Holiday Knight: a Videogame with Skill-based Challenge Generation

João Pardal
Instituto Superior Técnico
Lisboa, Portugal
jflpardal@gmail.com

Supervisor: Carlos Martinho
Instituto Superior Técnico
Lisboa, Portugal
carlos.martinho@tecnico.ulisboa.pt

***Abstract*—Challenges in video games tend to be created in a 'one challenge for all players' fashion, which creates different experiences for different players given that they do not all possess the same skills required to overcome said challenges. Some games offer the choice between a few difficulty settings like the well known easy, normal and hard format.**

But what if instead of one difficulty for everyone or making the player choose the difficulty he wants, the game could adjust its challenges to suit each player in a way that would make the experience felt by all players similar?

Based on a previous work [1] that proved that, for the game used, such a model increased both time spent playing and number of times the game was replayed, when comparing an adaptive model with a conventional way of creating challenges, the work presented in this document investigated how the adaptation of the challenges affected the experience of the player where both versions of the game used the same method to create the challenges, but where one of them would adapt the challenges to the player and the other would not.

The conclusions were that there was no significant difference in time played nor in how many times the game was replayed, but the challenge felt by the players of the adaptive version was more homogeneous, meaning the challenge felt by testers was very similar, and that no loss of competence was felt by these players when compared to the static version.

***Index Terms*—Procedural Content Generation; Player Skill Monitoring; Engagement; Adaptive Challenges**

I. Introduction

Nowadays most of the digital content a person consumes is somewhat adapted to them. Whether it is the music we hear on Spotify [1], the videos we watch on Youtube [2], the shows we watch on Netflix [3] or video games on Steam [4], while using these services, they gather data relating to what content is engaging for each user, and with it they predict content that he would like to consume, with the objective of showing the most engaging content to each individual user, hoping that that will make people satisfied with the service they are being provided, making them use the service often.

The premise is: if we know what content is engaging to a user, we can adapt the content we show them to maximize satisfaction on the service we are providing.

[1]https://www.spotify.com
[2]https://www.youtube.com
[3]https://www.netflix.com
[4]https://store.steampowered.com

Video games being a type of digital content as well, much like the services stated previously, aim to give their users (players) an engaging experience. At their very core, video games can be seen as a combination of tasks or challenges the game tells the player he can/must do to progress. So if engagement is to be maximized on video games, the tasks they give to the player are the feature to adapt to each player's taste.

As Csikszentmihalyi [2] observed, how engaged someone is while performing a task is related to 2 variables: (1) the difficulty of the task and (2) the skill the person has to perform it.

While some people can overcome a certain challenge easily, if another person is asked to overcome the exact same challenge they might have more trouble overcoming it. Whether they take longer to do it, if they can even complete it or how they will perform it varies from person to person.

It then seems safe to assume that in order to make a video game as engaging as possible, for each individual player there needs to be a way to measure the skill of the player on a given task or challenge, and adapt, within reason, its difficulty to the player's skills.

II. Related Work

In this chapter concepts necessary for developing this work like the *Cognitive Flow* state and *Procedural Content Generation* will be introduced, along with a work developed to create a progression model that is dependent on the player's performance. There will be a brief discussion after each concept to explain why they are important to this work.

A. Measuring Engagement - Cognitive Flow

Cognitive Flow is the mental state achieved while performing a task that is characterized by [3]: (1) extreme focus on a task, (2) a sense of active control, (3) merging of action and awareness, (4) loss of self-awareness, (5) distortion of the experience of time and (6) the experience of the task being the only necessary justification for continuing it.

According to Csikszentmihalyi [4], how engaged a person is while doing a task can be measured by relating 2 dimensions: (1) the challenge of a task and (2) the skill of the person performing it. If the difficulty of the task is too high when compared to the skill the person has to complete it, the person

will feel anxious, on the other hand, if the skill the person has to complete the task is too high, the person will feel bored.

In either case, this might lead to the person quitting the task, so, to keep the person engaged it is necessary that the difficulty of the task is more or less the same as the skill the person has to complete it.

This work's objective is to keep the player engaged to play. Understanding cognitive flow is important because it makes clear the two concepts that affect the engagement of the player while playing: the challenge (or difficulty) of a task, and the skill of the player in performing said task, along with how they should be balanced to keep the player engaged.

