TÉCNICO LISBOA

PLEASED

PLayEr Affective Simulation for progrEssion Design

Bernardo Brás Lourenço

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Carlos António Roque Martinho

Examination Committee

Chairperson: Prof. António Manuel Ferreira Rito da Silva
Supervisor: Prof. Carlos António Roque Martinho
Member of the Committee: Prof. Maria Beatriz Carmo

November 2017

Acknowledgments

First I would like to thank my thesis supervisor professor Carlos Martinho, for helping me in every stage of this work and without whom this thesis would not have the same value. My friends Ricardo Almas, João Santos and Nuno Silva for helping me testing this work since day one. Lastly, I would like to thank my family for their support not only in this work but throughout the entirety of my time at Instituto Superior Técnico.

Resumo

A geração procedimental de conteúdo é uma técnica utilizada em vários jogos. Mas se o jogo incluir muito conteúdo gerado desta maneira, torna-se difícil testar-lo com jogadores antes do lançamento do mesmo. Nestes casos é utilizado um agente de IA, para verificar todo o conteúdo, mas este apenas realiza validações básicas e não nos dá uma avaliação subjectiva como a de um jogador. Para estas situações, parece não existir uma maneira viável de testar grandes quantidades de conteúdo gerado procedimentalmente.

Criámos uma metodologia que utiliza um agente afectivo para testar um jogo utilizando a personalidade e perícia de um jogador, de modo a que se obtenha algum feedback emocional numa sessão de *playtesting*. Este agente é uma combinação entre uma arquitectura de agentes afectivos e um modelo de personalidade. Apresentamos o PLEASED, uma implementação da nossa metodologia que utilizámos para testar esta abordagem.

Para verificar a eficácia da nossa abordagem, desenhámos e desenvolvemos um jogo para ser avaliado pelo PLEASED. A eficácia do nosso sistema foi medida, comparando o feedback do mesmo com o feedback dos jogadores. Os nossos resultados sugerem que o PLEASED é mais adequado para simular jogadores casuais do que veteranos e, que estes últimos têm diferentes critérios para avaliação de jogos, em relação aos casuais.

Palavras-chave: geração procedimental de conteúdo, playtesting, modelo de jogador, computação afetiva, emoção, personalidade

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, emotion, personality

Contents

List of Tables

List of Figures

Chapter 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. This evolution may indicate that usage of PCG will probably increase in the future.

The vision of the future for PCG bring us ambitious projects like multi-level, multi-content PCG, this means for a given game engine and a set of game rules being able to generate all content of the game. Another one of these visions is PCG-based design that refers to creating games that do not only rely on procedural content generation, but for which PCG is an absolutely central part of the game. Some progress have been done in this area in games like Galactic Arms Race[1] and Endless Web[2] but these ones are still based on established game genres and core parts of the games could function without PCG[3]. The most ambitious project in this area is the power to generate complete games, for example, a generator capable of creating not only content for a given game, but the game itself. This means being able to create the rules, reward structures and graphical representation as well as the levels and characters by itself [3].

The use of the PCG brought a lot of changes to the game industry, mainly the fact that when a game uses PCG, it needs less game content done manually. Game content would take less time in the development process, giving more time to the game designer to focus other aspects of the game.

Nowadays it is used in games such as Civilization IV[3] that is a turn based strategy game that allows unique gameplay experience by generating random maps and Minecraft[4], one of the recent popular indie

[1] Galactic Arms Race. Evolutionary Games. Evolutionary Games. 2014.
[2] Endless Web. UC Santa Cruz's Center for Games and Playable Media and Northeastern University's Playable Innovative Technologies Lab. 2012.
[3] Civilization IV. Firaxis Games. 2K Games. 2005.
[4] Minecraft. Mojang. Mojang. 2011.

games featuring extensive use of PCG techniques to generate the whole world and content.

Figure 1.1: Minecraft.

Traditional PCG techniques are not controllable enough, meaning that not all important aspects of the generated content can easily be specified by the designer or by an algorithm[2]. When a content is procedurally generated, the designer have less control over its content. With less control there is more room to the introduction of game design flaws in the game.

In this document, we present a methodology to validate the design quality of the procedurally generated content so the game designer can have a quality control over the content generated this way. With this approach we can check what parts of the game potentially contain design flaws and how much impact they have in the overall game experience. The advantage of using our methodology to test a game instead of the traditional testing methods is that our process is automatic but also simulate the player's emotions, giving us the subjective evaluation that currently can only be obtained thought playtesting sessions. As the gameplay experience is tied to the personality of the player and also his skill, we have to consider them in our methodology as they crucial components in the evaluation of the game design. Using our approach the game designs can configure our system with the personality of the game's target audience and their skill to test the PCG created levels with it, having this way a instant level content quality subjective feedback.

1.1 Motivation

No Man's Sky[5] is an action-adventure survival video game developed and published by the indie studio Hello Games. The game was released worldwide in August 2016. The gameplay of No Man's Sky is built on four pillars: exploration, survival, combat, and trading. Players are free to perform within the entirety of a procedurally generated deterministic open universe, which includes over 18 quintillion (1.8×10^{19}) planets, many with their own sets of flora and fauna. This game is the most recent release of a PCG reliant game. The PCG is used in the creation of really large maps, in this case 18 quintillion of planets and each planet have much content to be explored by itself. A player can be his whole life testing this

[5]No Man's Sky. Hello Games. Hello Games. 2016

game that he probability will not be able to explore all the planets. An AI controller can do this job, but will not give us any subjective evaluation that could be the best feedback in this type of games. How can this game developers guarantee that all the planets generated procedurally deliver a good experience to the player?

Figure 1.2: No Man's Sky.

1.2 Problem Description

Nowadays PCG reliant games are mainly tested by either players or AI controllers[4]. Both approaches have advantages and disadvantages, let's talk about some of them. 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. That being said, how can we have a subjective evaluation of the whole content of a game like No Man's Sky?

1.3 Solution Description

Testing PCG games can be an issue especially if we use it to create large maps or to create real time adaptive content as this content is based on the player's actions. What if we combine the untiring AI

controllers with the subjective emotional feedback of a player? We could then test large amount of content automatically and have feedback similar to a player. But if we can go further and add a personality to this mix, now we can test the game using various types of players based on their personality. An affective agent with an associated personality would fill this role as he could pass the game level automatically and also generate emotions base on a personality. This affective agent can give us information like, if the level is possible to be finished, information that an IA controller also can give us but on the other hand can also report the emotions felt in the all level based on the personality that it has. This emotions report can be used to gives us information about the gameplay experience.

The main goal of this work is to create a system to aid game designers in the development of a PCG reliant game giving them indications about the gameplay experience quality 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 a graph of all the emotions felt throughout the level. This graph can then be read by a game designer, the graph will give indications what parts of the game are giving the player the best and worst experiences. This information can then be used by the designer to adjust the PCG game to give a player a more pleasant game experience.

With this work we wish to improve the testing phase of the PCG game development by giving the game design crucial feedback of all the procedurally generated content. This is an improvement for current approaches of testing as in this way we can have an emotional evaluation of large game content that can be used to predict the player's gameplay experience. Also our affective agent can use multiple personalities to pass the level, this can be used by the design to test the game with multiple player types, seeing this way what kind of players enjoy the game the most. For example the system could give us feedback for casual players that can be different from the hardcore player's feedback.

In the next chapter we will be focused on related work, we will see more in depth what is PCG, then we will discuss what is flow as it this related with the game enjoyment. After we will also discuss personality models and affective agent architectures. On chapter three we will explain our methodology for aiding the testing phase of PCG reliant games and also describe our implementation of this methodology. In the next chapter we will describe all the preparation made so we can perform the experience and also the experimental process used. On chapter five we will expose the results of our experiment. The final chapter is reserved for the conclusions on this work and also what can be improved in a future work.

Chapter 2

Related Work

In this chapter we will discuss some of the concepts and works that are related with our work. We will start by exploring what is PCG and its characteristics. Then we talk about flow as it explains how and why players enjoy a game. After that, we present some personality psychology models and also some personality player models as they can help us integrate personality in our affective agent. We will cover some affective agent architectures that can be used in our methodology. And finally we will talk about LOT-R that is a study about optimism and pessimism of a person.

2.1 Procedural Content Generation

Procedural Content Generation (PCG) in games is the algorithmically creation of game content with limited or indirect user input[3]. This content can be levels, maps, game rules, textures, stories, items, quests, music, weapons, vehicles, characters etc, in other words everything that we can find in a game. The most important requirement of the content generated is being playable, in other words having the same behavior of the non-procedural generated content. One of the advantages of using PCG is the fact that it does not require a human designer or artist to generate game content. This way game companies can use PCG to reduce production costs. Through the use of PCG game developing teams would not need to worry about small details that can be automatically generated, giving them more time to focus in other aspects of the game. Other advantage is the generation of content can be tailored to the tastes of the player in real time.

A PCG solution should have the following properties:

1. Speed: The acceptable speed for content generation can vary from milliseconds to months depending if it is generated during the gameplay or development of the game.

2. Reliability: Every content produced should meet success parameters, for example when a new level of a game is generated, it should be always possible to solve it.

3. Controllability: A human user should be able to set up some parameters to control the content generated.

4. Expressivity and diversity: The content generated should have some degree of diversity and should be expressive enough so the player do not feel that he is playing a similar level with only different colors for example.

5. Creativity and believability: The ways that some content can be generated is limited but nonetheless it should be generated as many different ways as possible and as believable as possible.

The research fields of this area ranges from the generation of all the content of a game to the creation a game completely free from human intervention.

Content Evaluation

Nowadays the quality of the content generated is evaluated in many ways. The most obvious way is to ask players how were the experiences after they played the game with the new content. This evaluation is normally done through rating-based questionnaires, evaluating every content generated. Other way of evaluation is to measure physiological manifestations and/or behavioral playing patterns that may map to a particular player state[1], like frustration for example. The play-testing sessions can not be used to test every single detail of a level that was generated so prior to these sessions, the content is evaluated by logical operations to check for basic design flaws.

2.2 Flow in Games

The Flow concept came from positive psychology but quickly spread to other areas in particular game design. In games, Flow concept can be used to assert if a player is really enjoying a game. As our work is all about the evaluation of the gameplay experience, this is a concept worth exploring.

Flow is a mental state of operation in which a person performing an activity is fully immersed in a feeling of energized focus [5]. Csikszentmihalyi identified eight elements in flow: goals must be clear; challenge must be present; feedback must be precise; focus should be always present; the sensation of control; loss of self-consciousness; transformation of time in flow state the notion of time is changed;action and awareness, must be merged. Some of these elements must be present in order to experience the flow state. When we talk about a challenge, we can directly relate to the player's skill, in flow state challenge and skill must be at the same level of correspondence. If the level of challenge is higher than the level of skill the player will experience anxiety. If, on the other hand, the skill is higher the player will experience boredom as we can see in figure 2.1.

Sweetser and Wyeth presented the GameFlow criteria that is an adaptation of Csikszentmihalyi's flow theory to game concepts [6]. In this new concept there are eight distinct elements: challenge, skill, concentration, clear goals, feedback, immersion and social interaction. These elements were created from the ones in the original flow theory but now adapted to games. Now challenge being the obstacles that the games present to the player and skill the player's skill. The concentration from the player is always required. The goals being clear is what keeps the player motivated to achieve them. The player should always get appropriate feedback so that the flow state is not broken. Immersion should be always a

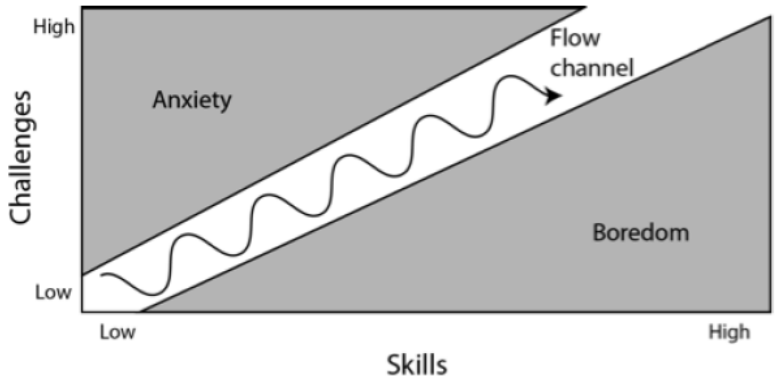


Figure 2.1: Representation of the flow zone.

goal in games but is also related to flow state and social interaction can be created through this same immersion as two players join up to do a quest, for example. When testing this theory Sweetser and Wyeth concluded that analyzing a game thought elements can give us a prediction how it will perform but it's better used as an expert review method or to structure a play-testing session. Saying this, we can view the GameFlow criteria as tool for testing a game that is in the first stages of development.

2.3 Personality models in Psychology

The evaluation of a game is always related with the player's personality. For example an aggressive player would like to kill enemies even if it is not the objective , on other hand a defensive player would try to circle around the enemies to get to the objective. For that reason we need to identify the personality of the player, so we can use it on our agent. Two versions of the agent with two imbued personality will generate two different evaluations of the game. In this section we will discuss personality models that can be used in our work.

