Player Affective Simulation for Progression Design

Bernardo Brás Lourenço
bernardo.lourenco@ist.utl.pt

Instituto Superior Técnico, Porto Salvo, Portugal

November 2017

Abstract

Procedural content generation is a technique used in many games. But if the game generates too much content this way, it can be difficult to test all possible scenarios with players before launching the game. In these cases, an AI agent is used to test the whole content, but it only performs basic validations and does not gives us the subjective feedback of a player. In this situation, it seems there is no viable way of testing large amounts of generated content. We created a methodology that uses an affective agent, to test a game using the personality and skill of a player to give us some emotional feedback on the playtesting session. The affective agent is a combination of an affective agent architecture and a personality model. We present PLEASED, an implementation of our methodology that was used as an example of our approach. To evaluate how effective is our approach, we designed and developed a game to be evaluated by PLEASED. The effectiveness of our system was measured by comparing feedback from PLEASED with players' feedback. Our results suggest that PLEASED is more suitable to simulate casual" players than hardcore" players and that hardcore" players follow different criteria to evaluate a game when compared to casual ones.
Keywords: procedural content generation, playtesting, player model, affective computing, personality

1. Introduction

Procedural content generation (PCG) has been used as a way of increasing replayability and creating large environments for the players to explore[1]. Nowadays one of the aspects that PCG research focuses on is personalization, this means that this technique is starting to be used to create real time adaptations in games to meet the player needs and preferences. This is called Experience-Driven Procedural Content Generation (EDPCG), where the experience given to the player is a result of the adaption of the game to the player's actions and emotions[2]. The use of PCG is changing, as in the past was mostly used to increase a game's replayability and now is starting to see the light in other forms like EDPCG where the goal is more complex and adaptive.

Nowadays PCG reliant games are mainly tested by either players or AI controllers[3]. Both approaches have advantages and disadvantages. Human players give the best feedback possible as they are the end user of our game but PCG reliant games can have too much content to be tested by some players in a limited time. Additionally game development is an iterative process that may require multiple play testing sessions to be performed and also the game may need to be tested by different types of players, for example casual and hardcore players. We can have all these different types of players testing all our content for each iteration of the development process, but that would be a huge time consuming task. On the other hand we have the AI controllers that do not have this constraint, they can test games with much content in a acceptable amount of time but they do not give us the subjective feedback that is provided by the human emotions. The testing of a PCG or even EDPCG reliant game can be a hard task as in these games we are adding a new layer of complexity, the automatically generated content, this content can not be controlled as well as the manually created one.

In this work, we present a methodology to validate the design quality of the procedurally generated content by using an affective agent. This agent will pass the levels of the game using the personality and skill of the player that we are trying to simulate. In the end, this agent will generate information of all the emotions felt throughout the level. These information can then be used by the game designer to adjust the PCG reliant game to give the player a more pleasant game experience. To validate our approach, we created a system based on our methodology, called PLEASED. PLEASED is a system responsible of managing an affective agent and presenting emotional feedback of the levels tested by the agent.

The remainder of this document is organized as follows: (1) we start by presenting the theorial background of our methodology; (2) then we explain in detail our methodology's conceptual model and describe the technical development of the system PLEASED; (3) present user studies that were conducted to setup our experiment and we also describe the experimental process; (4) we present the results of our experiment (5) and conclude by presenting final remarks and future work.

1.1. Affective agent

Intelligent agents make decisions based on logic only and can not express any type of emotion. On the other hand affective agent use emotional states to process the information they receive as well as in their decision making mechanism. Emotions and personality started being used on intelligent agents so they start having a human-like behaviour. These human-like behaviour agents (affective agents) were used in some cases to create believable characters, for example in Martinho's Emotivector[4] and in other cases to simulate human behaviour, for example FearNot![5] (based on OCC[6]) and EMA projects. Emotion is what allow an intelligent agent to be able to "appear" to act like a human and personality controls what emotions are triggered. These two concepts are fundamental to the simulation of human behaviour. In our work we need to simulate a player passing a level of a game, so we need an agent capable of show a human-like behaviour, an affective agent. To be able to transfer the personality of a player into the affective agent we need a personality model to asses the player's personality and translate it into the affective agent. There is personality models in psychology like Five Factor model[7], Myer Briggs Type Indicator[8] ,Keirsey Temperament Model[9] and Cloninger's Temperament and Character Inventory[10] and also personality model based on players like Bartle player types[11], Demographic Game Design[12], Unified model[13] and Lazzaro's fun types[14]. All the these personality models are possible candidates to be used on our methodology.