B. Procedural Content Generation (PCG)

According to Togelius et al [5], PCG is the algorithmic creation of game content with limited or indirect user input, it refers to computer software that can create content on its own, or together with human players or designers.

PCG methods are developed and used for a number of different reasons in games, including saving development time and costs, increasing replayability, allowing for adaptive games, assisting designers and studying creativity and game design.

According to Yannakakis et al [6], in order to generate the same gameplay experience, very different game content will be needed, depending on the players skills, preferences and emotional profile [7]. Experience Driven Procedural Content Generation (EDPCG) is a subtype of PCG that focuses on the quality of the content generated, linked to the experiences of the player, where it searches through the available content and generates the one that will optimize the experience for the player. Its components are: (1) Player Experience Model: player experience is modeled as a function of game content and player, (2) Content Quality: the quality of the generated content is assessed and linked to the modeled experience of the player, (3) Content Representation: content is represent to maximize efficacy of the generator and (4) Content Generator: the generator searches the content space for content that optimizes the experience for the player according to the model.

This article starts by justifying that the growing need to adapt the content of a video game to the player arises from the diversity of people playing games compared to a few years ago, and in order to create the same gameplay experience for everyone, there is a need to adapt the content of the game for each individual's skill, preferences and emotional profile. This is what this work hopes to accomplish, trying to provide the same gameplay experience to everyone that plays the game by adapting its content (in this work's case, the challenges) to the player. After this, the authors give a blueprint of the architecture necessary to create a EDPCG solution that is based on the player, with all the different components needed to achieve this.

C. Progression Model

Catarino et al. [1] proposed a skill-based progression model based on EDPCG, for video game *Smash Time* [8], where the skill that the player is displaying throughout the session is used to choose the best challenge in a pool of possible challenges with the objective of creating a more engaging experience for each player. He divides the architecture of his solutions into the following 3 big models: (1) player performance model, (2) content variety model and (3) content generation model, along with 3 concepts used by these models: (1) challenges, (2) obstacles and (3) tags. All of these will be explained in section IV, since these concepts are used in this work as well.

Comparing these models/concepts with the architecture presented by Yannakakis, in Catarino's work, the *Player Experience Model* is implemented via the performance and variety functions, that will model the player experience as a function of game content and what the player is doing.

The *Content Quality* is the information stored in *Player Performance Model* and *Content Variety Model*, that will measure how good the generated content is, comparing it to what we want the experience of the player to be.

The *Content Representation* is implemented using *Tags* and *Challenges*, in order to maximize the performance of the generator.

Lastly, the *Content Generator* is the entire architecture of the work, but the final result comes from the implementation of the *Content Generator Model*.

This work will serve as a base to the work presented in this document. The work described in this document will try to understand how impactful it is to adapt the challenges of the game by comparing two versions of the same algorithm, one where there is adaptation and one where there is not.

III. Testbed Game

Holiday Knight was developed from scratch for this work. It is a game inspired in Enter the Gungeon [9] and Binding of Isaac [10], where the main character fights a number of enemies by shooting at them while at the same time dodging their shots, also known as *Bullet Hell* or *Shoot'em Up* Games.

The game consists in several rooms with enemies, in order to go to the next room the player needs to clear all enemies in the current room.

The objective is to clear twenty rooms before the character's life reaches zero. If the character's life reaches zero, the game resets, meaning he is put back in the first room and the progress of cleared rooms is set to zero.

A. Enemies

Enemies are a challenge's unit in the game, the player succeeds or fails the game based on his performance against enemies. Enemies are characterized by:

- Their sprite and animations: the art that visually represents them, fig. 2, and the animations that controls their visual aspect;
- Their stats: how many health points they have, how close to the player they get and their size;
- Their shooting pattern or behaviour: each enemy has a distinct way of shooting bullets.

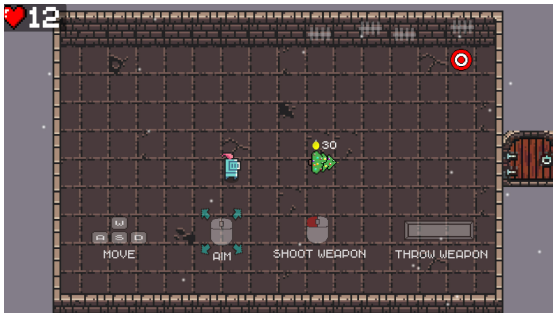

Fig. 1. First room of the game where the controls are layed out, as well as the actions the character can do.

