

Towards Player-Driven Procedural Content Generation



Noor Shaker

Center for Computer Games Research

IT University of Copenhagen

A Thesis Submitted for the Degree of

Doctor of Philosophy

December 2012

Abstract

Generating immersive game content is one of the ultimate goals for a game designer. This goal can be achieved if taken into account that players' perceptions of the same game differs according to a number of factors including: players' personality, playing styles, expertise and cultural background. While one player might find the game engaging, another may quit playing as a result of encountering a seemingly insoluble problem. One promising avenue towards optimizing the gameplay experience for individual game players — and thereby attempt to close the *affective loop* in games — is to automatically tailor the game content in real-time. To realize *player-driven procedural content generation* one needs to specify the aspects of the game that have a key influence on the gameplay experience, identify the relationship between these aspects and player experience and define a mechanism for tailoring the game content to each individual needs.

In this dissertation we attempt to address the following research questions towards the aim of generating personalized content for the player: *how can we measure player experience, how can we represent game content, playing style and the in-game interaction, what features should be used to capture player experience and how can they be extracted, how can we model the unknown function between game content, player behavior and affect, how can we generate game content that is tailored to particular player needs and style, how often game content should be adapted, and how the adaptation mechanism can be tested?*

We focus on 2D platform game genre as a testbed for our player-driven procedural content generation framework and we investigate several approaches for generating game content in that game. Crowd-

sourcing experiments are designed to collect gameplay data, subjective and objective indicators of experience from human players: three datasets differing on the number of participants and types of features are collected and analyzed. Computational models of player experience are built on game content, gameplay, and visual reaction features capturing various aspects of the in-game interaction.

Given the high dimensionality of the feature space, a feature selection method is implemented to select the subset of relevant features. Different forms of representation are considered for capturing frequencies, temporal and spatial events. Quantitative measures of player experience are constructed on the crowd-sourced data collected.

As soon as models of player experience are built, a real-time adaptation framework is designed which is guided by the models. The models are used as heuristics in the search of personalized content. Two adaptation mechanisms have been tested in this thesis: the first is based on exhaustive search and the second is based on genetic search. The mechanisms are tested with artificial agents and human players.

The key findings of the thesis demonstrate the ability of the *player-driven procedural content generation* framework to recognize playing behavior differences and to generate player-centered content that optimizes particular aspects of player experience.

Acknowledgements

I thank my supervisors Georgios Yannakakis and Julian Togelius for their support, guidance, encouragement and valuable conversations, without them none of this would be possible. I owe you both a big thanks.

Thanks also to Miguel Nicolau and Michael O'Neill from University College Dublin in Ireland for hosting me there and helping me understand and employ grammatical evolution in my research. I also thank Stylianos Asteriadis and Kostas Karpouzis for their contributions in collecting and analyzing one of the datasets that constitute this thesis.

I am also very grateful to all of those players who participated in my experiments and helped me collect the data I need to conduct my research.

A big thanks goes to my mother, Amal Ballouk, for her encouragement, patient and unwavering faith in me. For without her support and prayers, I would not be where I am today.

Loads of thanks to my sisters, brother, and nephew for their cheerful conversations, love and support. I also thank AbdAllah Nizam who encouraged me to pursue a PhD, without his support and advice, my research path would have been harder.

Last, but not least, I would like to thank my husband, Mohamed Abou-Zleikha, for his constant love, support, encouragement, tolerance and faith in me. He has always known how to make my life brighter.

To Amal, Mohamed, Nada, Nuha and Ousama; my family

Contents

Contents	vi
List of Figures	xiii
1 Introduction	1
1.1 Motivation	2
1.2 Problem Formulation	3
1.3 Questions and Objectives	4
1.4 Challenges	5
1.4.1 Player Experience Modeling	5
1.4.2 Online Content Generation	5
1.5 Our Approach	6
1.5.1 Player Experience Modeling	6
1.5.2 Content Quality	8
1.5.3 Content Representation	8
1.5.4 Content Generator	9
1.6 Summary of Contributions	9
1.7 List of Papers	12
1.8 Outline of This Dissertation	14
1.9 Summary	15
2 Related Work	17
2.1 Theories of Emotion	17
2.2 Computational Models of Emotion	19
2.2.1 Emotions in Games	20