In our implementation PLEASED we used the emotivector as agent architecture and focused only on one personality trait optimist/pessimist using the LOT-R[15]. The reasoning behind this decision is that the emotivector takes into account the past events to infer the emotion that is currently being felt, this is a factor that must be taken in account in games when calculating the player's emotions. For example, if a player lost five matches in a row in a game, he is probably now frustrated at least. In this example the history of the player affect his current emotional state. In the emotivector this history of past events is used to generate a expectation and the final emotion is the result of the mismatch between the expectation and reality. As in the Emotivector, the expectation is one of the core parts of the algorithm, we also decided to use the LOT-R as the personality trait that will be used in our system. Being pessimist or optimist affect the expectation of future events. So in our system we will be integrating the LOT-R into our version of the emotivector.

1.2. Emotivector

Martinho's work introduces an anticipatory sensory interface module composed of affective anticipatory mechanisms: the emotivectors[4]. This module is used to give an agent the ability to produce believable and understandable behaviour. In other words it transforms an agent into a believable synthetic character. An agent is composed of sensors, effectors and processing unit. The emotivector's module, called the salience module, fits in this architecture by reading the information that is gathered by the sensors. With this information the salience module will compute the emotions (based in both attention and emotions theories) by using the mismatch between the sensed value and the expected one. This expected value is calculated using the history of sensed values. The emotions are generated based on the reward and punishment system. For example if the sensed value is higher than the expected one, we have a reward better than expected and that can trigger a hope emotion. We can have a better, worst or expected reward and the same for the punishment.

1.3. Life Orientation Test Revised

Life Orientation Test Revised (LOT-R)[15] is a 10-item questionnaire that measure optimism versus pessimism. Of the 10 items, 3 items measure optimism, 3 items measure pessimism, and 4 items serve as fillers. Respondents rate each item on a 5-point scale: 0 = strongly disagree, 1 = disagree, 2 =neutral, 3 = agree, and 4 = strongly agree. The max score of the questionnaire is 24, people with this score are highly optimistic, on the other hand people with 0 score are highly pessimist.

2. Solution

This section presents in detail the methodology that we created to evaluate PCG reliant games. Henceforth, we present (1) our methodology's conceptual model that can potentially evaluate any type of PCG reliant game, (2) and PLEASED an implementation of our conceptual model.

2.1. Conceptual Model

Our methodology (seen in figure 1) core idea is to use an affective agent to pass a game and collect his emotional feedback. To achieve this goal we need to transfer the main traits of a player to the agent so

we can simulate the player. The skill and the personality of the player are two traits that influence his game experience. So our model will use both these traits and will incorporate them in our affective agent. In order for the agent to generate emotions we need an affective agent architecture. On the other hand we also need a model that can translate the personality of a player into the agent. Our model is then composed of a affective agent architecture and, a personality model that work together to produce a affective virtual experience. This experience can be used by a game designer to improve the game.

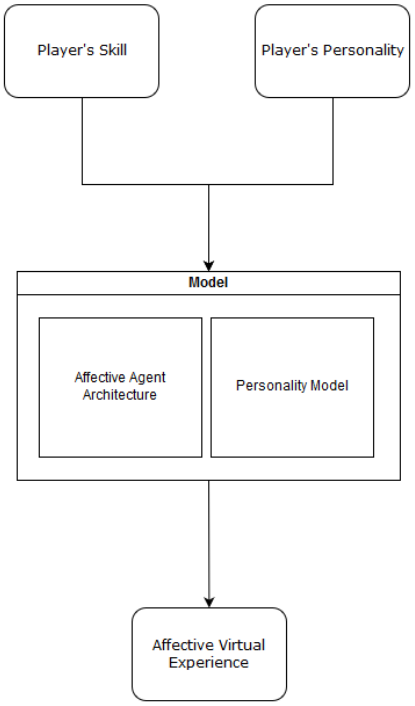

Figure 1: Conceptual Model

2.2. PLEASED

Our system PLEASED is based on the conceptual model where we use the LOT-R as player's personality model and emotivector as the agent architecture. In PLEASED we still use the player's skill as input of the system but now we have two new inputs, the list of challenges and LOT-R score. The player's personality was replaced by the LOT-R Score given by the LOT-R questionnaire. The list of challenges is the level of the game we want to test split by challenges. A challenge in our system is a obstacle that the player must overcome in order continue progressing through the level, for example killing an enemy can be a challenge.

PLEASED is composed of the emotivector and an adaptation done to emotivector to use the LOT-R score in the calculation of the emotion. The system uses a modified version of the emotivector's sensantion calculation method with only six sensations instead of nine that were used on the original emotivector. But our system's affective agent still uses the mismatch between the expectation and reality that is described in the emotivector. The emotivector uses the expectation based on past events as part of the calculation of the emotion. In our system this is also an important variable to be considered in the final emotion calculated but in our case the expectation is influenced by the personality trait. The personality trait that is incorporated in our affective agent is the optimism or pessimism of a player. This personality trait is measured by using the LOT-R explained previously and is used to calculate the expectation of the agent in each part of the level. This trait was chosen as the expectation plays an important role in the emotivector architecture and having an accurate expectation can improve the reliability of our system.