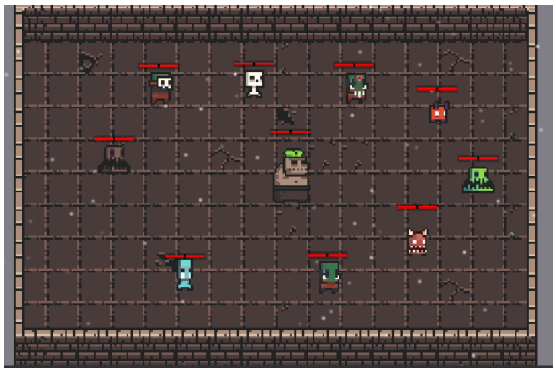

Fig. 2. The ten default enemies in the game.

IV. Implementation

This model was developed with the intention of creating a more engaging experience with the intention of making players have longer playing sessions and or these sessions being more enjoyable. To accomplish this, it is necessary to measure how well the player is doing against the different challenges the game has and give that performance a value so that it is possible to compare the performance of the player on the different challenges.

Additionally, repetition of challenges is undesired, since trying to overcome the same challenges can become boring, so asides from the performance of the player against the challenges, the novelty of a challenge will be taken into consideration as well when choosing the best challenge to present to the player.

Given this, we can define three components that will make the model:

1) Content Generation Model, section IV-E;
2) Player Performance Model, section IV-F;
3) Content Novelty Model, section IV-G;

A. Progression Model Cycle

The cycle for the progression model is as follows, these steps will be explained in further detail throughout this section:

1) Generate a new challenge to present to the player, using the *content generation model*;
2) Update the *content novelty model* for the obstacles in the challenge;
3) Record the performance of the player against each obstacle of that challenge;
4) Update the *player performance model* with the performance values of each of the obstacles overcame;
5) Repeat from step 1.

B. Challenges

Challenges in the context of this work are characterized by:

- A number of obstacles, between 2 and 4;
- The types of each obstacle, explained next;
- Its *novelty* value;
- The player's *predicted performance* value against this challenge;

All of the above characteristics are defined by the progression model and a challenge is considered overcome when all the obstacles in it are overcame.

C. Obstacles

Obstacles are the singular entities that carry meaning to the model and they are domain (game) based. The obstacles for this game are each enemy, section III-A, that will be part of a challenge that the player needs to overcome.

To overcome an obstacle in the testbed game, the player needs to reduce its life to zero.

The way the obstacles are given meaning to the model is by adding a set of tags that characterize them. Tags are described next.

D. Tags

For the model to work, there needs to be a way of associating meaningful characteristics of an obstacle to a performance and a novelty value.

The way this is done is to associate tags to obstacles that have the characteristics that we want to base the performance and novelty on. For example, if the game has an enemy that has high attack stats, can only attack with a sword and moves slowly, we can give it tags such has {*highDamage, meleeRange, slowMovement*}.

With this, the model supports different enemies having commons tags, one such example is fig. 3, that will be used throughout this section to illustrate how different parts of the model work.

In this specific work, each enemy is only characterized by a tag with its name, rather than its characteristics. After overcoming said enemy, each of these tags will be given a performance value. More on this in section IV-F.

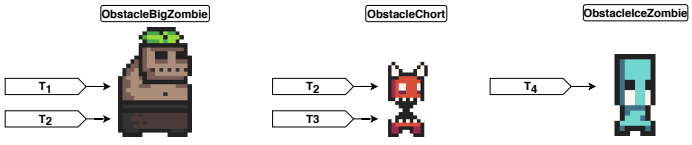

Fig. 3. Example of tags for different obstacles.

E. Content Generation Model

This component is the one responsible for connecting the *player performance model*, *content novelty model* and the game content.

Every time the player overcomes a challenge, this is the component responsible for triggering the mechanisms necessary for generating the next possible challenges, and picking the best one to present to the player.