2.3.1 Five Factor model

Five Factor Model is a theory that define a person's personality through five different dimensions or traits. This theory was tested in a wide range of participants of different ages and culture and proved to be consistent[7]. These traits are Extraversion, Agreeableness, Conscientiousness, Neuroticism and Openness to experience. The Extraversion relates to the desire of external stimulation, people with high score in this trait tend to look for others and being more social in general. Agreeableness deals with the capacity of cooperation of a person and how receptive is to others. Conscientious people are self-discipline and prefer to plan rather than having a spontaneous behaviour. Neuroticism trait reflects how easily a person can develop negative emotions like anger and anxiety. And finally Openness to experience trait is related to creativity and imagination, basically distinguish a creative intellectual person from a realistic pragmatic one.

2.3.2 Myer Briggs Type Indicator

Myer-Briggs Type Indicator (MBTI) is an instrument that allows us to identify psychological preferences. Originally Carl Jung developed the theory of psychological types[8]. This theory states that a human have two pairs of cognitive functions: thinking vs feeling for judging and sensation vs intuition for perceiving. These functions can then be expressed in an introverted or extroverted way. MBTI was based on Jung's theory but added a new cognitive function: judging vs perceiving for representing the interaction with the outside world. While the Jugian theory only supports eight personality types , the MBTI supports a total of sixteen types since with the new cognitive function. In the field of gaming, the MBTI is one of the most used psychological model due to its simplicity and easy adaptation to player models like DGD[9]. However its validity has been questioned in the field of personality psychology[10]. One of the problem is the mutually exclusive types that do not allow to identify a personality which have the two sides of the type in the same proportion.

2.3.3 Keirsey Temperament Model

David Keirsey through the analysis of classical temperaments and MBTI presented his theory. This theory is focused in temperaments rather than thoughts and emotions, therefore it presents four new tempers[11]. From the sixteen junctions of MBTI , Keirsey identified the following tempers: the SJ or Guardians , the SP or Artisans, the NT or Rationals and the NF or idealists. The Artisans like to be free to what they want when they want, they are impulsive and show strong resilience to failure. The Guardians seek security and feel obligated to follow the rules. The Rationals like to understand how the things work and are self-critical. The Idealists like to cooperate with people and look for self-actualization.

2.3.4 Cloninger's Temperament and Character Inventory

Cloninger's Temperament and Character Inventory (TCI) is an inventory for personality traits to describe individual differences in psychopathological behavior. Other models try to describe normal forms of behavior but TCI describes maladaptive ones. In this theory Cloninger introduces seven dimensions: four temperaments and three characters [12]. The temperaments are individually heritable and can be seen in early stages of personal development. Novelty Seeking, Harm Avoidance, Reward-dependence and Persistence are the temperaments. To explain the differences between people with the same temperaments we have the three characters that are: Self-directedness, Cooperativeness and Self-transcendence. These characters are also related to the way we learn. Each one of these dimensions has a varying number of subscales.

2.4 Personality based player models

Now that we talked about personality models in psychology, now it's time to talk about the personality models for players. The personality models that we will present in this section were formulated also to model the personality of a person but more specifically a personality of a player.

2.4.1 Bartle player types

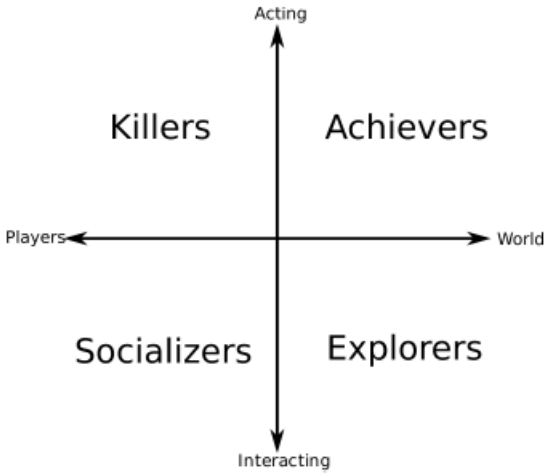


Figure 2.2: Representation of the four Bartle Types.

Bartle player types is one of the first player models and was created by Richard Bartle thought the analysis of the players interactions in MUD's (Multi User Dungeon). Bartle identified that there was two dimensions player-world and interacting-acting[13]. These dimensions are related to gameplay preferences, player-world refers to the player liking to interact with other players or with the environment and interacting-acting is the way the interaction is done. Each combination of these preferences lead us to a player type, there are four in total(figure 2). Players that like to act on other players are called killers and their objectives are to show superiority or bring grief to other players. The other type that likes to act is the achiever but they act on the world , their goal is to control the environment around them. In the interaction side we have the socializer that likes to interact with other players their enjoyment of the game is playing with other people and have interactions with them. And finally the explorer prefers to interact with the world, they like to discover new things and gain knowledge about the game. The biggest downside of this player model is that was created based on MUD players, for this reason using this player model in other game genres have limitations.

2.4.2 Demographic Game Design

Demographic game design is a popular model that focus on market oriented game design and was proposed by Bateman[14]. In this model four types of players are identified: the conqueror, the manager, the wanderer and the participant each of them with two subgroups to distinguish between experienced and casual players. The conqueror displays strategic and logistics skills, the manager present strategic and tactical skills, the wanderer have a diplomatic and tactical skillset and the participant have a diplomatic and logical one. This player model is based on the psychology model MBTI that only uses two cognitive functions, these can be related to these four types of players as we can see in figure 2.3.

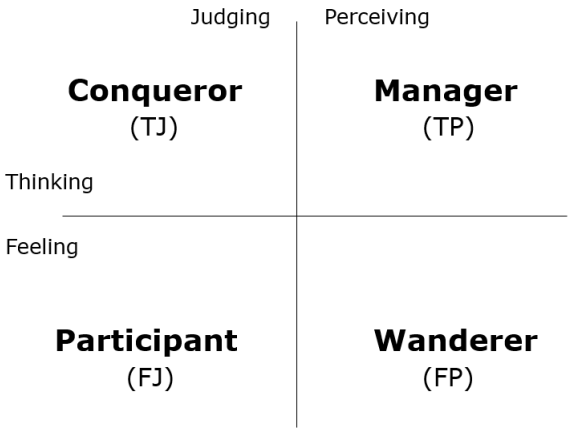


Figure 2.3: Relation between Demographic Game Design and MBTI.

2.4.3 Unified model

Since the some of the previously presented personality models can be related with these player models, a unified model that correlate them was proposed by Steward[15]. In this unified model there is a relation between DGD1,the four Bartle types and the four Keiser temperaments. For example, the Kersey temperaments are a superset of the four Bartle types. The killer is a specific type of Artisan, the Socializer a type of Idealist, Explorer a type of Rational and Achiever a type of Guardian. Other correlations can be seen in more detail in figure 2.4.

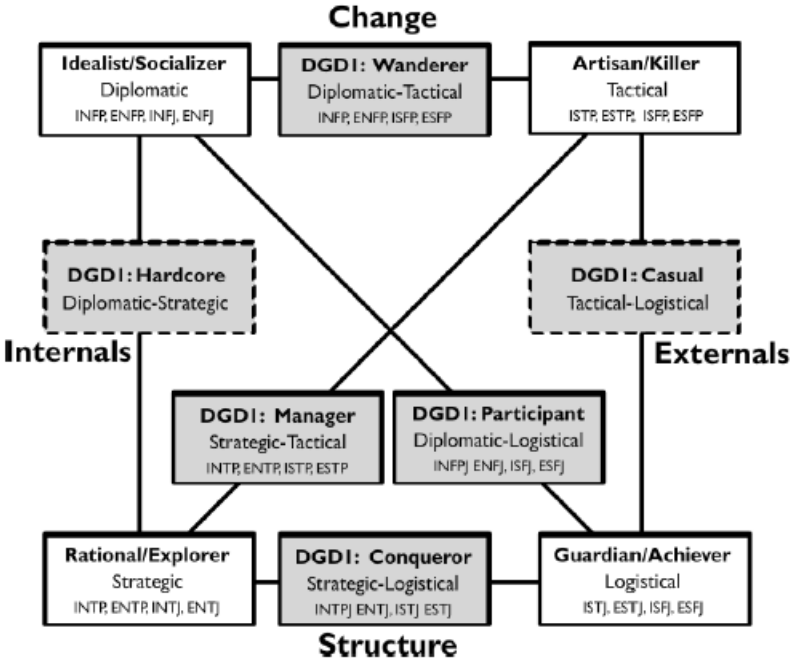


Figure 2.4: The Unified Model.

2.4.4 Lazzaro's fun types

Lazzaro's fun types classify players thought their motivations and emotions during the gameplay[16]. A player can be one of the following types:

- Hard fun, players that like challenges, problem solving and normally have emotions like frustration and triumph when playing.
- Easy fun players that like intrigue, curiosity and enjoy feeling emotions like wonder and mystery.
- Altered State, the enjoyment of the players derive from their internal experiences in reaction to the visceral, behaviour, cognitive and social properties.These type of players have associated to them emotions of excitement and relief.
- The People Factor players that use the games as a way for obtaining social experiences. Normally these players feel amusement and pleasure from the success of pupil that they mentor.

2.5 Affective agent architectures

A software agent can be viewed as conjunction of three components: sensors, effector and processing[17]. The agent use the sensors to perceive the environment then, it process the information and act on the environment through the effectors. Now an affective agent have the same components but the emotions influenciate all the cycle. The emotions affects of all parts of the agent, from the way the agent perceives the world to the way it interacts with it. In this section we will present agent architectures, that are relevant for our work.

2.5.1 The OCC Model

Before we present the architecture that we consider relevant for our work, we will introduce the OCC(Ortony, Clore and Collins) model that is used in some architectures that we will describe later on this section. The OCC model consider that emotions develop as a consequence of certain cognitions and interpretations[18]. These interpretations can be different depending on the person and the person's culture which can lead to different emotions. This theory exclusively concentrates on the cognitive elicitors of emotions. In OCC model there are three aspects that determine these cognitions: events, agents, and objects. The main objective of Ortony, Clore and Collins's research is to investigate the possibility to design a formal system or a computer, that is able to draw conclusions about emotional episodes which are presented to it.

2.5.2 FAtiMA

FAtiMA(Fearnot AffecTIve Mind Architecture) is a hybrid agent architecture (BDI and reactive) with emotions. This architecture was created to control the agents in the project FearNot![19]. FearNot! is an application for anti-bullying education. In this architecture the appraisal process is based on the OCC model. The emotional state generated by the appraisal process is sent to the action-selection system. This system is composed of two layers. A reactive one that generate actions without making plans, it

is just a action that is done in response to an emotional state (for example a kid starts crying when bullied). The second layer is the deliberative one and is divided in two kinds of coping: emotional-coping and problem-focused coping. This layer use the emotional state to create plans to achieve goals that are important to the agent, for example an agent begs to stop being bullying, in order to achieve the goal of not being hurt by others.

2.5.3 Emotivector

Martinho's work introduces an anticipatory sensory interface module composed of affective anticipatory mechanisms: the emotivectors[20]. 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. In our work we want that our agent have a human-like behaviour and emotions while playing the game so we can use the emotivector to achieve that purpose.

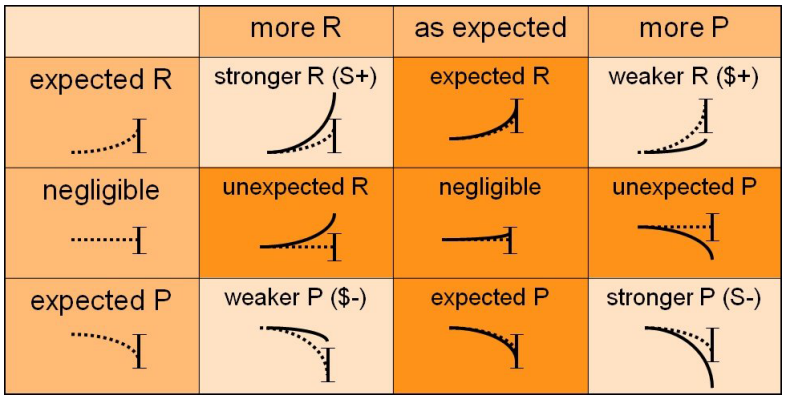


Figure 2.5: The nine sensations of the Emotivector.

As previously explained 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. All the nine possible situations are described in the figure 2.4. We can have a better, worst or expected reward and the same for the punishment.

2.5.4 EMA

EMA(EMotion and Adaptation) is a computational framework where appraisal and coping act as core of reasoning components for human like agents [21]. EMA was used in a military reality training software, called Mission Rehearsal Exercise, to control virtual humans. The EMA's algorithm has five steps. First it constructs a vision of the world through BDI (beliefs,desires and intentions). In the second step, it generates multiple appraisal frames that characterize features of the causal interpretation in terms of

appraisal variables. In the third step it maps these frames into individual instances of emotions. After that,it aggregates the instances into a emotional state and mood. And finally it chooses a coping strategy in response to the current emotional state.

2.6 Life Orientation Test Revised

Life Orientation Test Revised (LOT-R) 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 pessimistic.

People differ widely from each other in how they approach the world. Some people tend to be favorable in their outlook. These optimists expect things to go their way, and generally believe that good rather than bad things will happen to them. Other persons have an opposite set of beliefs. These pessimists expect things not to go their way, and tend to anticipate bad outcomes[22].