CONTENTS

2.3	Affective Computing and the Affective Loop	21
2.4	Affect Recognition	22
2.4.1	Affect Recognition in Games	23
2.4.1.1	Objective Measures of Affects	23
2.4.1.2	Subjective Measures of Affect	25
2.4.1.3	Fusing Modalities for Affect Recognition	27
2.5	Computational Aesthetics	29
2.5.1	Theories of Computation Aesthetics	29
2.5.2	Patterns in Game Design	30
2.5.3	Categories of Computational Aesthetics	31
2.5.3.1	Aesthetics as Player Experience	31
2.5.3.2	Aesthetics as Player Emotion	31
2.5.3.3	Aesthetics as Style	32
2.6	Procedural Content Generation	32
2.7	Procedural Content Generation in Games	33
2.7.1	Motivation	33
2.7.2	Examples in Commercial Games	34
2.7.3	PCG Middleware	35
2.7.4	Types of PCG	36
2.7.5	Search-based Procedural Content Generation	40
2.7.5.1	Content Representation and Quality	41
2.7.6	Experience-Driven Procedural Content Generation	42
2.7.6.1	Player Experience Modeling	44
2.7.6.2	Personalized Content Generation	46
2.7.6.3	Assessing the Quality of the Personalized Content	48
2.8	Summary	49
3	Tools	51
3.1	General AI Techniques	51
3.1.1	Evolutionary Computation	51
3.1.1.1	Genetic Algorithms	55
3.1.1.2	Genetic Programming	56
3.1.1.3	Grammatical Evolution	56

3.1.2	Artificial Neural Networks	58
3.1.2.1	Single-layer Perceptron	59
3.1.2.2	Multi-layer Perceptron	60
3.1.3	Evolving Artificial Neural Networks	61
3.1.4	Preference Learning	61
3.1.5	Feature Selection	62
3.1.6	Neuroevolutionary Preference Learning	63
3.2	Sequence Mining	65
3.2.1	Definitions	65
3.2.2	Apriori Algorithm	66
3.2.3	Sequential Pattern Discovery: SPADE	67
3.2.4	Generalized Sequential Patterns	68
3.3	Summary	70
4	The Testbed Game	71
4.1	Platform Games	71
4.2	Super Mario Bros	74
4.3	Infinite Mario Bros	77
5	Content Generators	79
5.1	Level Representation	80
5.2	Notch Level Generator	81
5.3	Parameterized Level Generator	82
5.3.1	Content Features	83
5.3.1.1	Basic Parameterized Generator	83
5.3.1.2	Advanced Parameterized Generator	84
5.4	Grammatical Evolutionary Generator	86
5.4.1	Design Grammar	87
5.4.2	Conflict Resolution	90
5.5	Summary	92
6	Expressivity Analysis	94
6.1	Expressivity Analysis	95
6.2	Experimental Setup	95

CONTENTS

6.3	Expressivity Measures	97
6.3.1	Frequency Analysis	97
6.3.2	Linearity	99
6.3.3	Density	100
6.3.4	Leniency	102
6.3.5	Compression Distance	103
6.3.6	Sequential Patterns	104
6.3.7	Histogram comparison	107
6.4	Summary	112
7	Modeling Player Experience	115
7.1	Neuroevolutionary Preference Learning	116
7.2	Feature Extraction	116
7.3	Feature Selection	117
7.4	Model Optimization	118
7.5	Summary	119
8	Data Collection and Feature Extraction	121
8.1	Experimental Protocol	122
8.2	Content Data	123
8.3	Gameplay Data	124
8.4	Player Experience	125
8.5	Head Movement Features	126
8.6	Datasets	127
8.6.1	Dataset 1: Basic Parameterized Generator	127
8.6.1.1	Content Features	128
8.6.1.2	Gameplay Features	128
8.6.1.3	Player experience	130
8.6.2	Dataset 2: Advanced Parameterized Generator	130
8.6.2.1	Direct Features	132
8.6.2.2	Sequential Patterns	133
8.6.2.3	Mining Sequential Features	139
8.6.3	Dataset 3: Behavioral and Visual Cues	143