To start, it will generate a population of 50 random challenges that represent the pool of possible challenges that the model will choose the best solution from to present to the player. After this generation the work of this component is cyclical and its steps are as follows:

1) Of the information stated in section IV-B, that comprises a challenge, this component starts by attributing, to one possible challenge, a *number of obstacles* and the *type of each obstacle* at random.
   After this, the *predicted performance* value, see section IV-F3, and *novelty* value, see section IV-G1, are calculated for this possible challenge. These values need to be compared to some kind of intended values for the next challenge to be generated, so that the model knows how useful this possible challenge is.
   So, the *content generation model* asks for the *intended performance* and *intended novelty* values for the next challenge to be generated, more on these values in section IV-F4 and section IV-G2 respectively, and calculates how close the predicted values are to the intended ones, as stated in eq. (1) for performance, and in eq. (2) for novelty. *Performance proximity* ranges from 0 to 40 (this value is explained in section V-A3) while *novelty proximity* values range from 0 to 100.

$$performanceProximity = |predictedPerformanceForPossibleChallenge - intendedPerformanceForNextChallenge| \quad (1)$$

$$noveltyProximity = |noveltyForPossibleChallenge - intendedNoveltyForNextChallenge| \quad (2)$$

   At this point the model knows the proximity of this possible challenge to the intended challenge, but it still needs to take into account the weight given to each of these parameters. These weights are set by the game designer, and range from 0 to 100. With this, the *total utility* of a challenge is given by eq. (3).

$$totalUtilityIn\% = 100 - performanceProximity * performanceWeight + noveltyProximity * (100 - performanceWeight) \quad (3)$$

   This value represents the accuracy, between 0 and 100%, of how good this possible challenge is, when compared to what the designer intended the next challenge's performance and novelty to be;
2) After calculating the *total utility* for a challenge, this value is compared to the best possible challenge's *total utility* the *content generation model* found so far. If the *total utility* of this challenge is higher than that of the best possible challenge so far, it becomes the best;
3) Steps 1 and 2 are repeated until all possible challenges have been evaluated;
4) When all possible solutions have been evaluated, the *content generation model* holds the best challenge;
5) The *content novelty model* is updated so that the tags for all obstacles in the best challenge have their counts updated;
6) The best possible challenge is created in the game as the next challenge for the player to overcome.

F. Player Performance Model

This model serves four purposes: (1) Storing performance values for each tag, (2) analyzing the player performance for each challenge that is generated, (3) predicting the player's performance against a possible challenge and (4) defining and informing *content generation model* what the intended performance for the next challenge is.

1) Heuristic for Performance: The success of the overall work presented in this document is strongly impacted by the choice of the heuristic that is chosen to represent performance, and this is the part of the model that is most domain based. This describes what variables are being taken into consideration to calculate performance, the metrics that represent how the player's skill will be measured in the model.

In this work the decision on which heuristic to use was based on being as simple and domain free as possible, so the choice was to use time to overcome each obstacle in a challenge as the heuristic. The reasoning for this was two folded: (1) in this work part of the focus is to see if the model works for the testbed game genre, it is not to find the best heuristic, and (2), to try to create a model as generic as possible for this work, so that the results are not domain based, and that people can apply this model as a baseline in other game genres.

The time only starts counting, for each obstacle, after the player hit him the first time, and the time stops being counted after the obstacles was overcame.

2) Player Performance Analyzer: This is the component responsible for giving a performance value to all the tags for all the obstacles in a challenge, after the player overcame it.

For this, this component needs information related to the *tag performance history*, that stores the last 10 performance values of the player for each tag, and *tag performance average*, that is a single value that represents the average of the data in *tag performance history*, and it is the value used to calculate the *predicted performance* value in section IV-F3.

As described before the heuristic being used to calculate the performance is the time it takes the player to overcome each obstacle. The higher the time, the worst the performance against said obstacle. After the challenge is overcome, all obstacles on it have an associated time it took to clear them, as shown in fig. 4.

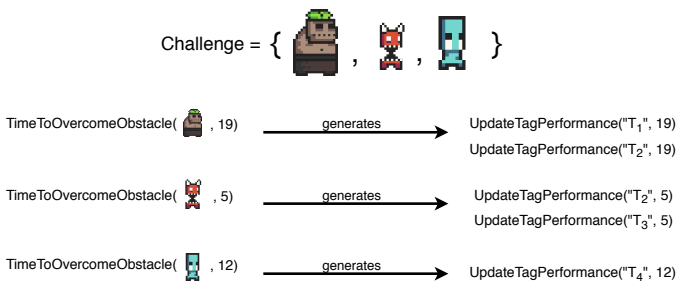

Fig. 4. Time it took to overcome each of the obstacles in this challenge.