2.2.1 Architecture

PLEASED calculates the emotions for each challenge, a player can feel more than one emotion for each challenge depending on his personality and skill. That said we will start explaining how a emotion is calculated for one single challenge as all the challenges are calculated in the same way but with different histories, as our system take in account what happened in the game before that single challenge. As we saw in the emotivector, the emotion is generated by the mismatch between the expectation and reality. In basic principles, is like this, if we expect something to happen and it didn't happen we feel sad. On the other hand, if we expect something bad and something bad happens, we feel less sad than in the first situation because, we expected it. So expectation and reality are the two core concepts of our system and the more accurately we can model them, the better our system will perform.

Our system's algorithm starts by computing the player's expectation to overcome the challenge. To achieve this goal we must consider the personality trait measured by the LOT-R (optimistic/pessimist) and the history of past attempts to overcome the challenge done by the player. Let's start by explaining how the history of past attempts influence the person's expectation about the challenge. For example if a person starts to play a game that he has never played his expectations of passing the very first obstacle of the game are neutral as he has never done it so he does not know if he can do

it or not. Now if the player tried one time to pass the obstacle and failed, his expectation about that obstacle changed because he now knows that it is easier to fail than to succeed. The player failed a second time, his expectation about passing the challenge has changed again, now he know that is hard to pass it. Finally the player passed the challenge, the expectation stabilized now he knows that if he is not careful as the last try he might fail again. In our system this scenario is modeled by the history of the attempts done by the player in our game. In this history of attempts, each attempt is saved as a number between 0 and 100, where 0 is a completely failed attempt, 50 is a situation where the player passed but with the worst score, and 100 the player passed the challenge flawlessly. Now that we have the attempt history we must convert it into the expectation of the player. We use a window of the last five tries done by the player to calculate the expectation. We use only the last five tries to model the fact that the player is in a learning process, and the first tries that he did in the challenge would not influence his expectation, as for the player they were part of the learning process and don't have much weight in his current expectation. We then take these last five attempts' values and apply a weighted average on them, giving more weight to the more recent attempts. The formula used in this calculation is the following:

$$at_n = attemptscore \tag{1}$$

$$f(x) = attemptweight \tag{2}$$

$$w_x = \frac{f(x)}{100} \tag{3}$$

$$Expectation = \sum_{n=1}^{5} at_n \times w_n \tag{4}$$

The value of each weight varies according to the game we want to evaluate with our system. After applying the formula we have a value between 0 and 100, this value we call expectation skill component.

Next we will introduce the personality trait in the expectation calculation. If a person is pessimist, he expects the worst more often than the optimist person so this trait will affect the expectation of our system. So the expectation skill component is a value between 0 and 100 so we could assume that more than 50 the person is expecting to pass the challenge and less then 50 he is expecting to fail the challenge. But we also have a third situation in the expectation scenarios where a person failed as many times as succeeded, and do not know if is capable of passing the challenge again, he is doubting wherever he will be able to pass it again or not. So we can have three different expectations, the player thinks he can do it, the player is in doubt and the player thinks that can not do it. In our system, these are three different intervals in the 0 to 100 expectation skill component range. So we used the score of optimist/pessimist measured by the LOT-R that is a value between 0 and 24 to help calculate these intervals. We tested the system with the most optimist and the most pessimist players to check how we can incorporate their values of the LOT-R in the system and we came to a conclusion that this value have a linear relation with the expectation intervals. So we used the LOT-R value to calculate the two values from 0 to 100, these are the lower interval X and upper interval Y where if the expectation skill component is less than X, the player expects to fail the challenge, if it between X and Y or equal to them, he has doubts about success of the next try and if it is greater than Y, he expects to pass the challenge. The formulas to calculate the expectations intervals through the score of the LOT-R are the following:

$$ps = personalityscore \tag{5}$$

$$UpperInterval = -\frac{17 \times ps}{15} + \frac{1080}{15} \tag{6}$$

$$LowerInterval = -\frac{17 \times ps}{15} + \frac{931}{15} \tag{7}$$

After expectation the other core concept of our system is the reality, and this component is simulated using the player's skill that is one of the inputs of our system. We use this skill to simulate the player's attempts based on probability, the player's score for each type of challenge is a percentage chance that the player have to succeed in each attempt. To calculate if the player passed the challenge in the current attempt, we use a random number generator that create a number between 0 and 100 and then we check if this number is less or equal to the chance to pass the challenge, if so the system consider that he passed the challenge in this attempt. Otherwise the system consider that he failed to pass the challenge. For example

a player have a skill score of 60 in the challenge then we use the number generator, we get a 40 this means that the player passed the challenge in that attempt. On the other hand if we got a 70 from the number generator, that would mean that the player didn't passed the challenge in that attempt and we must generate another number to simulate the next attempt of the player and so on until the player pass the challenge.