2.7 Related Work Discussion

After analyzing all presented affective agent architectures, we decided to use the Emotivector. This agent architecture takes into account the past events to infer the emotion that is currently being felt, this is a factor that must be take 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 situation 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.

Chapter 3

Solution

In this chapter, we introduce a methodology to give game designers, that use procedural content generation in their games, a way to test game levels automatically using profiles of players with different skills and personalities. This way while the game designer is developing the game he can automatically test it after some changes and see in our system the following information: what group of player will enjoy the change; if the level difficulty suddenly became too hard; what the players will feel in the new part of the game. When a developer change a rule in the level generation algorithm, he could be changing, for example, one hundred levels at the same time. In this case our system would be useful. The developer can use the system to check if there is any part of the level that is less enjoyable to some group of player. If that group of players are the game's target audience and, our system indicates that they are not liking the new changes, now he knows that he must revert the changes he just made because that is not going to please his target audience.

The main goal of this work is to create a system that is capable of not only model the emotions from a gameplay session but also predict the emotions that would be felt by the same person in different levels of the game. To achieve this goal we created an affective agent that completes the levels of the game by using a player profile. This profile consists in the player's skill and personality traits. After this agent completes the level it gives us all the emotions felt in the level by the player.

In this chapter we will start by introducing the conceptual model of our system. Then we will present our system, it's called PLEASED (PLayEr Affective Simulation for progrEssion Design). In this part we will explain how it works and how we combined the Emotivector with the LOT-R that we talked about in the previous chapter.

3.1 Conceptual Model

Our methodology (seen in figure 3.1) 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.

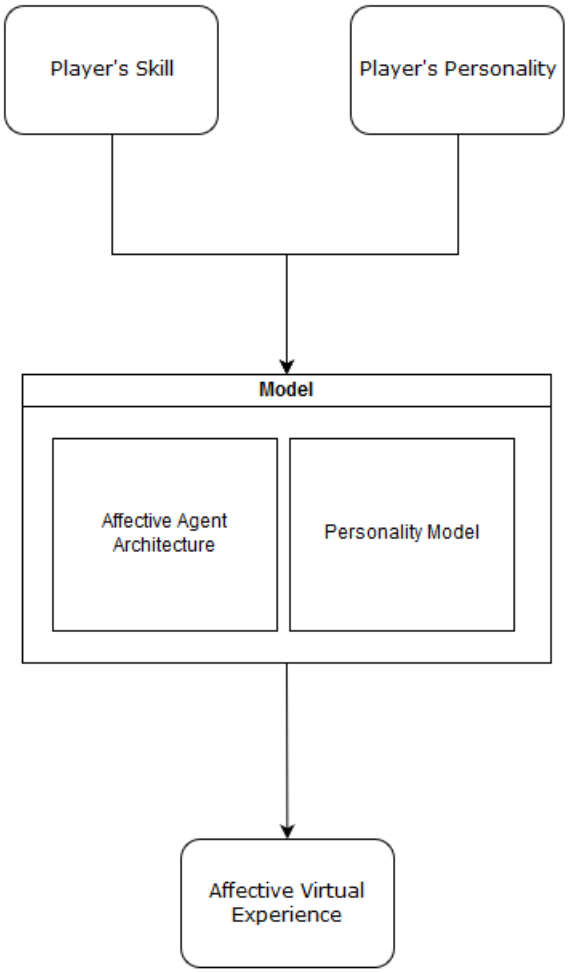


Figure 3.1: Conceptual Model

3.2 PLEASED

PLEASED (PLayEr Affective Simulation for progrEssion Design) is an implementation of the conceptual model previously presented. This system manages an affective agent that is responsible for completing a level of a game and retrieve emotional feedback from it. In the previous chapter we presented many candidates for personality model and agent architecture to implement our conceptual model. In the end we decided to use the emotivector and the LOT-R. In this section we will explain the architecture of

PLEASED

3.2.1 Model

Our system PLEASED (seen in figure 3.2) 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 use the player's skill by challenge 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 to continue progressing through the level, for example, defeat an enemy can be a challenge.

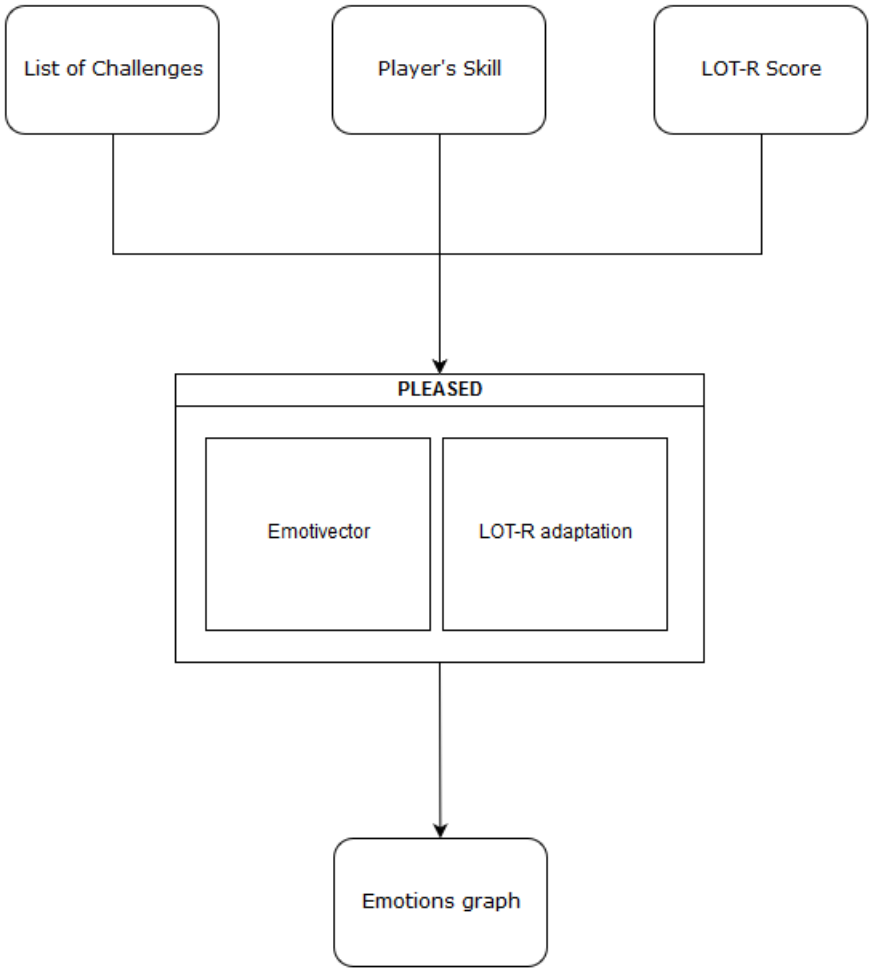


Figure 3.2: PLEASED model.

PLEASED is a combination of the affective agent architecture emotivector and the personality trait

LOT-R. The system uses a modified version of the emotivector's sensation calculation method with only six sensations instead of nine that were used on the original emotivector. Our system's affective agent uses the mismatch between the expectation and reality to generate emotions.

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 to calculate the expectation of the agent in each attempt of the challenge. This personality trait was chosen because is related with the expectation that plays an important role in the emotivector architecture and having an accurate expectation can improve the reliability of our system.

The reality component of our system is simulated by using the player skill by challenge that we receive as input. This skill represents the player's probability to pass the challenge in each attempt. We use this probability to simulate if the player passed the challenge or failed in current attempt.

Our system gives information of the emotions felt by the player in each attempt of each challenge. So for each attempt of each challenge our system will calculate the emotion felt. After we calculated the emotions of all challenges, we generate a graph showing the emotions felt in each challenge (multiple emotions in one challenge means that the player did more than one attempt on the challenge). This graph can then be read by the game designer, giving him indications of the game's quality.

3.2.2 Architecture

In this section, we will explain the PLEASED architecture (seen in figure 3.3). We also will describe how LOT-R was combined with the affective agent architecture emotivector, to form our system's affective agent and also how the feedback of our system is presented to the game designer.

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

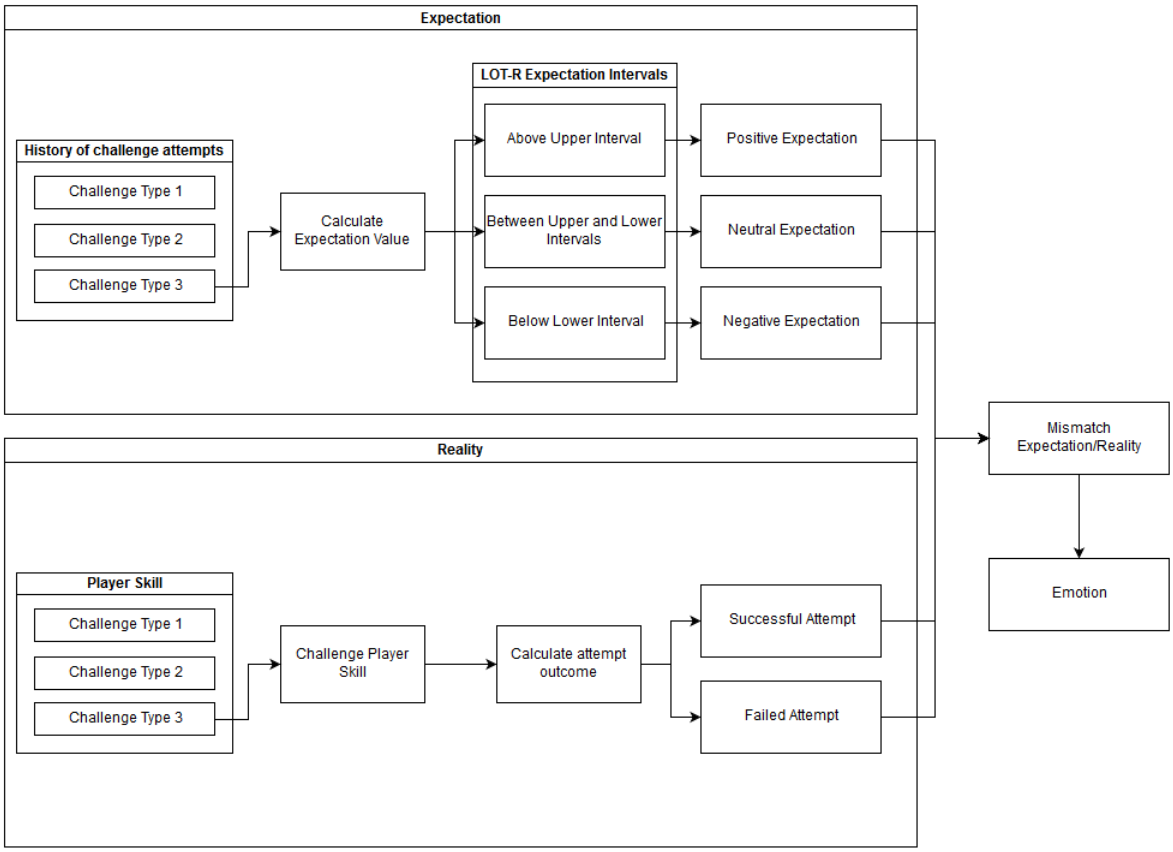


Figure 3.3: PLEASED Architecture.

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. Every other score between 0 and 50 is a failed attempt but higher scores means that the player is closer to pass the challenger and scores between 50 and 100 are all successful attempts with less or more flaws. 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:

$$at_n = attemptscore \tag{3.1}$$

$$f(n) = attemptweight \tag{3.2}$$

$$w_n = \frac{f(n)}{100} \tag{3.3}$$

$$Expectation = \sum_{n=1}^{5} at_n \times w_n \tag{3.4}$$

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 as it is part of the expectation and is related with the number of attempts to pass a challenge and thus is related with the skill.

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, this value is called personality score.

So we used the LOT-R value to calculate the two values from 0 to 100, these are the lower interval and the upper interval where if the expectation skill component is less than the lower interval, the player expects to fail the challenge, if it between the lower interval and the upper interval or equal to them, he has doubts about success of the next try and if it is greater than the upper interval, he expects to pass the challenge.

$$ps = personalityscore \tag{3.5}$$

$$UpperInterval = g(ps) \tag{3.6}$$

$$LowerInterval = g(ps) \tag{3.7}$$

To find the function to calculate the Upper and Lower 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. The test was performed with two pessimist players and two optimistic players, we presented them a difficult challenge of a 2d platform game so none of them would pass at the first try. Before every attempt we would ask them what were their expectations for the next attempt. With this feedback we adjusted the intervals to their expectations. After we had the intervals of the most optimistic players and the most pessimistic players, we used the two Upper Intervals from both optimistic and pessimistic players and we did a linear interpolation of them and the same was done to the Lower Intervals. The formulas to calculate the expectations intervals through the value of the LOT-R are the following:

$$UpperInterval = -\frac{17 \times ps}{15} + \frac{1080}{15} \tag{3.8}$$

$$LowerInterval = -\frac{17 \times ps}{15} + \frac{931}{15} \tag{3.9}$$

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, 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 has no experience in that kind of games more so he will feel unexpected type emotions. 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:

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

Table 3.1: 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 3.4), 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.