Then, the analyzer is responsible for updating the *tag performance* values for all tags in the challenge's obstacles in the following way, example in fig. 5:

- The time to beat an obstacle is added to the *tag performance history* of the obstacle's tag(s), in the form of a pair *<tag(s) for the obstacle, time to beat obstacle>*;
- The *tag average performance* is updated to all tags that had their *tag performance history* updated;

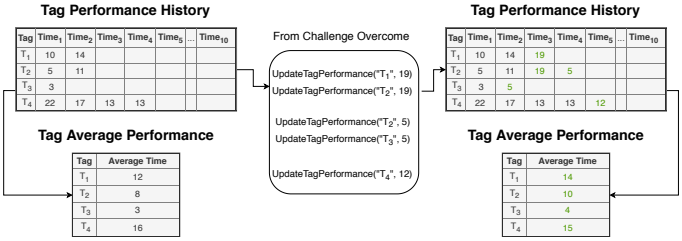

Fig. 5. Player performance analyzer updating performance data after a challenge has been overcome.

When the game is initialized the performance values associated with each tag are bootstrap values that were calculated empirically, so that the model starts with some idea of the expected performance against each tag.

3) Player Performance Predictor: In order to know how good or bad a possible challenge is, it is necessary to have an estimate on the player's performance for that possible challenge so that it can be compared with the intended performance for the next challenge.

This component is responsible for calculating the *predicted performance* value for a given challenge. For each possible challenge generated, this component goes through the tags of the obstacles that compose the challenge, calculates its *predicted performance* value and informs the content generation model of said value.

The values used to calculate the *predicted performance* value for a challenge are : (1) how many obstacles the challenge has and (2) the average of all obstacles' in that challenge *tag performance average*:

$$\begin{aligned} predictedPerformanceIn\% = \\ avgPredictedPerformanceForObstaclesInChallenge \\ * performanceWeight \\ + numberOfObstaclesAs\% \\ * (100 - performanceWeight) \end{aligned} \tag{4}$$

where *numberOfObstaclesAs%* is given the following formula, where *Max#Obstacles* means maximum amount of obstacles a challenge can have and *Min#Obstacles* the minimum amount, in the case of this model, these values are 4 and 2 respectively:

$$\begin{aligned} numberOfObstaclesAs\% = \\ \frac{numberOfObstaclesInThisChallenge - Min\#Obstacles}{Max\#Obstacles - Min\#Obstacles} * 100 \end{aligned} \tag{5}$$

where *performanceWeight* is a value between 0 and 100 that determines how much each of the two values will influence the overall predicted performance for a challenge. In this work, the value used for *performanceWeight* was 65.

As mentioned in section IV-E, the *predicted performance* has to be a value between 0 and 100 so that it is comparable to the *novelty*, and in a more generic sense, to create an abstraction between the heuristic value used to calculate performance and the rest of the model.

For this the player performance analyzer will transform the time value in a comparative performance value, where 0 will be associated with the tag the player took the longest to overcome, and 100 to the tag that he took the shortest amount of time, with a formula similar to that of eq. (5), where the maximum and minimum number of obstacles are changed with maximum and minimum times, and the output is the performance for a tag in percentage. It is important to remember that this performance value is supposed to be a value that indicates how well the player will do against this specific tag, 100 means the player has been having the best performances against obstacles with this tag, and 0 means the player has been having the worst performances against obstacles that have this tag.

When the challenge is complete, the *tag performance average* is recalculated for all obstacles, transforming this time into percentages:

- The maximum and minimum time to beat any tag is recorded throughout this part, storing the tag that had the highest and the lowest time to beat, by default maximum value is set to 100, and minimum to 0. It is extremely important to refer that this maximum is refereeing to the performance, which is inverse to the time, so maximum performance refers to the minimum time, and minimum performance refers to maximum time;;

- For each tag, *tag performance average* is calculated, using all values in the *tag performance history* for that tag, and the amount of times that tag was used;
- Then, the *tag performance average* is compared to the minimum and maximum times, and if the average is bigger than the minimum, the average is now the minimum, if the average is smaller than the maximum, it is now the new maximum;
- Lastly, after these averages have been calculated for all tags, the model associates a performance of 0% to the maximum value, 100% to the minimum value, and calculates all other performance values in between by linearly interpolating between these two values like in eq. (5).

This value is then returned as the *predicted performance* value for that possible challenge.