So after we calculated the expectation of the player and also simulated the success of each attempt based on the player's skill, we can now calculate the emotion felt in each attempt of a given challenge. Our model can generate a subset of the the sensations identified in the emotivector, these sensations are: Stronger Reward; Expected Reward; Unexpected Reward; Unexpected Punishment; Expected Punishment; Stronger Punishment. Each one of these emotions is generated from the mismatch between expectation and reality. In our system each combination of expectation and reality is mapped to one of these emotions. So let's start with the unexpected emotions. When the player's expectation is doubt about the success of the next try, any emotion that he might feel is unexpected either he fails or succeeds so if he succeeds it is generated an unexpected reward and when he fails it is generated an unexpected punishment. For example, when a player is starting to play a new game type, like a shooter, anything that he starts doing he has no idea if it is easy or hard and anything that happens he will not expected as he have no experience in that kind of games. Now when a player have the expectation that he is capable of passing the challenge and then he fails, he will feel a stronger punishment emotion, as that is the scenario where the player feel the most negative emotion. For example when a player passed the same challenge over and over again and then he fails that challenge that he consider easy, he can feel that his skill is decreasing. On the other hand if that player pass the challenge that he passed over and over again he will feel a less intense emotion as Expected Reward, as that is what that player is used to. Lastly when a player expects to fail and succeeds, he has a symmetric emotion generation compared to the case that we talked previously. When a player that is expecting to fail, succeeds he will feel the most intense positive emotion the Stronger Reward. For example when a player is trying to pass the final boss of a game and he is failing over and over again, when finally pass the boss he will feel extreme happiness. On the other hand if he fails again in the same boss, he will feel just a bit sad as he knew from previous attempts that this boss is hard, and in that case he would feel an Expected Punishment. Additionally the Expected Punishment is the emotion that may cause the player to fell frustrated and ultimately lead him to give up on the game. As this factor is also an important information that our system can provide to the game designer, we will record the number of times the player felt this emotion before giving up on the game. The emotion generation logic is presented in the following table:

Expectation	Successful Attempt	Failed Attempt
Positive	Expected Reward	Stronger Punish
Neutral	Unexpected Reward	Unexpected Punish
Negative	Stronger Reward	Expected Punish

Table 1: Emotions generated by the difference between the expectation and reality.

After we calculated the emotions for each attempt of the level we use this information to build a graph that shows this data. In this graph we can see for each challenge what emotions were felt and how many times each of them were felt. In x axis of the graph (seen in figure 2), we have the number of the challenges and in y axis we have the number of times the emotion was felt in the challenge. Each color represents an emotion and, we can have multiple emotions in one challenge.

2.2.2 Modes

Our system has two modes that we can use. The main difference between these two modes is the source of the player's skill data that they use. The Player Real Mode uses total information of a gameplay session, for example how many times a player failed in each challenge, to calculate the reality component of our system. The Simulation Mode uses this gameplay session information to calculate the skill of the player and then use this skill in the algorithm as we explained previously.

The Player Real Mode's main objective is to test the emotion modelling module of our system. In this mode we simple use the information of what happened in a gameplay session to generate the emotions graph. This mode is useful to check if our system is failing in emotion modelling department as it does not use the skill calculator to generate the emotion graph. In this mode when we arrive at the calculation of the reality component of our system's algorithm we do not generate a random number to check if the player passed the challenge or not, we already know if the player passed or not in that attempt as this mode receives as input the game log file that contains how many times the player failed the challenge.

The Simulation Mode can be used in PCG reliant games, as a way to test all the generated levels because it uses the skill of the player to simulate the