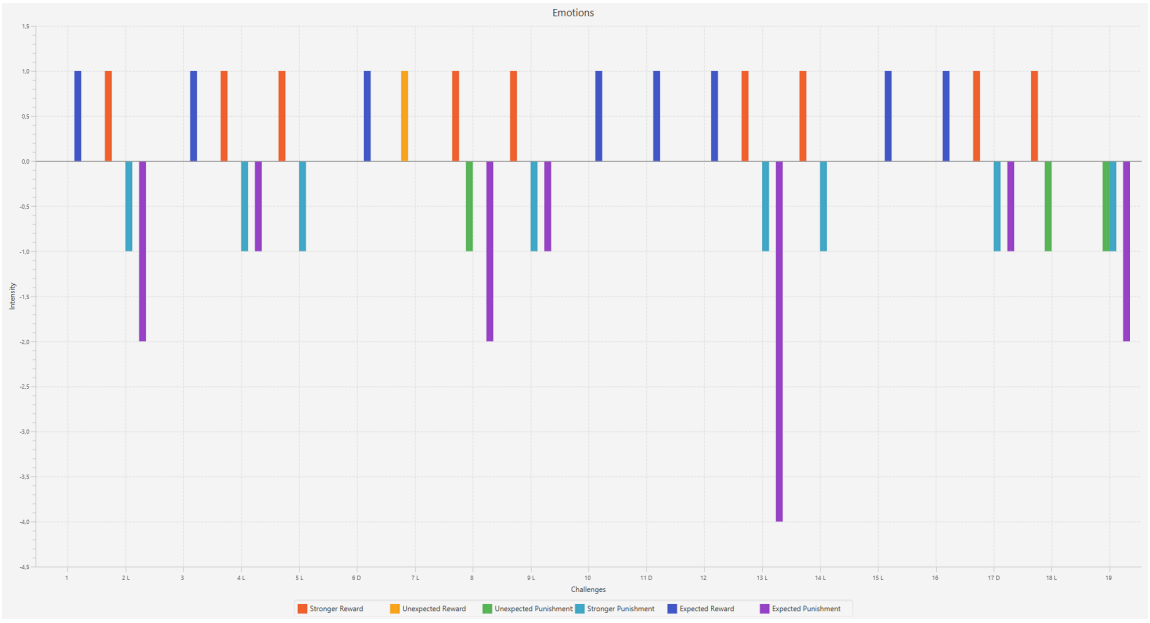


Figure 3.4: Emotion Graph Example. Emotions from left to right: Red - Stronger Reward, Yellow - Unexpected Reward, Green - Unexpected Punishment, Light Blue - Stronger Punishment, Dark Blue - Expected Reward, Purple - Expected Punishment. X Axis - Number of the challenge, Y Axis Number of times the emotion was felt.

3.2.3 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.

Player Real Mode

The Player Real Mode's main objective is to test the emotion modelling module of our system. In this mode we simply use the information of what happened in a gameplay session to generate the emotion 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.

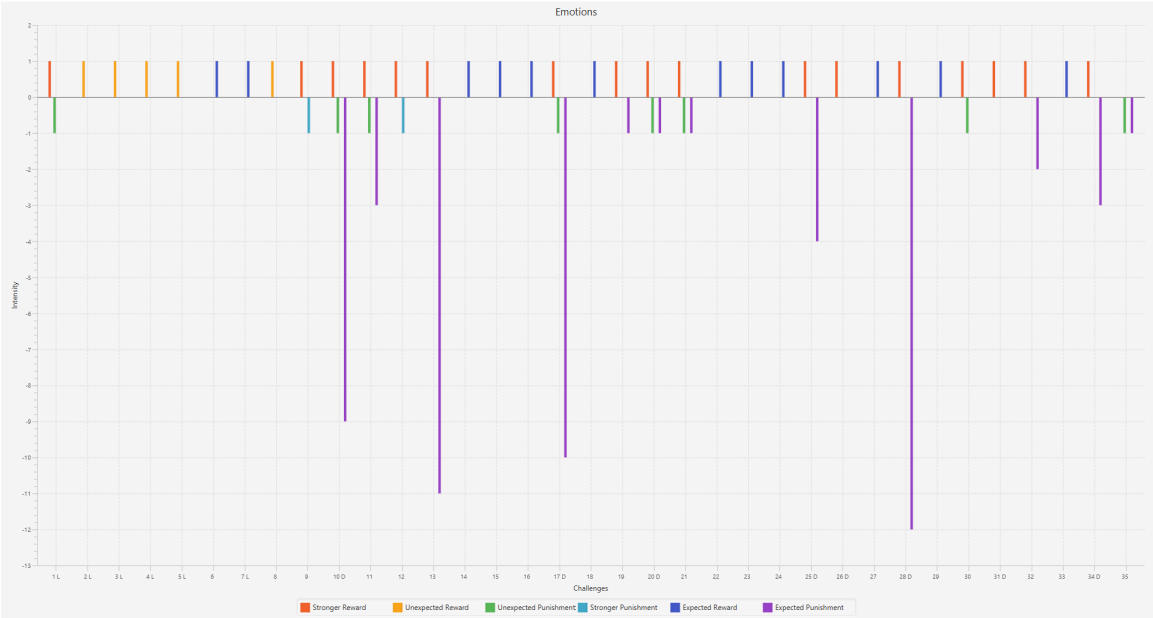


Figure 3.5: Player Real Mode Graph. Emotion Graph Example. Emotions from left to right: Red - Stronger Reward, Yellow - Unexpected Reward, Green - Unexpected Punishment, Light Blue - Stronger Punishment, Dark Blue - Expected Reward, Purple - Expected Punishment. X Axis - Number of the challenge, Y Axis Number of times the emotion was felt.

Simulation Mode

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 his performance in a level that he never played, 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 (3.10) 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 (3.10) 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.

$$ChallengeSkill = \begin{cases} \frac{100}{f}, & \text{if } f \geq 1 \\ 100, & \text{if } f = 0 \end{cases} \tag{3.10}$$

$$ChallengeTypeSkill = \frac{\sum_{n=1}^{z} ChallengeSkill_n}{z} \tag{3.11}$$

In (3.11) 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.

Figure 3.6: Simulation Mode Graph. 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. X Axis - Number of the challenge, Y Axis Number of times the emotion was felt.

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.

3.2.4 Implementation

Now that we explained our model, we will talk about how it was implemented. Our system was made in Java (version 8) [1] using Eclipse (version 4.5.2) [2]. Java is a general-purpose computer programming language that is object-oriented and Eclipse is an integrated development environment used in computer programming, and is the most widely used for Java programming. Technology wise the requirements of our system were to receive as input a file, the log of the gameplay, and generating as output a graph of the player's emotions. Between the the input file and generated graph there is our model. So we choose the input file to be a XML file as it a good way to transfer information between a game and another program. For the graph generation we used the JavaFx library[3]. Our system have a simple

[1] http://www.oracle.com/technetwork/java/ - [Online; accessed 15-October-2017]
[2] http://www.eclipse.org/ - [Online; accessed 15-October-2017]
[3] http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm - [Online; accessed 15-October-2017]

interface where we can introduce parameters, calculate the player's skill and choose the mode that we want to execute (Simulation Mode or Player Real Mode). The parameters that we can introduce are the personality score and how many simulations of the level the Simulation Mode will perform.

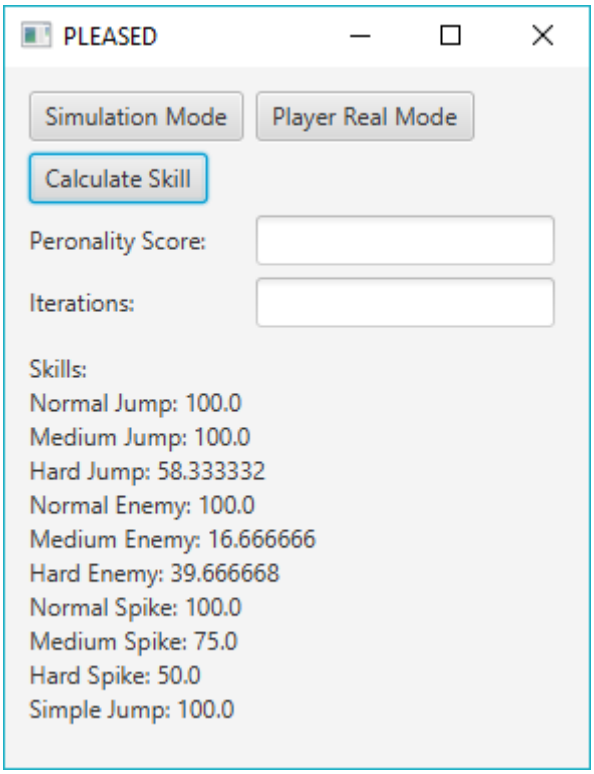


Figure 3.7: System Interface

The XML file received (seen in figure 3.8) by our system should contain information about each challenge of the game and also what were the challenges that the player liked and the ones that he disliked. Each challenge must have the following information: identifier of the challenge, the name of the type of that challenge, the order of that challenge inside the same type and the number of failed attempts performed in that challenge.

```
<challenge ChallengeNumber="1">
  <type>Normal Spike</type>
  <orderType>1</orderType>
  <numberOfFailures>1</numberOfFailures>
</challenge>
```

Figure 3.8: XML Challenge Example

In the Player Real Mode after we choose the graph mode that we want to generate, upload the game log XML file and introduced the LOT-R Score, our system read the file. After reading the whole file to memory, it will apply our model to each attempt of each challenge and store the emotions generated by each attempt. After processing all data, it will use the stored emotion information to generate the graph. The difference of the input data between calculating the Simulation Mode and the Player Real Mode in that in the Simulation Mode we choose a map of the new level to be simulated, how many times it will be simulated.

3.3 Summary

In this chapter, we presented an overview of our model. We divided this overview in two parts, the conceptual model and an implementation of the conceptual model. The conceptual model is the core idea of our work and can be implemented in a different way than the one that we presented, PLEASED. After we presented PLEASED, one implementation of our conceptual model that uses the agent architecture emotivector and the personality trait optimist/pessimist given by the LOT-R. We explained how we combined the emotivector and the LOT-R into one agent architecture and also how it works. As our system have two modes in which it can be used, we introduced them and explained what were the differences between them. Lastly we described how PLEASED was made in a technical point of view and how we can use the system.

Chapter 4

User Study

In this chapter we will present all the components we used to test our system PLEASED. To validate our system PLEASED we need a game that it can evaluate, so we created a game for that purpose. But we need to guarantee that our game is good enough in terms of usability, mechanics and fun factor so the experiment of the system don't get affected by the game's poor performance. To solve this problem we performed usability studies with players until we assured that they had the experience we wanted them to have. In the previous chapter we introduced our system, as we want to use it to evaluate our game, we need to make the adaptations to our model to be able to evaluate it. To perform these adaptations to PLEASED we had to perform studies with players until the feedback given from them would match the feedback from PLEASED. After we created and tuned the game and adapted our system PLEASED to it, we elaborated the experimental process.

This chapter starts with the introduction of the game that we used in the evaluation of our system. After that we will talk about the game's usability tests that were made to make sure that the game was delivering the experience, we wanted it to deliver. We will also explain how we adapted our system PLEASED to this game in particular and what were the tests made to do this adaptation. In the end we introduce our experimental process.

4.1 The Game

For this work we created a game to use as an example of how our system could be used to evaluate a game. This game was created from scratch using Unity[1]. Unity is a cross-platform game engine developed by Unity Technologies, which is primarily used to develop video games and simulations for computers, consoles and mobile devices. For our experiment we created a game for Windows as it is the most used operating system and we want to reach as many people as possible but it could be easily ported to other platforms.

Our game is a side-scrolling 2d platformer (seen in figure 4.1) where the player must overcome challenges like killing enemy, jumping gaps and pass thought a series of spikes. The character have unlimited number of lives and when dies it respawns near the place where he died.

[1] https://unity3d.com/ - [Online; accessed 15-October-2017]

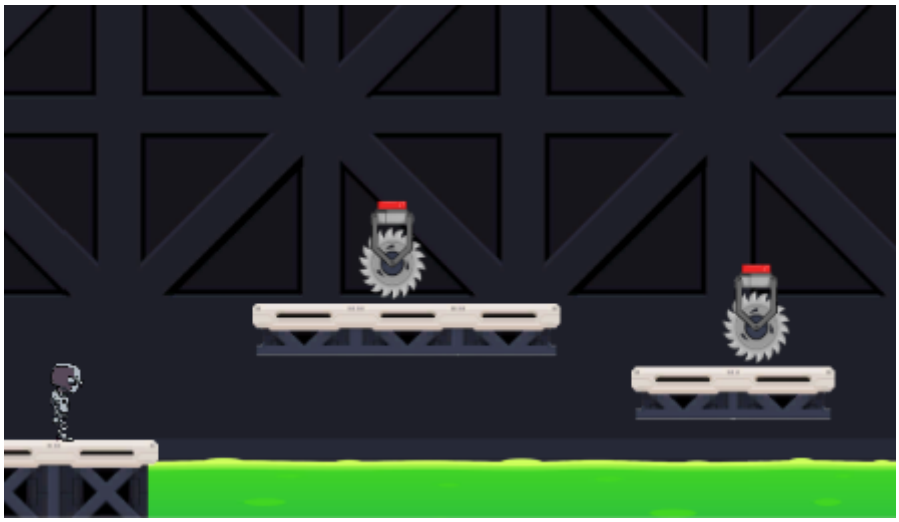


Figure 4.1: Screenshot of the game.