8.6.3.1	Head Movement Features	145
8.7	Summary	148
9	Player Experience Modeling: Experiments	150
9.1	Correlation Analysis	150
9.1.1	Dataset 2: Advanced Parameterized Generator	151
9.2	Nonlinear Relationships	155
9.2.1	Dataset 1: Basic Parameterized Generator	155
9.2.2	Dataset 2: Advanced Parametrized Generator	157
9.2.2.1	Engagement	159
9.2.2.2	Frustration	163
9.2.2.3	Challenge	165
9.2.3	Dataset 3: Behavioral and Visual Cues	167
9.2.3.1	Player Experience Modeling through Gameplay and Content Features	167
9.2.3.2	Player Experience Modeling through Mean Head Movement Features	168
9.2.3.3	Player Experience Modeling through Visual Re- action Features	171
9.2.4	Fusing Features for Modeling Player Experience	171
9.2.4.1	Modeling through Gameplay/Content and Mean Head Movement Features	173
9.2.4.2	Modeling through Gameplay/Content and Visual Reaction Features	174
9.2.5	Significance Analysis	175
9.2.5.1	Adjusting the Models for Control	179
9.3	Comparison	181
9.3.1	Scalability	181
9.3.2	Modeling Accuracy	183
9.4	Summary	184
10	Game Adaptation	185
10.1	Feature Analysis and Adaptation Frequency	185

CONTENTS

10.1.1	Level Segmentation	186
10.1.2	MLPs Performance on Partial Information	187
10.1.2.1	Analysis	187
10.2	Adapting Game Content	193
10.2.1	Exhaustive Search	193
10.2.2	Evolving Personalized Content	194
10.3	Summary	195
11	Evaluation	196
11.1	AI Agents	196
11.2	Dataset 1: Basic Parameterized Generator	197
11.2.1	Experiment 1: Optimizing Player Experience for a Fixed Playing Style	198
11.2.1.1	Statistical Analysis	201
11.2.2	Experiment 2: Dynamic Adaptation to Changing Playing Styles	203
11.2.2.1	AI Agents	203
11.2.2.2	Human Players	204
11.3	Dataset 3: Behavioral and Visual Cues	204
11.3.1	Optimizing Player Experience for a Fixed Playing Style	205
11.4	Dataset 2: Advanced Parameterized Generator	208
11.4.1	AI Agents: Optimizing Player Experience for a Fixed Play- ing Style	209
11.4.2	Statistical Analysis	213
11.4.3	Discussion	216
11.5	Summary	217
12	Conclusions	218
12.1	Contributions	221
12.2	Limitations and Opportunities	223
12.2.1	Tools	223
12.2.2	Methodology	225
12.2.3	Adaptation Framework	226

12.3 Extensibility	228
12.3.1 Player Experience Modeling	228
12.3.2 Adaptation Methodology	229
12.4 Summary	229
Bibliography	231

List of Figures

1.1	The main components of the player-driven procedural content generation framework (which instantiates the experience-driven PCG framework of Yannakakis and Togelius [2011]) and the approach we follow for each component.	7
2.1	The circumplex model of affect.	20
2.2	The Flow channel (Csikszentmihalyi [1991]).	21
2.3	The self-assessment manikin (SAM) protocol to assess affective dimensions valence (top panel) and arousal (bottom panel) (Bradley and Lang [1994]).	26
2.4	An example of a multimodal emotion system (Nasoz et al. [2003]).	28
2.5	Snapshot from Pandora city in the Avatar film.	33
2.6	Snapshot from The Lord of The Rings film.	34
2.7	Snapshot from the Rogue video game (adopted from wikipedia.org).	35
2.8	Two example creatures created in The Spore game (adopted from gamingprecision.com).	36
2.9	Snapshot from Diablo (adopted from wikipedia.org).	37
2.10	Snapshot from Minecraft (adopted from mojang.com).	37
2.11	Snapshot from Spelunky (adopted from tig.wikia.com).	38
2.12	Three example weapons created in the Galactic Arms Race game for different players.	39
2.13	The main components of the experience-driven procedural content generation (Yannakakis and Togelius [2011]).	43
2.14	The adaptive game system diagram (Charles and Black [2004]).	47