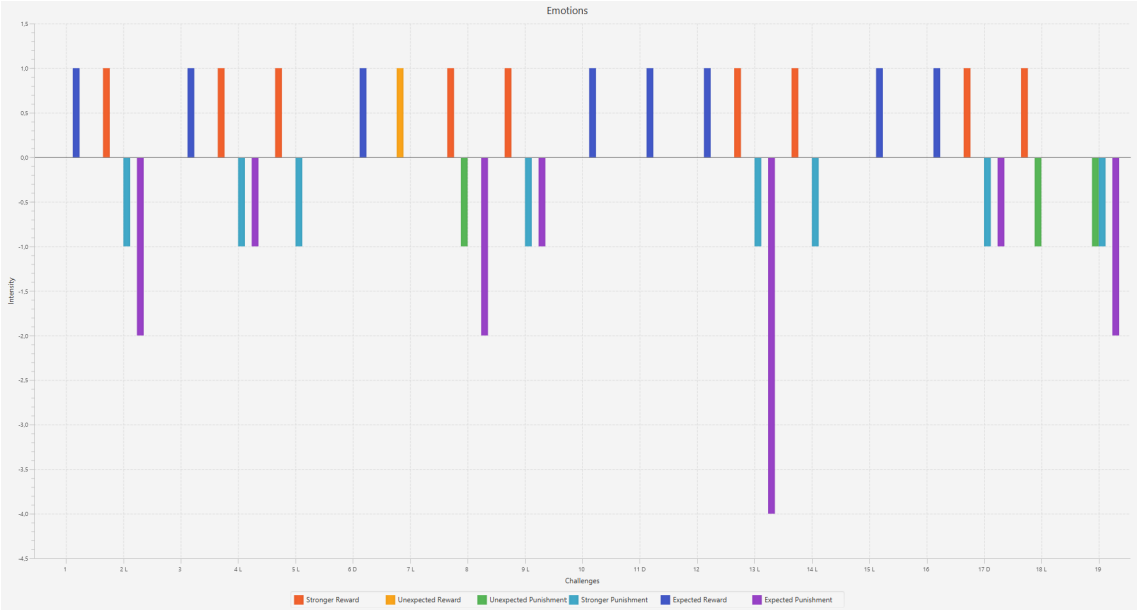

Figure 2: Emotion Graph Example. Emotions from left to right: Red - Stronger Reward, Yellow - Unexpected Reward, Green - Unexpected Punishment, Light Blue - Stronger Punishment, Dark Blue - Expected Reward, Purple - Expected Punishment.

gameplay session, making it possible to test hundreds of generated levels in a short amount of time. Besides that we can change the affective agent's personality traits and test with different personalities the same game, this way we can check what kind of personalities enjoy it the most. Before we can evaluate a game by this method, we must first calculate the skill of the player we are simulating. Given that we can split the game into a sequence of challenges and the same challenge can appear at least three times, we found that three repetition of the same challenge was enough to assert the players skill in that particular challenge by performing tests with players in early stages of the system's development. In these tests we concluded that in average there was an learning curve until the third challenge of the same type then the player's skill would stabilize. Using the information of how many attempts were performed in each of the three challenges of the same type, we can calculate the skill of the player for that challenge type. For this calculation we use a formula (8) that take in account the fact that the player is still learning, and it punish his skill score less for the first failed attempts of each challenge. In (8) f represents the number of failed attempts before the player passed the challenge. If the player passes the challenge at the first attempt we give the player 100 as Challenge Skill for that challenge.

$$ChallengeSkill = \begin{cases} \frac{100}{f}, & \text{if } f \geq 1 \\ 100, & \text{if } f = 0 \end{cases} \quad (8)$$

$$ChallengeTypeSkill = \frac{\sum_{n=1}^{z} ChallengeSkill_n}{z} \quad (9)$$

In (9) z is the number of challenges passed of the same type. After the player passed a level we use the information of the attempts done in each challenge to calculate the Challenge Type Skill for every challenge type in the game.

As this mode is based on probability of the player being successful in each attempt using his skill score, we have to run the same level multiple times to check what emotion were felt most of the time in each challenge. So the Simulation Mode simulate the same level one hundred times so we can have a good perception of the player's emotions.

3. User Study

In this section we present (1) the game that was used to test our system PLEASED and also two studies that were performed in preparation for our experiment. (2) One of the studies was about the game's usability and the other (3) was about the adaptation of our system PLEASED to the game. (4) Lastly we describe the experimental process of our work.

3.1. Game

For this work, we created a game to use as an example in our experiment. This game was created from scratch using Unity™ game engine. Our game is a side-scrolling 2d platformer where the player must overcome challenges like killing enemy, jump-

ing gaps and pass thought a series of spikes. The character have unlimited number of lives and when dies it respawns near the place where it died. The game have two levels.

3.2. Usability Testing

The game's usability test enables the identification of the key problems of our game's experience. When testing a system like PLEASED that evaluate the game by the emotions felt, we need to ensure that our game deliver a good experience in some parts of the level at least so we can detect the difference between bad experiences and good ones on our PLEASED emotion graph. So to ensure that our game was in fact delivering a good player experience, we tested it with players.

Sample: This study was carried out with five players. Two of these player were casual players and three of them were hardcore players. Their ages were between 28 and 31 years old (M = 28.8, SD = 1,30 , 2 female). The test was conducted via Skype with each one of the players individually. Each session lasted between forty minutes and one hour. Three sessions were performed with each one of the players in three different iterations of our game.

