, becoming, at some point, frustrating to the players. By not using preset game flow rules, that lead to linear gameplay experiences, like many games do and like the previous progression model did, we believe to have managed to create more dynamic gameplay experiences that, ultimately, also increase the replayability and life time of the game. We believe this is a good progression model that allows to reduce the game designer amount of work, such as to define a dependency flow chart to guide the PCG, as it happens in some games, or to define the probabilities of specific content with a difficulty level associated in relation to the duration of the game session as it was happening in the previous progression model that is currently being used in the official version of the testbed game and in many other endless games. Finally, we believe to have developed a promising skill-based progression model that procedurally generates game content based on the player skill and content variety. An appropriate estimator of player skill and a robust progression model were designed, implemented and tested in this dissertation. Hence, we argue that the presented approach to model progression in an endless single player video game, has a great potential to be applied successfully to other endless games and even to

video games in general. This progression model is data persistent and can be used in the normal levels (not endless) of the campaign mode of a videogame since the player performance model may be carried forward as the player progresses through the game. In order to generalize the progression model to other videogames, the Challenges, Obstacles and Tags concepts must be adapted to the specific context of the video game content. Lastly and, once the player skill and performance evaluation were quantitatively modelled in the Player Performance Model, one just needs to adapt the performance measurement method to what is most suitable to the game in which is being implemented. Additionally, the progression model proposed showed promising results that also demonstrate the potential for extensibility to other games.

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. C. Martinho, P. Santos, and R. Prada, Design e Desenvolvimento de Jogos, ch. 4. Lisboa, Portugal: FCA, 2014
5. J. Chen, "Flow in games (and everything else)," in Communications of the ACM, vol. 50, No. 4, pp. 31–34, ACM, 2007.
6. 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.
7. D. Cook, "The chemistry of game design." https://www.gamasutra.com/view/feature/129948/the_chemistry_of_game_design.php, July 2007.
8. M. Csikszentmihalyi, Flow: The Psychology of Optimal Experience. Harper & Row, 1990.
9. P. Pereira, "Modelling progression in video games." MSc Thesis, Instituto Superior Tecnico, Univer- ´ sity of Lisbon, 2016.
10. 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
11. W. IJsselsteijn, Y. Kort, and K. Poels, "The game experience questionnaire." https://pure.tue.nl/ws/files/21666907/Game_Experience_Questionnaire_English.pdf, 2013.