3.1	The general workflow of an Evolutionary Algorithm (Eiben and Smith [2008]).	52
3.2	The general scheme of an Evolutionary Algorithm (Eiben and Smith [2008]).	53
3.3	An example of n-point crossover (a) and a uniform crossover (b). . .	55
3.4	An example of a mutation operator.	55
3.5	Illustrative grammar for generating mathematical expressions. . .	58
3.6	The artificial neuron.	59
4.1	Two examples of 2D platform games. Sub-figures (a) presents a level in Space Panic, the first platform game (adopted from wikipedia.org). Sub-figures (b) presents a level from the platform game Donkey Kong (adopted from arcade-museum.com).	72
4.2	Two examples of 2D platform games.	73
4.3	Two example levels of the original Super Mario Bros game (adopted from ian-albert.com).	76
4.4	Snapshot from Infinite Mario Bros, showing Mario standing on horizontally placed blocks surrounded by different types of enemies.	77
5.1	The geometric representation of the different chunks used for constructing Infinite Mario Bros levels.	80
5.2	Example levels from Infinite Mario Bros generated by the original Notch generator with different seeds and difficulty values.	82
5.3	Enemies placement using different probabilities: high probability is given to placement around horizontal boxes, P_b (a), around gaps, P_g (b), and to random placement, P_r (c).	85
5.4	An example level generated by the parametrized generator using six content features.	86
5.5	The first version of the grammar employed to specify the design of IMB levels.	88
5.6	An example level generated by the first version of the grammar. The design illustrates a number of limitations in the grammar such as the placement of enemies and the generation of boxes.	89

LIST OF FIGURES

5.7	The final version of the grammar employed to specify the design of the level. The superscripts (2, 6 and 10) are shortcuts specifying the number of repetition.	91
6.1	Two example levels generated by the GE-generator using the second version of the grammar.	97
6.2	Average and standard deviation values of eight statistical features that have been extracted from all generated levels across all generators.	99
6.3	Two example levels with different linearity values.	100
6.4	The average and standard deviation values for the expressivity measures for all generators.	101
6.5	Three example levels with different density values.	102
6.6	Two example levels of different leniency values.	103
6.7	Snapshot from a level and the corresponding structure sequence representation.	104
6.8	The histograms of the linearity, leniency and density measures for the 1000 levels generated by Notch generator.	108
6.9	The histograms of the linearity, leniency and density measures for the 1000 levels generated by the parameterized generator.	109
6.10	The histograms of the linearity, leniency and density measures for the 1000 levels generated by the GE-generator.	110
6.11	The histograms of the linearity measure for the 3000 levels generated by Notch generator, the parameterized generator and the GE-generator.	111
6.12	The histograms of the leniency measure for the 3000 levels generated by Notch generator, the parameterized generator and the GE-generator.	111
6.13	The histograms of the density measure for the 3000 levels generated by Notch generator, the parameterized generator and the GE-generator.	112
7.1	The three-phase player experience modeling approach followed. . .	118

8.1	Snapshot from a level and the corresponding platform structure sequence representation, P (a), and enemies and items sequence representation, I (b).	136
8.2	Graphical representation of the different actions that can be performed by the player.	138
8.3	Typical instances of players from Denmark (a,b) and Greece (c,d).	144
8.4	Typical Head Expressivity of player reacting to certain game events.	146
8.5	Visual reactions to game events.	148
9.1	Testing for statistical significance between the obtained performance of the different sets of features examined for modeling player experience. Solid arrows between two feature sets depict a significant difference on the average performance between them. Dash arrows depict average performance differences of no statistical significance. P-values are added next to significant differences.	177
10.1	Testing for statistical significance between the obtained performance of the different segments examined for modeling player experience. Solid arrows between two feature sets depict a significant difference on the average performance between them. Dash arrows depict average performance differences of no statistical significance. P-values are added next to significant differences.	192
11.1	Average and standard deviation values of several gameplay statistical features that have been extracted from 100 different sessions played by the two agents.	198
11.2	Optimized fun levels versus random levels for the two agents.	200
11.3	Optimized fun levels while monitoring the changes in predicting frustration and challenge for the two agents.	202
11.4	Optimized fun levels for the two AI agents	205
11.5	Optimized fun levels for four human players	205