Procedure: We started our usability testing sessions by introducing our game. After that, we asked the player to install the game in their computer. When the player started the game we would ask to screen share the game window so we can see his gameplay session. While the player was playing we just watched and answered the player's questions about the game. After they finished playing we asked them what they thought about the game and also what were the best and the worst parts of the game. We only asked this in the end of the game so we did not disturb the gameplay experience.

Identified problems and corresponding solutions: The problem that we had, in overall game experience, was that the player felt that some challenges were too easy and suddenly a challenge with much higher difficulty would appear. The flow zone was one of the factor that we stated as being important to be present in our game. So we transformed the easier challenges in harder ones and smoothed out the difficulty curve between the challenges. We found also that our hardest challenge was not that hard for the majority of the player specially the hardcore players and we wanted at least one challenge to be really hard so we could measure the number of tries, that a player performs before giving up on the game. To fix this issue we made a challenge specially to this situation, and tested it until the players found it hard to pass.

3.3. Parameter Fine-tuning

This study was performed to find the right adaptation of our system PLEASED to our game. The adaptions we need to perform were mainly in the expectation calculation. We needed to be able to classify each attempt with a number between 0 and 100. And we also needed to find the value of the weights of the last five attempt window that our system uses to calculate the expectation. Other adaptation was made to our system that was the mapping of the game's challenges type into our system, we needed this information so we can calculate the player skill and also to calculate the expectation as we need to identify the challenge type that we are currently processing.

Sample: This study was carried out with the same group of people from the previous study plus three new participants as we needed also to test with new players to check if we had different results. Their ages were between 15 and 31 (M = 25,87, SD = 4,82 , 3 female). This time we had 4 hardcore players and 4 casual players. This study was also conducted via Skype with each one of the players individually. Each session lasted between twenty minutes and forty minutes. Two sessions were performed with each player.

Procedure: We started our parameter fine-tuning sessions also by introducing our game. After that, we asked the player to install the game in their computer. After the installation we asked them the itens of the LOT-R and saved the score. When the player started the game we would ask to screen share the game window so we can see his gameplay session. Before each attempt we asked the player what he thought about the next attempt, we gave them 3 options to answer: "I will pass in next attempt" , "I don't know if I will pass" and "I won't pass in the next attempt" after passing 10 challenges the session was over. After each session, we used our system in debug mode with a initial configuration of the parameter we are trying to find and also with the personality score. We stopped our system in each expectation calculation to compare the expectation calculated by our system with the feedback from the player. We adjusted the parameter in each session to be able to calculate the same expectation of the player's feedback. In the next session we would have this configuration as the initial one and repeat the process. When we finished the study we calculated all the sessions again with the final parameters to check if we needed to change them.

Final Parameterization: We wanted to find the parameterization for the weight of each attempt for the expectation calculation, a number between 0 and 100. In this case we had an initial approach that was to measure how far from failure was a suc-

cessfully attempt and how far from success was an failed attempt. To do it we needed to measure for how much distance the player missed the jump. for example. We applied this in the first three session that we performed. and the fourth one without it because we started noticing that it was not affecting the final expectation. as the player was only seeing it in a binary way, success or failure. This feature of our system PLEASED appear to not be able to be applied to platform games. So we used in our system, 100 to represent a successful attempt and 0 to represent failed attempt.

In the case of the parametrization of the weight of the five window history of attempt we found a parameterization that covered 87 per cent of the cases that we collected in the sessions. This parameterization gives more weight to the more recent attempt as we predicted. The parameterization is the following:

$$f(x) = \begin{cases} 5 & : x = 1 \\ 10 & : x = 2 \\ 15 & : x = 3 \\ 30 & : x = 4 \\ 40 & : x = 5 \end{cases} \tag{10}$$

$$w_x = \frac{f(x)}{100} \tag{11}$$

$$Expectation = \sum_{n=1}^{5} at_n \times w_n \tag{12}$$

4. Experimental Process

The first step of our experiment is to answer a questionnaire. This questionnaire is divided in three parts: demographic data, player experience and personality trait information. In the demographic data we collect the age range as well as the gender of the person, with this information we can organize the results in different group of players. After, we gathered some information about the person's playing experience, as we want to know if the person is failing because do not play games or because do not like to play platform games. This is useful information that can help us understand for example the reason of the person give up on the level. Lastly we assess if the person is optimist or pessimist using the LOT-R 10-item questionnaire. In the end the person receive an email with his unique ID that will be used in the game as way of correlating the questionnaire with the gameplay session.

After receiving the email with unique ID, the player can start up the game using this ID and begin playing. At anytime during the gameplay session the player can give up playing the level that he is in. We gave this option to the player using other words as such you can click in the "finish this level" button any time, we did this so the player do not feel demotivated when clicking it. This can be used to discover the point when the player is so much frustration that prefer to stop playing rather than continue playing.