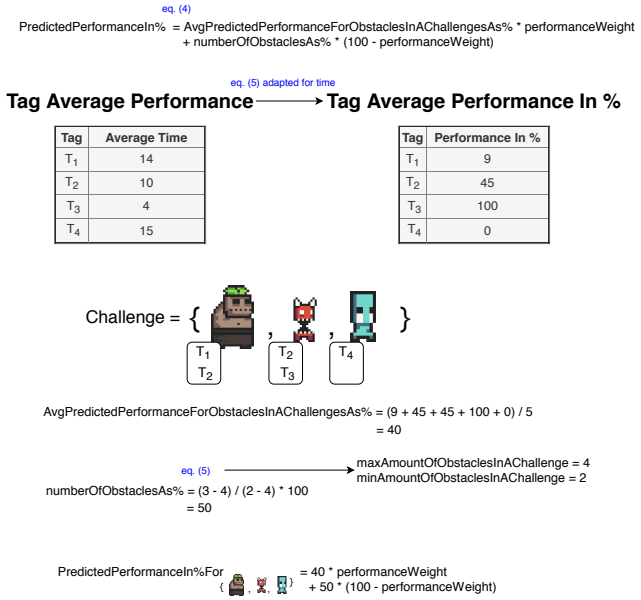

Fig. 6. Example of how *predictedPerformanceIn%* is calculated for a challenge.

4) Intended Performance Function: This function is defined by the game designer and it represents how the designer intends the performance of the player to vary throughout the game. The output of this function is influenced by the amount of challenges the player has overcome.

G. Content Novelty Model

This model serves three purposes: (1) storing novelty values for each tag, (2) calculating the novelty of a possible challenge (3) defining and informing *content generation model* of what the intended novelty for the next challenge is.

When the *content generation model* chooses the next challenge to be presented to the player, the novelty values for the tags on the obstacles of that challenge are updated by incrementing their *tag usage count*. But, as in *predicted performance* value, *tag usage count* needs to be a value between 0 and 100, so that it is comparable to the *performance* of a possible challenge, and in a more generic sense, to create an abstraction between the way the novelty value is calculated and the rest of the model.

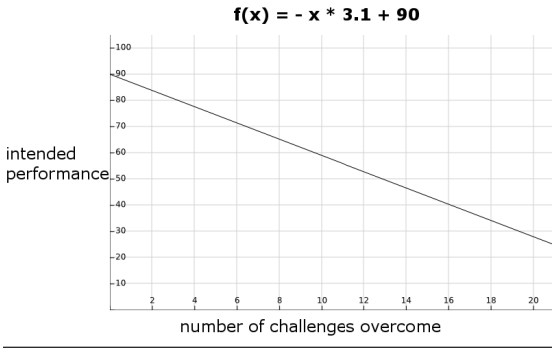

Fig. 7. Intended performance function used.

To store this information, a *novelty percentage* value is created, storing the current novelty value for each tag, each with a value ranging from 0 to 100, where 0 is attributed to the most used tags, and 100 to the least used one in the challenges presented to the player. All other *novelty percentage* values are linearly interpolated between these two values using eq. (6), and an example can be seen in fig. 8:

$$noveltyForTagTin\% = 100 - \frac{usageCountForTagT - leastUsedTagCount}{mostUsedTagCount - leastUsedTagCount} * 100 \quad (6)$$

Tag novelty values are updated every time a challenge is presented to the player.

Tag	Usage Count
T_A	2
T_B	11
T_C	3
T_D	5
T_E	9

Tag	Novelty in %
T_A	100
T_B	0
T_C	89
T_D	67
T_E	22

Fig. 8. Converting *tag usage count* to novelty percentage.

1) Content Novelty Calculator: In order to know how good or bad a possible challenge is, it is necessary to know the novelty for that possible challenge, so that it can be compared with the intended novelty for the next challenge. This component is responsible for calculating the novelty of a possible challenge. To do this, for each possible challenge generated, this component goes through the tags of the obstacles that compose the challenge, it averages their *novelty percentage* values, and informs the content generation model of said value.

2) Intended Novelty Function: This function is defined by the game designer and it represents how the designer intends the novelty of the challenges to vary throughout the game. The output of this function is influenced by the amount

of challenges the player has overcome. For this work, this function was constant and equal to 85.