In this game we want to give the player intense emotions in some parts and less intense emotions in others, so we can track them in our PLEASED emotion graph. To accomplish this, we created difficult challenges that the most casual players would give up on them and normal challenges that most player can pass with ease.

In this game we simulate a PCG game as we have three different types of challenges with three different level of difficulty, and each of them appear three times in each level. Also the challenge order inside the level follows a rule. In our game we have three levels of difficulty, normal, medium and hard. Our rule to order them is that we must have two challenges of the previous difficulty before we can have a higher difficult one, so we can have a regular level progression in order to the player enter the flow zone that we talked in chapter 2.

Our first level of the game starts with a learning area where the player can not die so he can try the game mechanics and learn some of the game logic, for example how he can kill the enemies that he might face. Our game have two levels and each level have the following challenges:

- Simple Jump
- Normal Jump
- Medium Jump
- Hard Jump
- Normal Enemy
- Medium Enemy
- Hard Enemy
- Normal Spike
- Medium Spike
- Hard Spike

4.1.1 Learning area

The Learning area (seen in figure A.1) was designed to teach the player the basic mechanics of the game. In this area we have red arrow pointing out what the player must do in order to advance in the level. Here we teach how to kill the enemies and also show that when the player kill a enemy a platform will appear, without this platform the player can not progress thought the level. The player can not die in this area and can only leave it when completed all the basic mechanics of the game.

Next we will describe each one of the challenges and what is their purpose in the game and in the experience.

4.1.2 The Jump Challenges

In our game there are three difficulties of jump challenges as we stated before. Each of them is needed to prepare the player to the next difficulty and also to give information of the player's skill. Let's start with the normal jump (seen in figure 4.2), this the first jump challenge that appears in each level, this is the easiest one. This challenge is composed of a easy to perform jump and a controlled landing. If the player touches the spike or fall into the green acid, he dies.

Figure 4.2: Normal Jump

In this challenge we wanted to give the player the opportunity to get used to the jump mechanic but also giving the player a danger to worry about, the spike. If there was no spike, the challenge would became trivial. As we already gave the player trivial challenges in the learning area of the level, we want the first real challenge to be a bit harder so we can put the player inside the flow zone as soon as possible. When the player is in the flow zone any big mismatch between the player's skill and the challenge's difficulty would cause the player to fall off the flow zone, and this moment can be caught by our system's emotion graph.

The next tier of jump challenge, is the medium jump (seen in figure 4.3). In this jump we wanted the player to grasp the max range jump of the character. In this challenge the player must jump at the edge of the first platform so we can make the pass successfully the challenge. This challenge is harder comparing to the normal jump. After passing this challenge, the player will know what is the max jump range that will be needed to pass the next difficulty of the jump challenges, the hard jump.

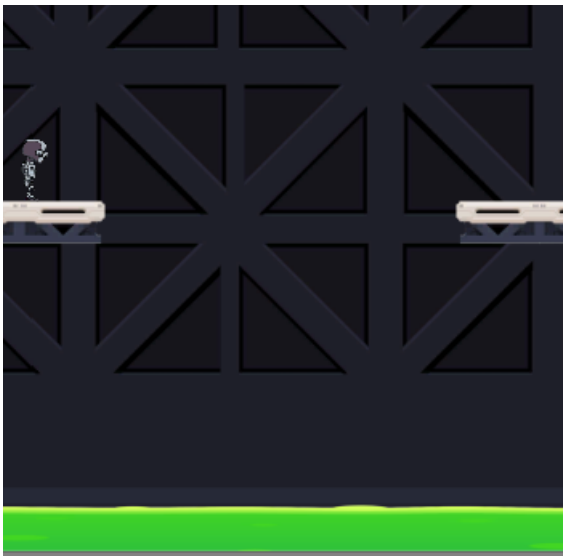


Figure 4.3: Medium Jump

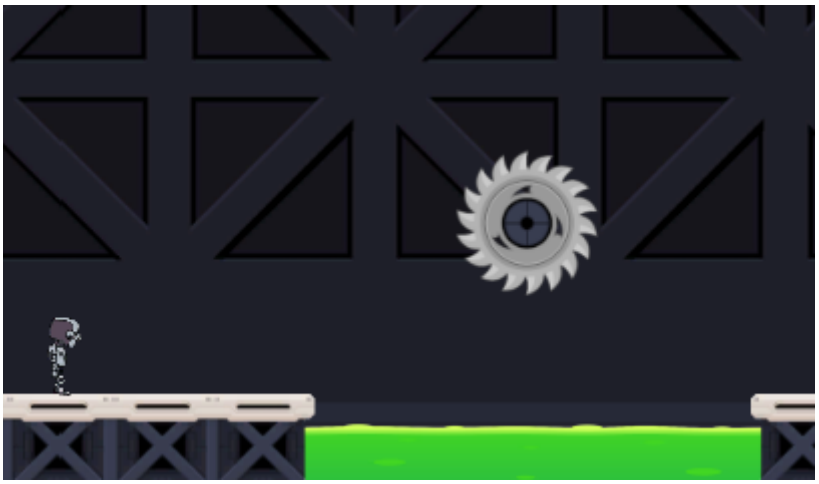


Figure 4.4: Hard Jump

In the hard jump challenge (seen in figure 4.4) we wanted the player to perform a max range jump but also worry about something else. In this case, the other thing that the player need to hurry about is a giant wheel that is going up and down in the middle of the jump, giving the player a synchronization problem to solve. So in this challenge, the player need to perform a max range jump with a perfect timing to be successful. This is the hardest challenge of this type and it is a combination of the previous difficulty with another factor, the wheel. With this challenge we can check if the addition of the wheel to the challenge keeps the player in the flow zone by watching the emotion graph that will be generated.

Lastly the simple jumps (seen in figure 4.5), these jumps are fillers of the level. Every time we needed a part to link two challenges, we add a simple jump. This is the easiest challenge of the whole level. We need to consider every part of the level, so we can have a view of the player's experience as a whole, for that reason we must consider the simple jumps in our calculations.

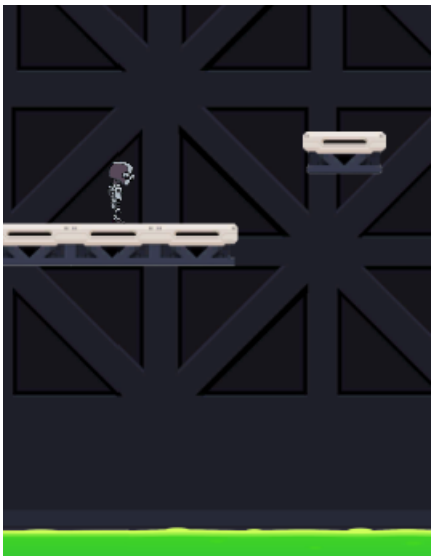


Figure 4.5: Simple Jump

4.1.3 The Enemy Challenges

The enemy challenges involve killing the enemies that appear throughout the level. In this game we wanted to measure the skill of the player in every part of the game. For that reason, the player can only continue the level if he kills the enemy that is in front of him. This way, we can collect information about his skill in killing enemies. We achieve this by removing a platform crucial to continue the level, and only make it appear when the player kills the enemy. As the jump challenges the objective here is also to make the player stay in the flow zone. But in this set of challenges, we have a challenge which its objective is to interrupt the flow zone. The hard enemy is designed in a way that is not fair to the player, to check what is the limit of tries that the player perform before giving up, this challenge will be explained later.

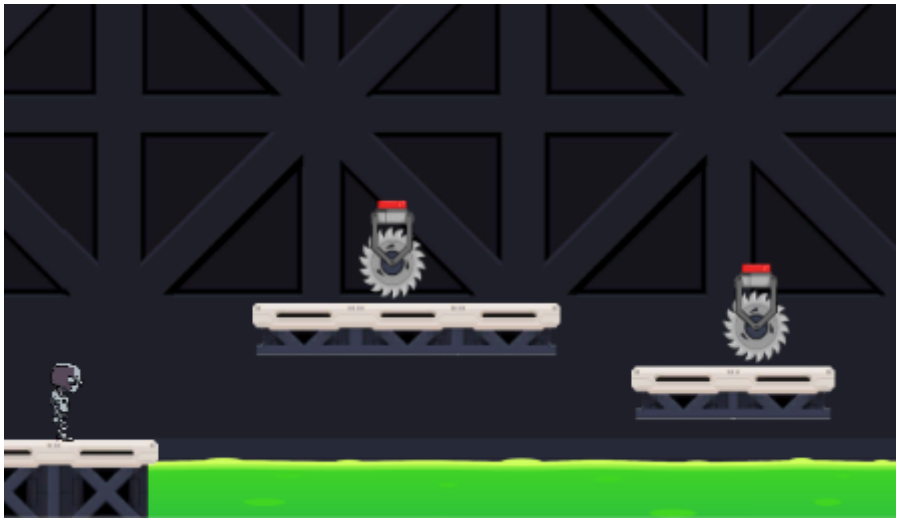


Figure 4.6: Normal Enemy

The normal enemy (seen in figure 4.6) is the first enemy that appears in the first level after the learning zone. This challenge contains two enemies, the first one is faster but patrols in a large platform the other

is slower but patrols in a small platform. The player do not need to kill both of them to continue the level, when one of them die a platform to continue the level will appear. This challenge give the player the option of killing both enemies or just one and avoid the other, but on the other enemy challenges the player must always kill the enemy. In this tier we let the player some degree of freedom in his choice of avoiding the enemy. In this game to kill an enemy the player must jump on the red button on top of the enemy.

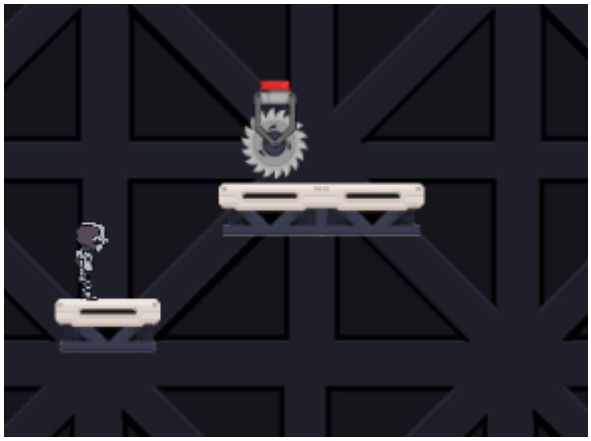


Figure 4.7: Medium Enemy

Next we have the medium enemy (seen in figure 4.7), this challenge is a problem of coordination. The enemy is fast and it is in a small platform that is always in upper ground in relation to the player's platform. This is one of the hardest challenges of the game. The player must jump to the small platform without touching the enemy, and then jump again and land in the enemy's red button. This an hard sequence of steps and any misstep can easily lead to the death of the player.

The hard enemy challenge (seen in figure 4.8) is the hardest of the game. This challenge requires precision, timing and quick reflexes. In this challenge the player must jump through the spikes and land on top of the enemy at the same time, otherwise he fails the challenge. There is other solution to this challenge that is to jump through the spikes land on the platform wait for the enemy to be near the character and jump again thought the spike and land on the enemy but this is really hard to perform. We made a challenge like this to test the player persistence and to check how many times a player tries to pass a challenge before giving up as this can be interesting to notice in the emotions graph, the pattern of a player giving up, like what emotion he felt before giving up.

4.1.4 The Spike Challenges

The spike challenges involve the player jumping to a hole between a series of spikes. These challenges often require the player to control the character in mid air so it can enter the hole a not get hit by the spikes. If the player touches a spike, he will be placed in the nearest checkpoint and must try to pass the challenge again. Like the other types of challenges, this one also increase in difficulty throughout the level.