When a player finishes the first level a screen will appear stating that now he must choose the best and worst parts of the level. After that, the level will appear again giving the player the option to classify each part of the level except the tutorial part. In this part, the level is divided by challenges and on top of each challenge, we have a button called "Select" and when the player click it, two icons appear one with a "like" and another with a "dislike", the player can use them to classify the challenge. With this information we can evaluate whether or not passing the challenge of the level had positive impact on the player. After this, the player starts the second level that is similar to the first one but with a different challenge order. Lastly the player also indicate the worst and best parts of the second level.

After the game experience, the player upload the game log files, one for each level. These logs contain the number of failed attempts for each challenge and the best and worst part chosen by the player. We then use these files directly as input of our system PLEASED, and we generate the Player Real Mode graph for the first level and we compare it with the best and worst parts chosen by the player to see if our system detected the those parts. This test is to verify if our system is generating the emotions correctly. Afterwards, we compare the Simulation Mode graph of the first level with the best and worst parts chosen by the player to see if our system detected the those parts. Then we can compare the results of these two tests to check if the Simulation Mode is working properly, because the Real Player Mode is a simple emotion modelling mode, and the Simulation Mode works in the same way but uses the skill of the player as input and uses prediction to generate emotions, so we can see if these two components (skill calculation and prediction) are working properly by comparing the results from these two tests. Finally we compare the Simulation Mode graph for the second level with the best and worst part for the second level, also to test if the system detected them.

5. Results

The experiment was open to anyone to participate during a period of a week and a half. This experiment was sent to the participants by email and also a link to the experiment was shared on Facebook. Fifty nine responses were sent to us, but only twenty four answers were valid. We considered in-

valid answers the ones that either did not send us the game log file or send us an invalid one. From the valid ones, eight completed the two levels, nine completed at least one level and eight gave up on both levels. We can still use the data of the participants that did not completed the levels, as we can consider that they were playing a smaller level.

We will present the result in the following order: first we will show the results of the Player Real Mode applied to the first level of the game; then we will use the skill collected in the first level to predict the first level, using the Simulation Mode on the first level ; and finally, the Simulation Mode on the second level using the skill score collected in the first level. After we present these system modes, we will also present conclusions as when a player is willing to give up on a level of a game. In each of these presentation we will split the collected data into hardcore players and casual players. In this experiment we had fourteen hardcore and ten casual players with valid answers.

In the experience we asked the players what were the best and the worst parts of the level but we only found a pattern to identify the worst parts in our system's graph as the best part can trigger multiple different emotions depending on the person. To detect the worst parts of the game in our system's graph we look for spikes of the Expected Punishment Emotion. When there is a spike of this emotion, the player start to think that he is not going to pass the challenge and start feeling frustrated.

The measurement that we used to calculate the effectiveness of our system, is the precision of the predictions of the worst parts of the level. This metric is calculated by the division of the correct predictions (parts of the level that both our system and the player pointed as worst parts) by the total of worst parts of the level pointed by the player, in that level. This give us the percentage that we call precision of the system.

$$Precision = \frac{CorrectPredictions}{TotalWorstParts} \tag{13}$$

Player Real Mode Results: In the Player Real Mode, we use all the data collected to generate the emotions that were felt during the game. Since in this mode, the system have full knowledge of what happened in the level, theoretically in this mode we are expecting to have the best precision of all the other modes presented here. This mode mainly tests how much precise is the affective agent, in measuring the emotions. The results of the Player Real Mode applied to the first level were, on average, 82 per cent precision when measuring hardcore players, and 100 per cent when measuring casual players. In this experiment we detected that hardcore players said sometimes that they did not like some parts of the level, that they passed at the first try. We talked to some these players about this decision, and in general they said that they did not like the design of the challenge. Other factor that contribute for less precision for the hardcore players is that since we did not limit the number of challenges that a player can indicate as worst parts, these players indicated more challenges that did not like than the casual players. In this matter it seems that casual players really indicated the worst parts of the level (challenges difficult to pass) and hardcore players indicate all the parts that they did not like.

Simulation Mode First Level Results: In the Simulation Mode of the first level we picked up the skill calculated from the performance of the player in this first level, and used it to predict the emotions felt in the first level using this skill. This mode was used to validate both our skill measuring system and prediction component, since we can compare the results obtained here with previous experience. The precision percent of the model here was 75 per cent for the hardcore players and 100 per cent for the casual ones. We see here that our system works as well as with the skill score as the with the full knowledge of the whole gameplay, despite having a slightly difference between the hardcore players precision in this mode and the previous one.

Simulation Mode Second Level Results: This is the main mode of our thesis, being able to predict what emotions will be felt by the player in future level using his skill and a personality trait. In this mode we had 71 per cent of precision for hardcore players and 92 per cent for casual players. As we saw the other experiences the hardcore players are harder to predict than the casual ones for the same reasons that we explained for the Player Real Mode. But the casual players also did not have the 100 percent we had in the previous two experiences.