V. Evaluation

In this section the 2 evaluation phases of this work are described: (1) the preliminary evaluation, where the main objective was to gather data from players, relating to the obstacles of the game, to be used in the model described in the document, and (2) the final evaluation where the main objective was to test if the model implemented supported our hypothesis.

A. Preliminary Evaluation

1) Objectives: The preliminary evaluation served 2 purposes:

(1) get bootstrap values for each tag's performance: initial values for each tag's performance are needed, bootstrap values that will be used when the player first plays the game and the model does not have data about the player's performance, being progressively replaced while he plays and (2) get a performance value to be attributed to a tag when the player dies alongside with finding a maximum value for the time the player can take to overcome an enemy.

2) Procedure: To get the bootstrap values for each tag's performance, a preliminary evaluation was made with 6 people, where the model was adjusted so that the best possible solution for each room was the one with the highest novelty utility, ignoring performance, to make sure the biggest amount of data for each tag was recorded the limit of storing 10 values per tag was lifted.

3) Results and Changes: To calculate the bootstrap values, the performance data for each tag was averaged and that value was assigned to the corresponding tag.

As to the question: what value should be given to the tag of an enemy when the player doesn't overcome a challenge and that enemy is still alive?, using the data collected from these tests, the 95th percentile of each tag was calculated, to mitigate the effect of outliers, and the highest value of all tags was the one used, in this case 40. This means that every time the player doesn't overcome a challenge, all enemies' that are alive in the room get a tag update with value *time to beat obstacle*, section IV-F2, of 40, where in average, during this evaluation, these value ranged from 8 to 19 seconds, depending on the enemy.

There is no difference, performance wise, to the player failing to overcome a challenge and taking a long time to overcome one.

It is important to note that the data extracted from this evaluation was not changed to accommodate this rule, meaning that all values that were above the aforementioned threshold stayed that way, this threshold was only imposed in the final tests.

Another important remark is that out of the 809 saved performance values across this evaluation, 54 occurrences where higher or equal to the previously established value of 40 (corresponding to 6,7% of the total values).

B. Final Evaluation

1) Objectives: Part of the contribution of this work is to understand if by creating more engaging challenges to the player (1) he plays the game for longer periods and (2) if his enjoyment of the time spent playing is greater and to understand the impact of that adaptation on the player's experience.

To do so this evaluation aimed to gather significant data from people that played the game in a way that would either backup or refute these ideas.

2) Procedure: To test the premise of the players spending more time playing when the challenges are adapted to him, the decision was to create logs that automatically saved the time the tester spent playing the game, as well as the total number of runs, i.e. times the tester played the game and how many rooms he cleared in each run.

The final evaluation was made with 40 people and consisted of 2 phases, where each tester was asked, firstly, to play the game for as long as he wanted, and then to fill a questionnaire, where alongside some questions relating to his video game playing habits, and the specific genre of the testbed game, the Game Experience Questionnaire (GEQ) [11] was integrated, as to evaluate the enjoyment of his time playing, and how his experience was affected by the adaptation of the challenges.

In the first part of the test, two versions of the game were used:

- *with model* : one where the tag's performance values were being updated throughout the test, with the model described in section IV;
- and *without model*: another where even though the challenges were picked using the cycle described in section IV-E as the previous version, the performance for the tags of the obstacles that constitute a challenge are never updated, these values, calculated in the preliminary tests, are the same throughout the game.

20 testers tested each of the versions.

3) Results: Of the variables that were being logged, none of them showed statistically significant differences.

Each of the 7 dimensions of GEQ were calculated based on the scoring guidelines provided by its creators [11], and were then put through the non parametric Mann-Whitney U Test, since each model only had 20 participants, to investigate if any of the dimensions had statistical significance. Of the 7 dimensions, 2 showed statistically significant differences: *Sensory and Imaginative Immersion*: more concretely item 19 of the questionnaire - "*I felt that I could explore things*" where testers felt they could explore more on the version without the model which can be explained by the heuristic used on the version with the model, where performance is attributed based on the time to clear a challenge, and *Challenge* for items 32 - "*I felt time pressure*" and item 33 - "*I had to put a lot of effort into it*", where testers that played the version with the model reported a more homogeneous level of challenge felt, meaning the challenge felt by testers was very similar, while

on the version without the model the challenge felt by players was more disparate.