The easiest challenge of this type is the normal spike. This challenge require the player to jump from

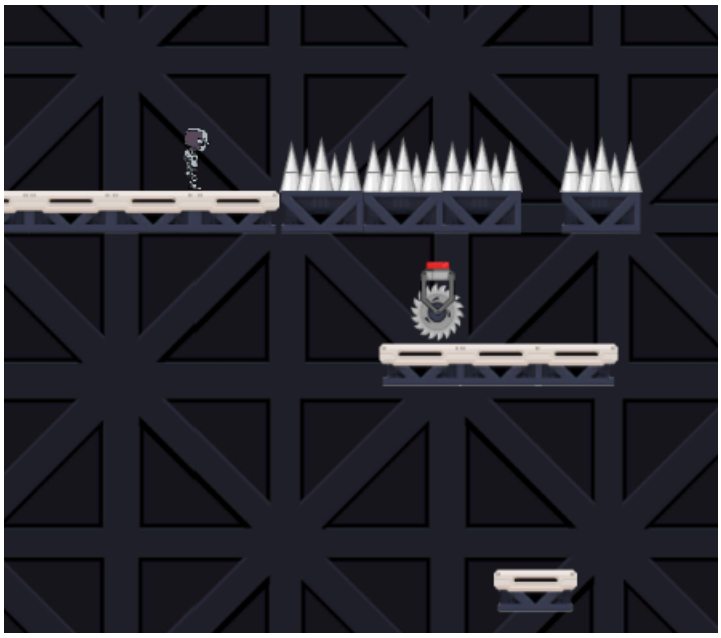


Figure 4.8: Hard Enemy

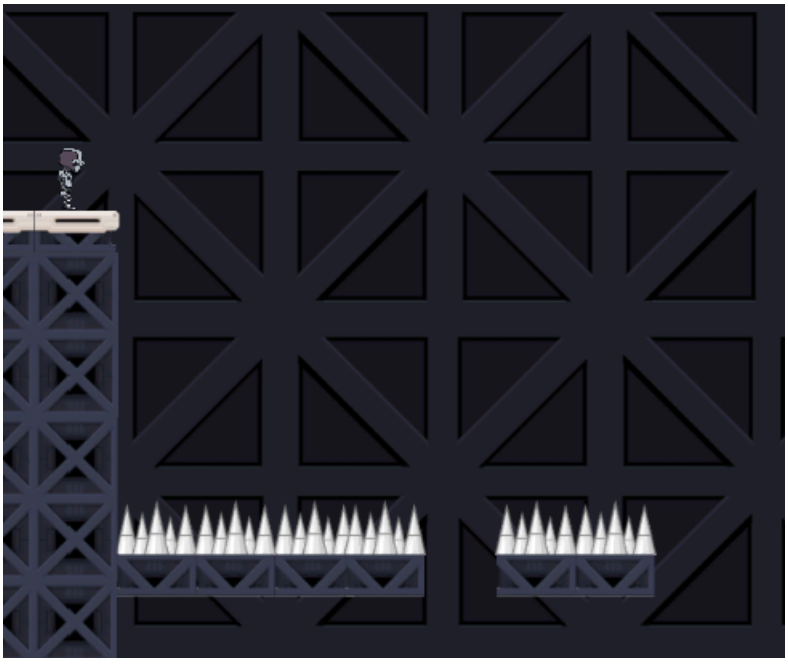


Figure 4.9: Normal Spike

a pillar into a hole surrounded by spikes. We used the normal spike (seen in figure 4.9) to introduce the player to this type of challenge as it is not as often found in platform games as the other challenges we used in this game.

Following the normal spike, we have the medium spike (seen in figure 4.10) that is aiming to be harder than the first one. In this challenge the player will have to jump from a higher ground with a smaller hole in the spike series. In this challenge the player must control the character jump in mid air to be successful. As this is a precision based challenge the hole in the spikes series have collision box as accurate as possible.

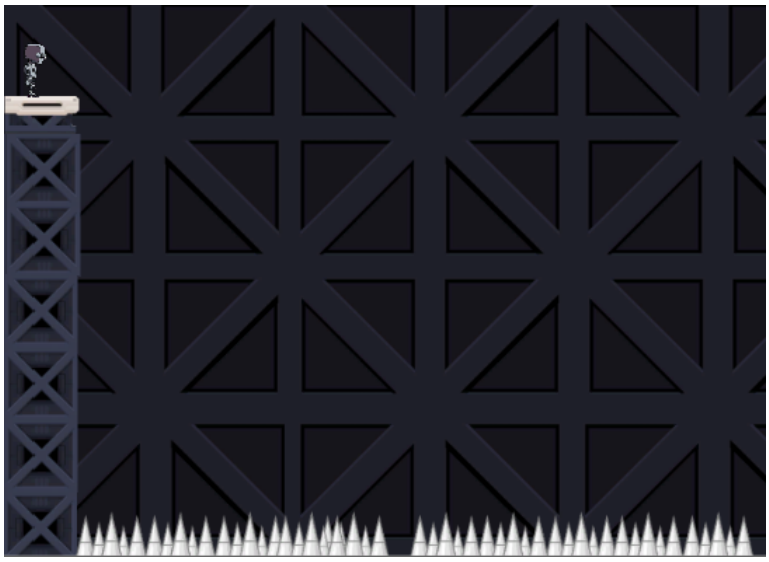


Figure 4.10: Medium Spike

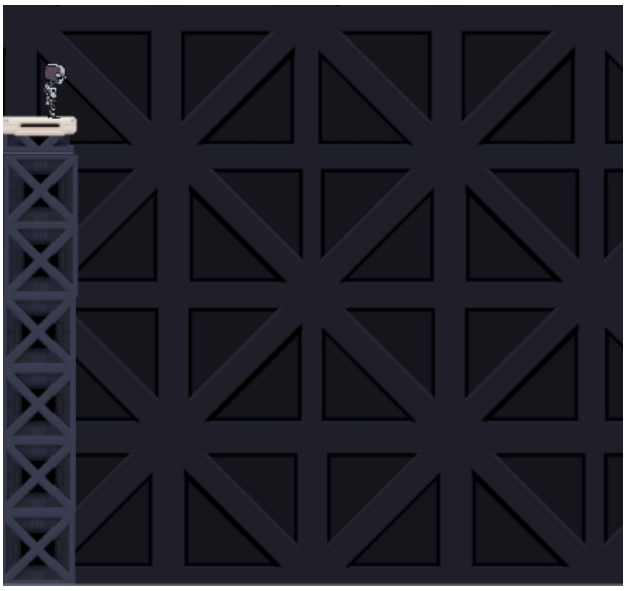


Figure 4.11: Hard Spike

For our last difficulty of this type of challenge, we have the hard spike (seen in figure 4.11). In this one we have a similar setup as the medium spike, jump from the same height. But this time the player will not be able to see the hole in the spike series when performing the jump. In this case the hole will be presented to the player in the beginning of the challenge briefly, the player must memorize the location of the hole or use anything in the scene as reference as where the hole is. This challenge uses both precision and photographic memory skills of the player. The hole in series of spikes is larger than the medium spike one to compensate the fact that the player not knowing the location of the hole.

4.2 Preliminary Testing and Parameterization

Before performing our experiment we tested our game with player to check if the game was delivering a good experience. The game experience that we want to give to the player is a gradual increase of difficulty in the level so the player can enter the flow zone. After we tuned our game, we started adapting our system PLEASED to the game. Firstly we mapped the challenges in our system, so it can calculate the skill and the expectation for each type of challenge. When we explained our system in the previous chapter, we said that when calculating the expectation using the past attempts performed in that challenge we needed to use the last five attempts, and we would calculate the weighted average of this values to find the expectation value. The weight of the values in our system depend on the game we are evaluating so we needed to test our system applied to our game with player, and with their feedback we found the parameterization of the weights of the attempts for our system. Additionally we also tested the value of success of each try that is a number between 0 and 100. In this section we will explain in detail both the game usability tests and parameter fine-tuning of our system.

4.2.1 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 bad experiences from the 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 think 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

During these three iterations of the game, we changed the game according to the player's opinion and the experience that we wanted them to have. The first problem that we had was that the mechanics of the game weren't clear. The player didn't know how to kill enemies as it wasn't clear that they needed so jump on top of them to kill them. So we added a new visual part to the enemy, a red button on top of them and we also created the learning area. In this area, we present an enemy to the player to kill and right on top of the enemy red button we pointed a red arrow at it, so the player understand that he needs to jump to that button in order to kill the enemy. Other game mechanic that was not clear is when we kill a enemy a new platform appear, and the players did not noticed this. So we also introduce this mechanic in the learning area with a another red arrow point to the location where the platform would appear after killing the enemy.

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.

4.2.2 Parameter Fine-tuning

In this section we will presented all the adaptations we did to the system PLEASED so it can evaluate our game. To be able to perform these adaptations we needed the players' feedback. The adaptions we needed to perform were mainly in the expectation calculation. We needed to be able to classify each attempt with a number between 0 and 100 as we explained earlier in this chapter. 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 need this information so we can calculate the player skill and also to calculate the expectation.

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 successfully 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.

$$at_x = \begin{cases} 100, & \text{if successful attempt} \\ 0, & \text{if failed attempt} \end{cases} \tag{4.1}$$

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:

$$f(x) = \begin{cases} 5 & : x = 1 \\ 10 & : x = 2 \\ 15 & : x = 3 \\ 30 & : x = 4 \\ 40 & : x = 5 \end{cases} \tag{4.2}$$

$$w_x = \frac{f(x)}{100} \tag{4.3}$$

$$Expectation = \sum_{n=1}^{5} at_n \times w_n \tag{4.4}$$

4.3 Experimental Process

Our process begin with the player receiving an email with the instructions he has follow in order to complete the experience successfully. These instructions start by stating that the player must take a questionnaire. After the questionnaire is completed the person will receive an email with his unique ID. Afterwards the person can play and select at the end of each level the best and worst parts of this same level. After the game experience, the player must upload the game log files that we talked earlier. These files will be used as inputs of our system PLEASED to generate the graphs that will be compared with what happened in the gameplay experience and the best and worst parts picked by the player.

Hi,

I would like to ask you to participate in my thesis experimentation.

Please follow these instructions:

1. Take this questionnaire: Questionnaire
2. You will receive an email with your **Unique ID** after you completed the questionnaire.
3. Download the game's zip file here: Game
4. After downloading create a new folder and extract the game to there.
5. Double click in the Game.exe that is located in the new folder that you created.
6. Enter the **Unique ID** that you received after completed the questionnaire.
7. Play the game and try your best. The game have two levels, you can click in "finish this level" any time you want, this will take you to next part of the instructions carefully.
8. Upload the XML files (their name start with "challengesM1_" and "challengesM2_" they are in same folder of the Game.exe) to this link: Upload Link

If you can also share my thesis experiment with your friends would be great.
You just need to give them this link:

https://www.dropbox.com/s/cia3nqgeg0jomn5/Game.zip?dl=0

and say that the instructions are in the README file.

Thank you for your participation.
You're awesome.

Best regards,
Bernardo Brás Lourenço

Figure 4.12: Email that the participants received.

4.3.1 Before playing the game

The first step of our experiment is to answer a questionnaire (seen in figure B.1, B.2 and B.3). 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.

4.3.2 Experiencing the game

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.

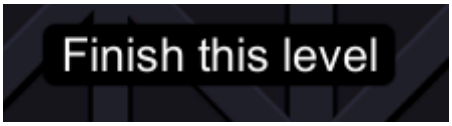


Figure 4.13: Button to give up.

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" (seen in figure 4.14), 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.

4.3.3 Post game analysis

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. In the classification part of the game log we only record the identifier of the challenge either on the liked list or disliked list (seen in figure 4.15).

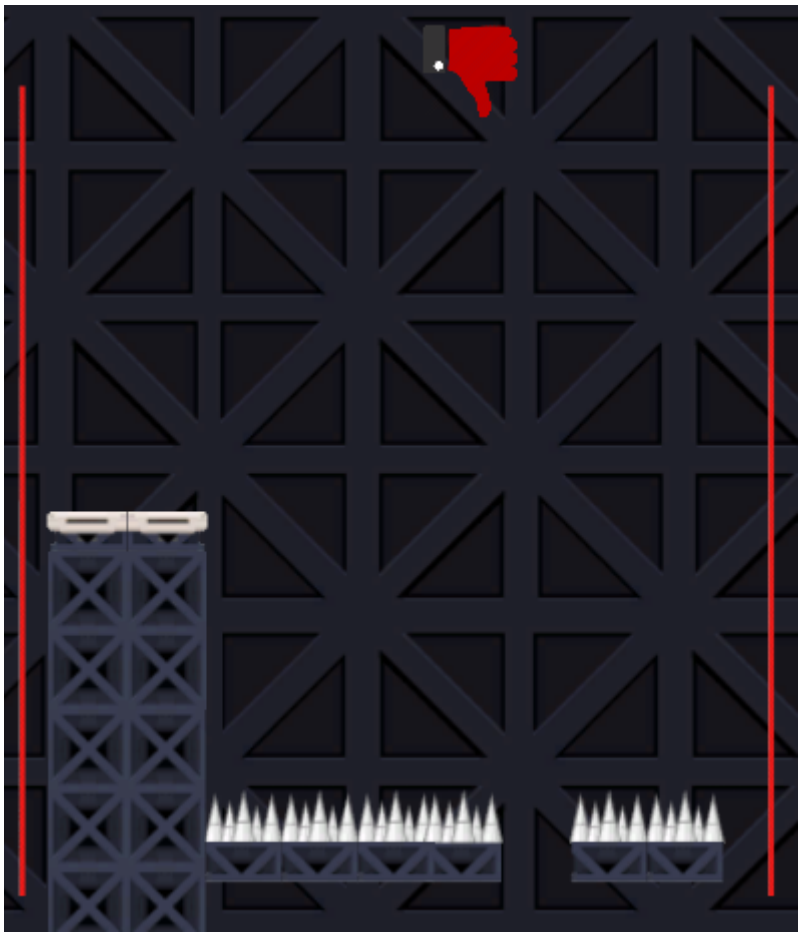


Figure 4.14: Example of a dislike classification of the challenge.

```
<challengesLiked>
  <challengeLiked>
    <id>1</id>
  </challengeLiked>
  <challengeLiked>
    <id>4</id>
  </challengeLiked>
  <challengeLiked>
    <id>7</id>
  </challengeLiked>
</challengesLiked>
<challengesDisLiked>
  <challengeDisLike>
    <id>10</id>
  </challengeDisLike>
  <challengeDisLike>
    <id>17</id>
  </challengeDisLike>
```

Figure 4.15: XML Liked and Disliked Example

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 first test 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 model detected them, testing this way that our model also detect flaws in design in levels that the player did not even played as in this test we are using the skill of the player that was detected in the first level, to simulate the second one. This approach also works if the player did not complete the whole level, in that case the skill will be calculated from the challenges he passed, and the Player Real Mode graph will only have the challenges he passed. On the other hand the Simulation Mode graph of the second level will have all challenges in any situation.