Number of attempts before giving up: Our system generates a graph of emotions that must be interpreted by the game designer. In this experiment we gave the player the option of giving up playing the level any time he wants, so we could check how many tries were made on the last challenge before giving up. This information can be used to check in our system's graphs when a player is about to give up on the level, so the game designers can change it. For the players who only gave up on one of the two level we have a average of 7 tries before giving up for the hardcore players and 27 for casual players. This data indicates that the casual players are more persistent then the hardcore ones. The hardcore players perhaps expect to pass all the level in few tries, and if they can not do that, they give up. For the hardcore players that gave up on

both level we have 27 tries before giving up in the first level and 15 on the second one. The casual players, on average, do 39 tries before giving up on the first level and 14 tries on the second level. As we can see all players, both casual and hardcore, have less tries in the second level. Most of the giving up situations on both levels occurred on the same challenge.

Overall Results: In overall, we had better predictions for casual players than for hardcore players. But the hardcore players precision results were always above 70 per cent, that is not a bad precision rate but could be better. For the casual players the precision only drops from 100 per cent, when we try to predict a new level, that show that we can adjust better the Simulation Mode, as it can be improved for both player types.

6. Conclusions

With this work, we present a methodology to evaluate PCG reliant games, using an affective agent to complete the levels of a game. After completed each level, this agent report all the emotions felt while passing the it. We designed a conceptual model for our methodology that can potentially evaluate any type of PCG reliant game. From this conceptual model, we created PLEASED in order to test our approach. We conducted an experiment where we used PLEASED to evaluate a platform game, that we created. To verify the effectiveness of PLEASED we compared the players' feedback of the game with the evaluation of generated by PLEASED. The results from this experiment suggest that PLEASED is more suitable for simulating casual players than hardcore players. A possible explanation for this fact, can be in the difference between the evaluation criteria of a casual player and a hardcore player. In this case, PLEASED would need to take in account player's experience in the game type we are evaluating, in order to proper simulate the hardcore players. This being true, we also need to add this component to our methodology's conceptual model as it is generic concept that can be applied to any game type.

6.1. Future Work

With regard to future work, we consider important exploring the use of our conceptual model in other types of PCG reliant games, to check if our approach is generic enough to reach them. This would be achieved by implementing our conceptual model with another affective agent architecture and another personality model, and combine both into a new system. But this time, adding a decision component to the affective agent so it can decide the path to take if the game present the player multiple paths to choose from, this way this new system could be applied to even more game types.

References

[1] Gillian Smith. The future of procedural content generation in games. *Proceedings of the AIIDE Workshop on Experimental AI in Games*, October 2014.

[2] G.N. Yannakakis and J. Togelius. Experience-driven procedural content generation. *IEEE Transactions on Affective Computing*, April 2011. doi:10.1109/T-AFFC.2011.6.

[3] Georgios Yannakakis Noor Shaker and Julian Togelius. Towards automatic personalized content generation for platform games. *Center of Computer Games Research*, 2010.

[4] C Martinho. Emotivector: mecanismo afectivo e anticipatório para personagens sintéticas. *Ph.D. Thesis*, 2007.

[5] J.Dias A.Paiva R.S.Aylett, S.Louchart and M.Vala. Fearnot! - an experiment in emergent narrative. *Springer Berlin Heidelberg*, 2005.

[6] G. Clore A. Ortony and A. Collins. *The cognitive structure of emotions*. Cambridge University Press, 1988.

[7] Robert McCrae and John Oliver. An introduction to the five-factor model and its applications. *Journal of personality*, 60(2):175215, June 1992.

[8] Carl Jung. Flow:the psychology of optimal experience. *Racher Verlag*, 1921.

[9] David Kersey and Marilyn Bates. Please understand me character and temperament types. *Prometheus Nemesis Book Company*, 1984.

[10] D. M. Scrakic C Cloninger and T. R. Przybeck. A psychobiological model of temperament and character. *Archives of general Psychiatry*, 50:975–990, 1993.

[11] Richard Bartle. Hearts, clubs, diamonds, spades: Players who suit muds. *The Journal of Virtual Environments*, 1996.

[12] C Bateman. Demographic game design. *International Hobo*, 2004.

[13] B. Stewart. Personality and play styles: A united model. *Gamasutra*, September 2011.

[14] Nicole Lazzaro. Why we play games: Four keys to more emotion without story. *XEO Design Inc.*, 2005.

[15] Michael F Scheier and Charles S Carver. Optimism, coping, and health: assessment and implications of generalized outcome expectancies. *Health psychology*, 4(3):219, 1985.