LIST OF FIGURES

11.6	Example levels generated to maximize predicted engagement, frustration and challenge for two human players with different visual reaction features. Sub-figures (a), (c) and (e) are levels generated to maximize engagement, frustration and challenge, respectively for the first player. Sub-figures (b), (d) and (f) are example levels generated to, respectively, maximize engagement, frustration and challenge for the second player.	207
11.7	The fitness function for the optimized levels evolved for the A* agent for each player experience state while monitoring the models' prediction of the other states.	210
11.8	The fitness function for the optimized levels evolved for Sergio's for each player experience state while monitoring the models' prediction of the other states.	211
11.9	The best levels evolved to maximize predicted engagement, frustration and challenge for the two agents. Sub-figures (a), (c) and (e) are levels evolved to maximize engagement, frustration and challenge, respectively, for the A* agent. Sub-figures (b), (d) and (f) are best levels generated to, respectively, maximize engagement, frustration and challenge for the Sergio's agent.	214
11.10	Average and standard deviation values of six statistical content features extracted from 100 levels evolved to optimize predicted engagement (E), frustration (F) and challenge (C) for the two agents. Enemies placement $E_p = 0$ when $P_g = 80\%$, $E_p = 0.5$ when $P_x = 80\%$ and $E_p = 1$ when $P_r = 80\%$	216

1

Introduction

“Video games sit at the confluence of history, technology, and art in such a way that’s found in no other medium, a place where influences from every creative field meet, mix, and recombine.”

– Daniel D. Synder, *the Atlantic*

Differences between humans are undeniable. Providing an engaging human computer interaction given this diversity requires adapting to a person’s tasks, preferences, and abilities (Gajos et al. [2010]). Computer games provide a rich and unique human-computer interaction medium for eliciting, capturing, synthesizing and altering user experience. This thesis explores computational means for generating personalized experiences for the player of a computer game.

The use of computer games in our life has significantly increased in the past decade. According to the US Entertainment Software Association (ESA) [2012] reports in 2012, nearly every device with a screen is used these days for playing games. This has been combined with the demands and production of more creative games leading the game production to be one of the most cutting-edge technology sectors in the U.S. economy. The recent survey in 2012 showed that 49% of the American households own a dedicated game console, with an average game player age of 30 and very similar percentages of male and female players (53% and 47% respectively).

Generating immersive and engaging gaming experiences can be viewed as the holy grail of modern game design and development. This goal cannot be achieved

without acknowledging that players' perception of the same game differs according to a number of factors including players' personality, playing styles, expertise and cultural background.

In this dissertation we address the problem of generating game content that is personalized according to a player's needs. More specifically, we attempt to give answers to the following questions: *how to measure player experience*, *how to identify the relationship between player experience and game content*, and *how to generate player-centered content that is unique and personalized*.

1.1 Motivation

Video games have been a flourishing industry for more than three decades now, with revenues surpassing even those of the movie and music industries (DFC [2012]; Global Movie Production [2012]; The Entertainment Software Association [2012]). Due to their high popularity, highly constrained software requirements but nevertheless immense computational demands, video games have traditionally been introducing leading technologies and pioneering methods in the field of human-computer interaction at large. Today's technologies have reached a point where new middleware can boost the gameplay experience, via the design of game content which is driven by computational models of the player and her experience (Yannakakis and Togelius [2011]; Yannakakis [2012]). To this aim, using *context* and *behavior*-based parameters to elicit information regarding the player's current state (and, consequently, obtain hints about her/his needs regarding interaction) is of primary importance for constructing personal behavioral and interaction-based models. Such models can guide a game adaptation process to achieve maximum engagement or possibly enable conditions of *flow* (Csikszentmihalyi [1991]) and *incorporation* (Calleja [2011]) and, ultimately, realize the affective loop (Höök [2008]; Sundström [2005]) in games.