4.4 Summary

In this chapter we presented the game that we used in our experiment, in order to test our system PLEASED. We walked through every challenge type of the game. In each challenge we detailed what was his purpose in gameplay experience. After we presented the study that we performed to test the game's usability. These tests were used to make sure that the players have the game experience, we wanted to have so we can detect it in our system's emotion graphs. After we talked about the adaptations that we did in our system PLEASED so it can evaluate our game and also how we found the our system's parameters to this game thought testing sessions with players. In the final part of the chapter we described our experimental process.

Chapter 5

Results

In the previous chapters we explained how the experience data will be acquired and also how our model works using the personality trait and the skill of a person. In this chapter we will be presenting the results of our experience. The experiment was open to anyone to participate during a period of a week and a half. This experiment was sent to the participant by email and also a link to the experiment was shared on Facebook. The age of the participants was fairly distributed by our age ranges (seen in figure 5.1). We had mostly male participants.

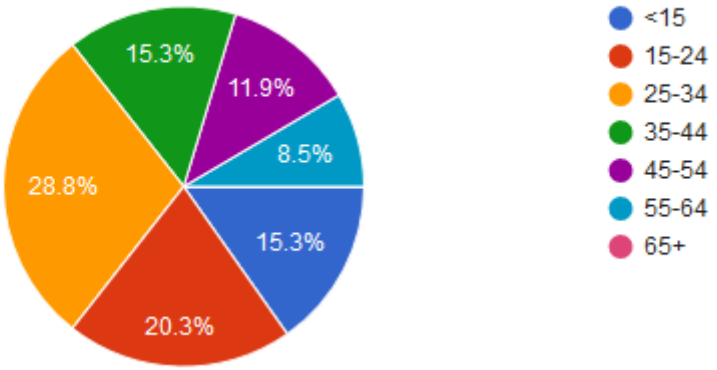


Figure 5.1: Age of the the participants.

Fifty nine responses were sent to us, but only twenty four answers were valid. We considered invalid 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 used both model modes, the Player Real Mode and the Simulation Mode, with the data we collected from this experience. 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

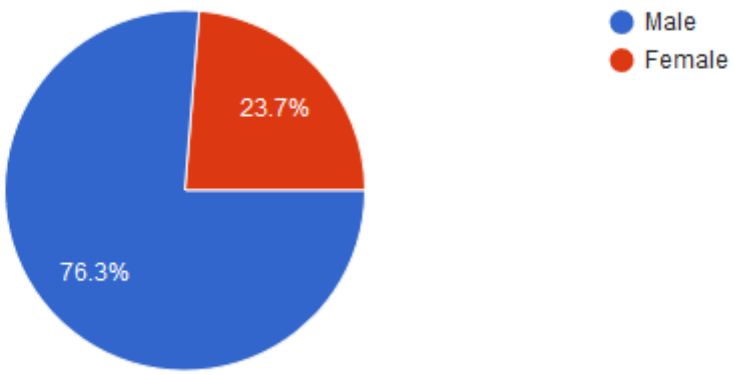


Figure 5.2: Gender of the the participants.

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. We consider that people that answered "I make some time in my schedule to play videogames" in the question "How do you play videogames?" of the questionnaire, are hardcore players, and people that gave other answer than that one ("I don't play videogames" and "I play videogames occasionally when the opportunity presents itself") are considered casual players. So we considered people that usually don't play games as a casual player. 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 spike in the Expected Punishment Emotion. When there is a spike of this emotion, the player start to think that he isn't going to pass the challenge, and start feeling frustrated. We can see how this situation looks like in our graph in figure 5.3.

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.

$$Precision = \frac{CorrectPredictions}{TotalWorstParts} \tag{5.1}$$

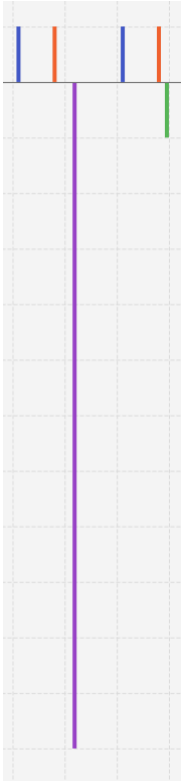


Figure 5.3: Spike in the Expected Punishment Emotion. The purple line represents the Expected Punishment Emotion.

5.1 Player Real Mode Results

In the Player Real Mode we use all the data collected and we use this data to find 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.

5.2 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.

5.3 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.

5.4 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.

5.5 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.

In terms of validations between modes to check components of our system, as we talked in the previous chapter, we have validation of both our skill calculator component and prediction module. This validation

is done by comparing the Player Real Mode from the fist level and the Simulation Mode from also the first level. In this field, we have a 6 per cent difference in precision for hardcore players and 0 per cent difference in precision for casual players, when comparing both graphs. It seems that is needed further improvement in the skill calculation and emotion prediction for hardcore players, but for casual players appear to be working properly. The other validation is between the Simulation Mode of the first level and the Simulation Mode of the second level (this represents a new level). For hardcore player we have a difference of 4 per cent, and for casual players we have 8 per cent difference for hardcore players when comparing both graphs. The difference between this two graph is that, one is prediction a known level where the skill was extracted from, and simulation mode for the second level is a prediction of a whole new level using the skill gathered in the first level. This difference in the precision between both graphs could mean that the skill is still increasing in the second level and our system didn't take that in account when generating the emotions. It means, that our system could need more than one level of gameplay to effectively calculate the skill of a player.

	Player Real Mode	Simulation Mode Level 1	Simulation Mode Level 2
Hardcore Players	82	75	71
Casual Players	100	100	92

Table 5.1: Precision Rates for all the modes.

5.6 Results Discussion

Given the results we presented here, it seems that our system PLEASED is more suitable for casual players. One of the main factors for this, could be what we stated previously that casual players really disliked the worst parts of the level and hardcore players indicate all the parts that they didn't like. It seems that hardcore players have more complex gameplay experiences that our system can not detect. Since our system is more indicated for casual players, it can be a good tool to aid the game design of games that use PCG, and their target audience is casual players. Also games that do not use PCG can benefit from using this system since he had 100 per precision in the modes that didn't predict a new level. On the other hand we tested our system for one game, with two level and a limited number of participants, the precision numbers can change in other games or other experiences. This aspect has to be explored, so we can fully understand the strengths of our system PLEASED.

Chapter 6

Conclusion

This work main objective is to give game designs of games that use PCG, a way to test automatically all the levels generated this way. To accomplish this we need a system that can predict the emotions of a player during the gameplay. These emotions can be related to the enjoyment of the game. Each player has his own experience while playing, that means that different players, could have different emotions to the same situation. This difference in emotions is related to the personality of the player. Also another important component of the player's enjoyment of the game is the skill he possess, passing a level in reasonable amount of time is more motivating than staying stuck, in a part of the level. Staying stuck in a part of a level, could lead to a player giving up on the game. Given that both skill and personality of the player are related with his enjoyment of the game, we can use them to evaluate the player's gameplay experience.

We created a methodology that uses an affective agent to evaluate the game experience of the player through emotions. This affective agent would pass the level gathering the emotions felt by the player. To effectively simulate the player emotions, this affective agent has the skill and personality of the player. To be able to have the skill and personality of the player, our affective agent is composed of an affective agent architecture in combination with a personality model. We need the personality model to be able to introduce the personality into the agent. The skill of the player, we represent in the agent as the rate of success of the agent's actuators. After the affective agent finish each level, we would generate an affective virtual experience that can be used to improve the game we are testing. This approach can potentially be used in any type of game where we can, effectively, replace the player by an affective agent in a gameplay testing session.

To evaluate the effectiveness of our approach, we created a system that we called PLEASED. PLEASED is an implementation of our conceptual model, that uses the emotivector as the agent architecture and the LOT-R as a personality trait, that models the optimism/pessimism of the player. Our system is based on the concept, that the emotion is generated by the different between the expectation of the player and reality. The expectation is calculated be using the history of past actions of the player and his personality trait optimism/pessimism score given by the LOT-R. Reality is simulated using the skill of the player, each attempt outcome of each part of the game is calculated using the probability of the

player pass the challenge (skill). PLEASED receives as input a game log of a gameplay session. This game log contains information of each part of the game, in particular how many tries were made by the player before passing each challenge. PLEASED uses this information to calculate the skill, that is used on the affective agent. After we have the skill of the player and his LOT-R score, we can simulate any level of the game using our system. After the system evaluate the level, it generates a emotion graph that can be read by a game designer to improve the game.

To test our system PLEASED, we created a game from scratch. This game is a side-scrolling 2d platformer, that is capable of creating a game log that can be then used by our system to generate the emotions graph. We designed the level to be similar to a game that uses PCG, as we have a set of challenges and we order them using a rule. This ensures that we have a game similar the real use of this system, a PCG reliant game. After we finished the game, we did some usability tests with players, to ensure that this aspect do not interfere with our experiment, as we could have players complaining about the usability and not the game design itself.

After we finished the game, we needed to adapt our system PLEASED to able to evaluate our game. These adaptations is consists in the configuration of the parameters of the expectation component of our system that can be different from game to game, and also the mapping of the challenges of the game in our system so it can process the game log information. The parameter fine-tuning was performed through tests with players using our game.

In the experiment we asked to the players to play the game, answer a questionnaire (this questionnaire included the LOT-R itens) and point out the best and the worst parts of the game. In this experiment we concluded that we can not identify what emotions are related with the best parts, as they are different from player to player. On the other hand we can identify the worst parts of the level, they are related with the frustration emotion. Saying this we considered the precision in detecting the worst parts of the level, as our system's success rate. As giving up on the game is the extreme situation of a player not liking the game ,we explored this aspect in our experiment by giving the player the option of giving up on the level any time he wants.

Our system was designed to have two modes, so we can separate the system's component that are the emotion modelling, skill calculation and emotion prediction. The Player Real Mode only uses the emotion modelling component of our system. The Simulation Mode uses all components. The precision results of the Player Real Mode represents how well we are modelling the player's emotions. So when we are comparing the precision result of the Player Real Mode with the Simulation Mode for the same level, we can detect potential problems in the skill calculation and emotion prediction components. Our experiment had twenty four valid answers, where fourteen were hardcore players and ten were casual players. For the casual player we had 100 per cent precision in both Player Real Mode and Simulation Mode of the first level, this means that we have the emotion modelling, the skill calculation and predict components of casual player working with a 100 per cent success rate. For hardcore players we had 82 per cent precision in the Player Real Mode and 75 per cent precision in the Simulation Mode of the first level, this means that our emotion generation for hardcore players is good and our skill calculation and emotion prediction is also good, as the perfect score for our skill calculation and emotion prediction would

be to have also a 82 per cent precision in the Simulation Mode of the first level. Despite both modes of our system provide relevant information, the Simulation Mode for the second level is the one that is most important for our work as it represents the emotion prediction of a new level. In this mode we still had different results for hardcore players and casual players, we had a 75 per cent precision for hardcore players and 92 per cent precision for casual players. Given that we can say that our system can predict better casual players and its best suited for games where the casual players are the target audience. For the hardcore player's prediction, we needed more investigation to understand what drive the hardcore players in games.

As we can see, our system is more likely to predict casual players than hardcore player across all the system's modes. This is an aspect that we did not thought in the beginning of this work, that would be a different analyzing a hardcore player or a casual player as they both have emotions. But our experiment showed us that hardcore players dislike more parts of the level than our system can detect. One the explanation for this, is that they have different classification methods compared to the casual players that seem to like the all the parts of the level that are not hard. As we said previously, we talked to some hardcore players that participated in our experiment about the reasoning behind the choice of the best and the worst part of the game and they all said basically the same thing, they did not like the way that particular challenge works even if it is easy to complete. Perhaps we needed to include the player's type (casual or hardcore) in the game type we are testing, (a player can be a casual in a platformer and an hardcore player in a shooter) as an personality trait in our model to better predict the hardcore players.