4) Summary: The two versions used in the final evaluation of this work were based on the same algorithm, minus the steps of updating the player's performance on one of them, so the results of the final evaluation showed the impact of adapting the challenges vs not adapting, using the same model to pick the best solution, where the time played and how many times the game was replayed were not significantly different, but the challenge felt by the testers was more homogeneous in the adaptive version, which supports the initial idea that the same level of difficulty can represent different perceive challenges for different people, which is what this model should accomplish, and even thought the challenge felt was lower for the adaptive version, the competence dimension did not show significant differences.

Another important conclusion came from the fact that testers that played the adaptive version felt they could explore significantly less than the other testers, which can be explained by the heuristic used and might indicate that the intended difficulty function should have a more of a sinusoidal form, rather than the straight line that was used, to let the player have more relaxed challenges where he feels like he can explore new tactics and behaviours in a less challenging environment.

VI. Conclusions

In this document a possible approach to adapting the challenges of a game to the player is presented. Keeping in mind that different players have different skill levels, this work focused on how to create a model that would analyze performance while the player is playing and use that data to pick what the model believes to be the best possible challenge, within a random population, comparing it to functions created by the game designer on how the performance of the player should evolve throughout the game, as well as the novelty of the content.

Holiday Knight, a *Bullet Hell/ Shoot'em Up* game, was created from scratch to serve as the testbed game for this model, where progress is made by killing all enemies in a room to go to the next. The model described in this document was used to choose what enemies to spawn in each room.

By analyzing the performance of the player against said enemies, and using it on the decision of what the next challenge should be, the hypothesis was that players would play for longer periods of time and/or replay the game more. Since this idea had already been tested by Catarino [1], this work was based on his, but applied to a different type of game, and the final evaluation of this work had the additional purpose of understanding how and what dimensions of the experience were affected by the fact that the challenge was adapted to the player.

For this, in the final evaluation of this work, two versions using the same model to generate the challenges of each room were used, where in one of them the player's performance was dynamically updated accordingly to the performance he was showing against each enemy, and on the other version these values were static.

The final results showed no significant differences in the duration of a play session nor on the amount of times the game was played.

Other results showed that the population that played the adaptive version felt a more homogeneous level of challenge than those with the static version, with no loss of competence. Additionally the population that played the adaptive version felt that they could not explore as much as the static version.

On a game designer note, this model was helpful in empirically setting a difficulty for each enemy created without the need of complex formulas. On the preliminary evaluation, done with 6 users, all of the encounter's performance values were being recorded, and after doing some data manipulation, difficulty values were found for each enemy. This way, the game designer could just focus on creating the enemies and the model took care of finding a difficulty value suited for each enemy.

Another important consideration is that with this model there was no need to create individual challenges by hand, all of it was automatized, the model was fed all the information needed about how the progression should be made, performance and novelty wise, and it picked the best challenges based on that.

References

[1] J. Catarino and C. Martinho, "Procedural Progression Model for Smash Time," in *Proceedings of the 1st IEEE International Conference on Games*. IEEE Press, 2019.

[2] M. Csikszentmihalyi, *Flow and the Foundations of Positive Psychology: The Collected Works of Mihaly Csikszentmihalyi*. Springer, 2014.

[3] S. Baron, "Cognitive Flow: The Psychology of Great Game Design," 2012. [Online]. Available: https://www.gamasutra.com/view/feature/166972/cognitiveflowthepsychologyof.php

[4] J. Nakamura and M. Csikszentmihalyi, *The Concept of Flow*. Dordrecht: Springer Netherlands, 2014, pp. 239–263.

[5] J. Togelius, N. Shaker, and M. J. Nelson, "Introduction," in *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*, N. Shaker, J. Togelius, and M. J. Nelson, Eds. Springer, 2016, ch. 1.

[6] G. N. Yannakakis and J. Togelius, "Experience-Driven Procedural Content Generation," *IEEE Transactions on Affective Computing*, vol. 2, no. 3, pp. 147–161, 2011.

[7] C. Bateman and R. Boon, *21st Century Game Design (Game Development Series)*. Charles River Media, Inc., 2005.

[8] "Smash Time!" Bica Studios, Android, iOS, 2016.

[9] "Enter the Gungeon," Dodge Roll, Microsft Windows, 2016.

[10] "The Binding of Isaac," Nikalis Inc., Microsft Windows, 2011.

[11] W. IJsselsteijn, Y. De Kort, and K. Poels, "The Game Experience Questionnaire," *Eindhoven: Technische Universiteit Eindhoven*, 2013.