The demand for automatically generated personalized content increases as more information about the users and their interaction with digital media is becoming available. Automatic content generation is likely to be of great importance for computer game development in the future; both offline, for making the game development process more efficient (design of content such as environ-

ments and animations now consume a major part of the development budget for most commercial games) and online, for enabling new types of games based on player-adapted content.

While game development has been centered primarily on the graphical representation of the game world and the non-player characters' (NPC) behavior, minor focus has been given towards identifying other aspects of game content that elicit particular player experiences and contribute to an engaging experience. Moreover, most of these attempts are theory-driven relying on qualitative approaches and they mostly tend to apply to games in general rather than to specific aspects of games (e.g. see [Koster \[2004\]](#); [Malone \[1981\]](#) among many).

As players tend to vary significantly in their preferences, it would be useful to have an algorithm that could observe a human playing a game and accurately judge what the human is experiencing as he/she is playing. Such a vital information would allow us to adapt the game to the player, and also help us understand how player affect is manifested via behavior.

1.2 Problem Formulation

Mainly motivated by the current lack of a quantitative entertainment formulation of computer games, the need for a better understanding of the relationship between game content and players' affective/cognitive state and the increasing interest in personalized and online (during play) automatic adaptation mechanisms, the focus of the work carried out is on constructing an estimator of players' experience derived from the in-game interaction. This serves as a fitness function for game content generation (content is generated online) providing players new game content based on how they individually have played previously.

In this thesis, we define a novel approach for automatic content generation by linking it with player experience modeling. We explore platform games as a test-bed (proof-of-concept) game for the player-driven Procedural Content Generation (PCG) paradigm.

In this chapter we present the basic components of the framework proposed and discuss the thesis' key contributions within each basic component.

The need of automatic personalized content generation is not limited to games.

The framework proposed is built for games; it can be generalized, however, to other human-computer interaction (HCI) domains. Example systems include recommender systems, web applications and interface design.

1.3 Questions and Objectives

Given the research motivations and goals discussed in the previous section, the following fundamental *objectives* are synthesizing the aims of this dissertation:

1. To construct an accurate indicator of player experience based on the interaction between the player and the game.
2. To apply an online adaptation mechanism that adjusts game content to accommodate to a specific player experience.

The *research questions* that the aforementioned research objectives generate are as follows:

1. How can we recognize players' affect while playing?
2. What are the features from the game content and players' in-game behavior that can help us predict players' affect?
3. How to efficiently represent playing behavior and game content?
4. How to construct models of player experience that can predict players' affective/cognitive state with high accuracy?
5. How can we adapt game content to enrich particular player experiences, and how often the adaptation should be applied?
6. How can we test the adaptation mechanism?

1.4 Challenges

In light of our motivations and objectives, the main challenges we consider in this dissertation are to provide accurate quantitative measures of player experience and to develop efficient tools to automatically tailor content generation to optimize the playing experience. On that basis, we investigate methods for player experience modeling that are capable of capturing and predicting players' affect with high accuracies and we explore techniques for automatic content generation that enable online adjustment of player experience based on the derived models.

1.4.1 Player Experience Modeling

Our hypothesis is that there is an unknown function between game content, player behavior and affect that can be approximated using machine learning techniques. Therefore, features capturing different aspects of the in-game interaction should be extracted and the importance of these features for predicting players' affect should be investigated.

The accuracy of modeling player experience depends to a great degree on the type and representation of features. The size of the feature set is also an important factor and therefore feature selection techniques should be considered for accurate modeling.

Finally, the modeling technique chosen affects both the performance and the interpretation of the relationship between game content, player behavior and reported affect. Therefore, models with powerful approximation capabilities should be examined.

1.4.2 Online Content Generation

Quantitative models of player experience can be used to analyze game-player interaction and the impact of game content on player behavior and experience. More importantly, these models can be further utilized to close the affective loop in games by sensing a player's response to game content, predicting her experience state as estimated by the models and ultimately alter content generation accordingly. As the models can be employed as a fitness function for assessing

game content quality for a particular player, the space of content can be explored and evaluated via these models.

The main issues to consider with personalized content generation are the efficiency and robustness of the adaptation method used. Searching for proper content for a player should be done rapid enough to allow for efficient online adaptation. This might be a challenging task when the size of the search space is rather large.