6.1 Future Work

For future work, we recommend adding more personality traits to our system PLEASED to check what is the impact in the hardcore players prediction as it seems, like we saw in the last section, that something is missing when it comes to predicting hardcore players. Maybe they see game in whole different way than casual players do. Other solution, can be the incorporation of a whole new personalty system using one of the personality models that we discussed in the related work chapter. Other component that can be explored in our system, is the interpretation of the graph generated. In this work we focused in identifying the worst part of the level using the Expected Punish Emotion, but there must be other patterns of emotion or sequences of emotions that can give us important information, if we explore this thematic. Other factor that can help the graph interpretation is the use of colors that are related with the emotion indicated by the graph. For instance, the color red has been associated with excitement, orange has been perceived as distressing and upset-ting, purple as dignified and stately, yellow as cheerful, and blue has been associated with comfort and security [23].

In this thesis, we focused on using our model do help the design of games that uses PCG but our system can also be used in games that do not use it. It would be interesting to see the utility of PLEASED as a real time test system. For example a designer can use our system as he is designing the level to check if he is doing a good job. This seems a good way of using our system if it is viable, and it can be explored in a future work.

Another topic that need further exploration, is the use of our conceptual model in other types of games. With this, we can check if our approach is generic enough to reach other types of games. To do this, we would need to implement our conceptual model with another affective agent architecture and another personality model, and combine both into a new system. But this time, we can add a decision component to our 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. Also we would need to split our new game's levels into challenges. This can prove to the a difficult task for games at first, but using the right mapping between the game actions and the challenges we can make a solid challenge set. For example, in a shooter game we can see the killing of another player as a challenge passed, and the death of ourselves as a failed attempt. After testing multiple games with multiple implementation of our conceptual model, we could then verify what implementation can be used in most games and start working on that one so it can reach more games, and eventually all of them.

Bibliography

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

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

[3] N. Shaker, J. Togelius, and M. J. Nelson. Procedural content generation in games: A textbook and an overview of current research. *Togelius, N. Shaker, M. Nelson—Berlin: Springer*, 2016.

[4] G. Y. Noor Shaker and J. Togelius. Towards automatic personalized content generation for platform games. *Center of Computer Games Research*, 2010.

[5] M. Csikszentmihalyi. Flow:the psychology of optimal experience. *HarperCollins Publishers In: Harper and Row*, 1990.

[6] P.Sweetser and P.Wyeth. Gameflow: A model for evaluating player enjoyment in games. *ACM Computers in Entertainment*, 3(3), July 2005.

[7] R. McCrae and J. Oliver. An introduction to the five-factor model and its applications. *Journal of personality*, 60(2):175–215, June 1992.

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

[9] C. Bateman and R. Boon. 21st century game design. *Charles River Media*, 2006.

[10] R. McCrae and P. Costa. Reinterpreting the myers-briggs type indicator from the perspective of the five-factor model of personality. *Journal of personality*, 57(1):17–40, Mar. 1989.

[11] D. Kersey and M. Bates. Please understand me character and temperament types. *Prometheus Nemesis Book Company*, 1984.

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

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

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

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

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

[17] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence, 2^{nd} edition, 1995.

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

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

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

[21] S. C. Marsella and J. Gratch. Ema: A process model of appraisal dynamics. *Cognitive Systems Research*, 10(1):70–90, 2009.

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

[23] K. NAz and H. Epps. Relationship between color and emotion: A study of college students. *College Student J*, 38(3):396, 2004.

Appendix A

Game Learning Area

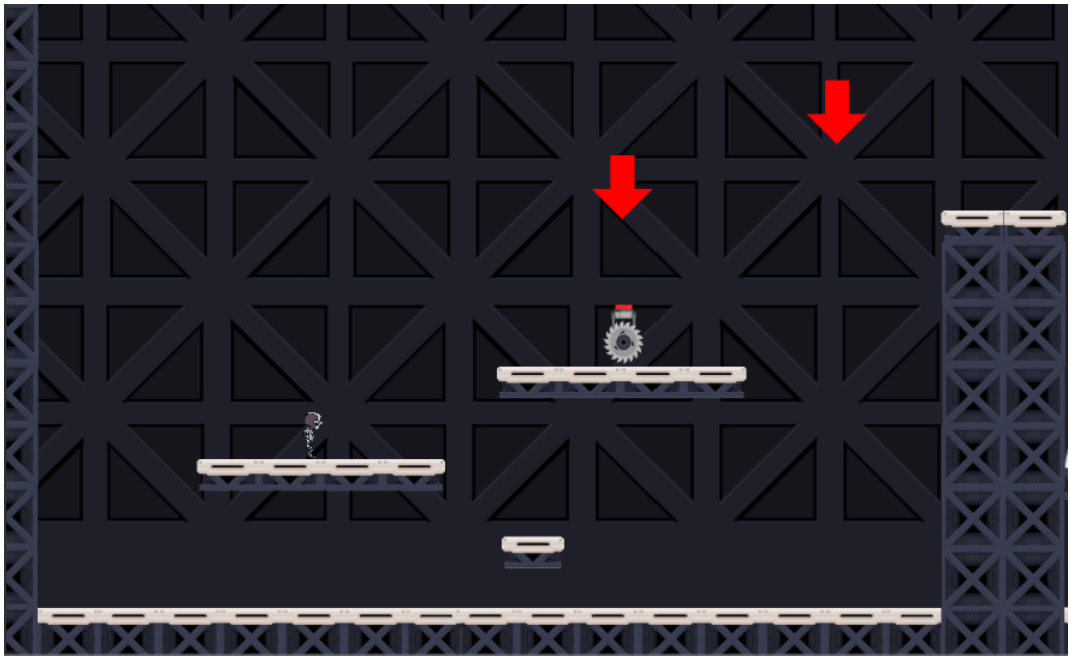
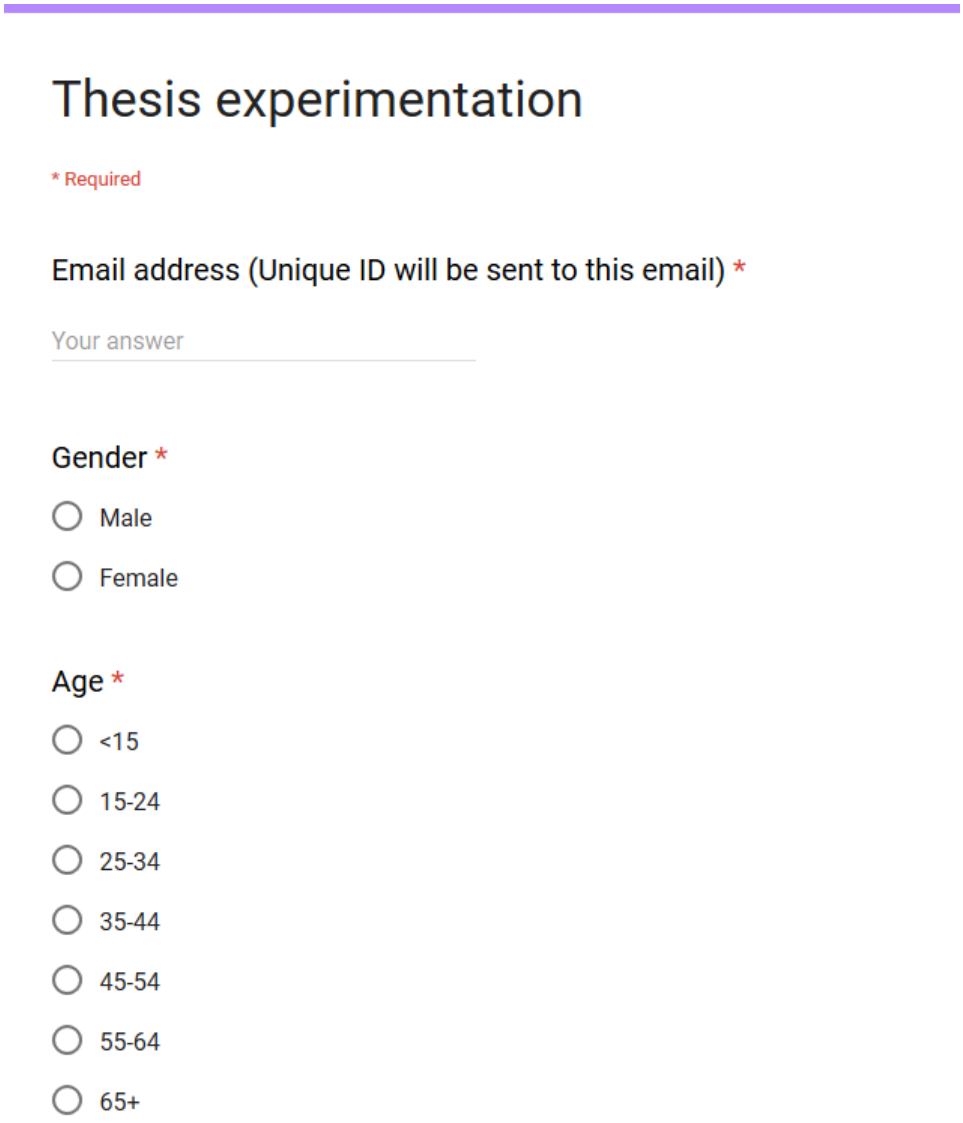


Figure A.1: Learning Area

Appendix B

Questionnaire



Thesis experimentation

* Required

Email address (Unique ID will be sent to this email) *

Your answer

Gender *

- Male
- Female

Age *

- <15
- 15-24
- 25-34
- 35-44
- 45-54
- 55-64
- 65+

Figure B.1: Questionnaire part 1

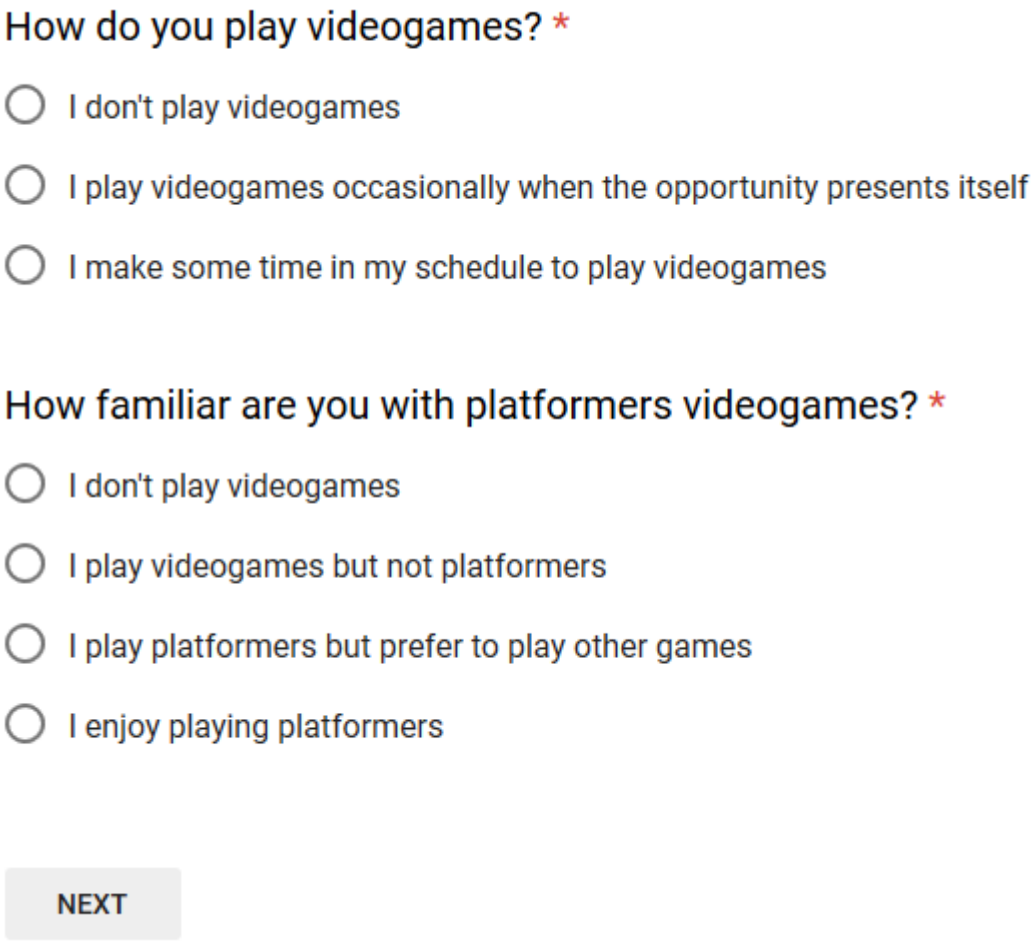


Figure B.2: Questionnaire part 2

About yourself

*

Please be as honest and accurate as you can throughout. Try not to let your response to one statement influence your responses to other statements. There are no "correct" or "incorrect" answers. Answer according to your own feelings, rather than how you think "most people" would answer.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
In uncertain times, I usually expect the best.	○	○	○	○	○
It's easy for me to relax.	○	○	○	○	○
If something can go wrong for me, it will.	○	○	○	○	○
I'm always optimistic about my future.	○	○	○	○	○
I enjoy my friends a lot.	○	○	○	○	○
It's important for me to keep busy.	○	○	○	○	○
I hardly ever expect things to go my way.	○	○	○	○	○
I don't get upset too easily.	○	○	○	○	○
I rarely count on good things happening to me.	○	○	○	○	○
Overall, I expect more good things to happen to me than bad.	○	○	○	○	○

BACK SUBMIT

Figure B.3: Questionnaire part 3