Having designed an efficient adaptation framework, the main questions remaining are *when* and *how frequently* adaptation should occur? The system should be able not only to identify what changes should be made, but also when these changes should be applied. This requires an analysis of the features that contribute to a specific player experience and an investigation of the size of the game session required to elicit a meaningful experience.

1.5 Our Approach

The general framework we propose to address the aforementioned objectives and challenges is depicted in Figure 1.1. As can be seen from the figure, we identify four basic components for the player-driven procedural content generation framework—which is inspired by the experience-driven PCG framework (Yannakakis and Togelius [2011])—and we present different approaches to implement each component. In the following, we briefly describe each component and refer to the chapter(s) it is discussed.

1.5.1 Player Experience Modeling

Models of player experience are built based on information collected from the interaction between the player and the game. Different types of features capturing different aspects of player behaviors are considered; *subjective* self-reports of player experience are collected by asking players to answer a set of pairwise (ranking) questions presented after playing a pair of game sessions (post-experience self-reports). The questionnaires ask the players to report their preferences of the three affective/cognitive states: fun/engagement, frustration and challenge.

1.5. OUR APPROACH

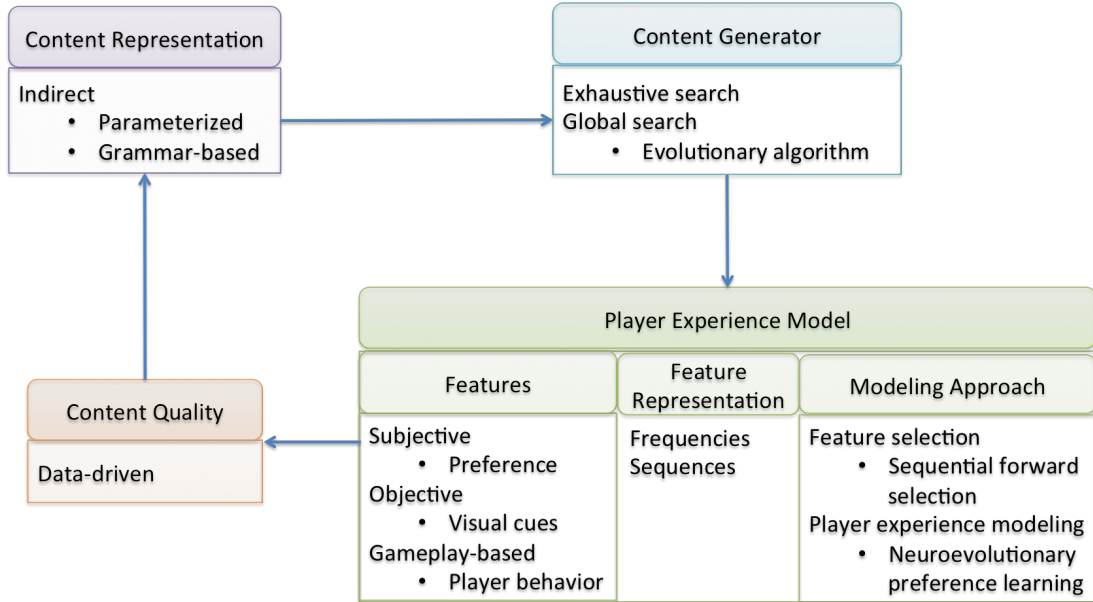


Figure 1.1: The main components of the player-driven procedural content generation framework (which instantiates the experience-driven PCG framework of Yannakakis and Togelius [2011]) and the approach we follow for each component.

For *objective* measures of player experience we analyze video recordings of gameplay sessions and extract information about head movement behavior in reaction to game events. Players' actions while playing the game are also registered and used as *gameplay* features to develop the models.

The different sets of features collected are represented as frequencies describing the number of occurrences of various events or the accumulated time spent doing a certain activity (such as the number of killings of a certain type of enemies or the total amount of time spent jumping), and/or as sequences capturing the spatial and temporal order of events and allowing the discovery of temporal behavior patterns.

Based on the features collected, a modeling approach is followed in an attempt to approximate the unknown function between game content and player behavior. The player experience models are developed on different types and representations of features allowing a thorough analysis of the player-content relationship. The framework proposed for developing the models of player experience is composed of two main steps: (1) given the large size of the player behavior feature sets, a

feature selection method is implemented to pick the minimal subset of features that are relevant for modeling, (2) neuroevolutionary preference learning is then adopted as a modeling approach, the network topology is optimized for best prediction of each reported experience state.

The approach protocol followed to collect the data along with the types of features collected and their extraction and representation methodology are presented in detail in Chapter 8, while our modeling framework is presented in Chapter 7.

1.5.2 Content Quality

A data-driven approach is followed to assess the quality of content by utilizing the player experience models constructed. The models are used to evaluate the content according to its appeal to a particular player given the information about her playing style. Game content features, as extracted from the game sessions, are used in combination with player behavior to evaluate different variations of content. Assessing content quality using the constructed player experience models is discussed in Chapter 9.

1.5.3 Content Representation

The choice of content representation is vitally important since it defines the search space that can be explored and it affects the efficiency of the content creation method. In this dissertation we focus on two different types of content representations. According to the first type, content is represented as a vector of features specifying some properties of a set of chosen content parameters that affect the gameplay experience. A grammar-based representation of content is implemented as a second content representation type. The grammar-based representation allows for greater content variation and more innovative content creation since it imposes fewer constraints on the search space.

The different types of representation proposed, the content generators built based on these representations, and a set of expressivity analysis experiments highlighting the expressive power of each generator are presented in Chapter 5.

1.5.4 Content Generator

Given a content representation, a content generator searches the space of content which is to be evaluated by a content quality measure based on the player experience models. As mentioned earlier, the content is ranked according to the experience it evokes for a specific player and the content generator searches the resulting space for content that maximizes particular aspects of player experience.

Since in our approach two forms of representations are explored, we investigate two techniques for exploring the search space generated by these representations. The dimension of content space created by the parameterized generator is relatively small and therefore exhaustive search is used as a search method providing efficient and fast solutions suitable for online content creation. On the other hand, the content space generated by grammar-based representation is rather large and methods with global search ability are required for effective exploration of such space. For this reason, a stochastic optimization technique by means of evolutionary algorithm is implemented.

More details about these two methods implemented for content adaptation are presented in Chapters 10 and 11.

1.6 Summary of Contributions

Although the work presented in this dissertation is mainly focused on games both in terms of player experience modeling and content generation, we consider the framework proposed and the approach followed to be relevant to other similar HCI applications. The approach is generic and applicable to various research areas such as user modeling, affect recognition, multimodal interaction and content personalization. We consider the following to be the most important contributions of this dissertation:

- **Crowdsourcing Player Experience:** There exist related studies prior to the work presented in this dissertation on modeling player experience (Martinez et al. [2009]; Pedersen et al. [2010]; Yannakakis and Hallam [2007]). In these studies, a similar protocol to the one we are following for data collection and model construction (using feature selection and neuroevolutionary

preference learning) was adopted. However, in most of these studies, the models are constructed based on data collected from a relatively small set of subjects. Fifty-six children in [Yannakakis and Hallam \[2007\]](#), 36 subjects in [Martinez et al. \[2009\]](#), and 120 subjects in [Pedersen et al. \[2010\]](#). Moreover, less complex games were employed for data collection and modeling: simple designed games on the Playware physical interactive game platform were used in [Yannakakis and Hallam \[2007\]](#) and the 3D prey-predator game Maze-ball was employed in [Martinez et al. \[2009\]](#).

The work presented in this dissertation takes this approach a step forward by testing the modeling method applicability when dealing with larger datasets and more sophisticated game context; the method is tested on three datasets of up to 780 participants collected through three new experiments that have been designed for Infinite Mario Bros. Building accurate data-driven models from such big datasets proves the predictive power of the method and its ability to capture large variety of playing styles.

The method we propose for modeling players' interaction with the game can be viewed as constructing quantitative measures of aesthetics in a 2D platform game. However, the view we take on aesthetics of level design is the player's perspective based on the content generated and the gameplay experience it provides.

- **New methods for content generation:** The work presented in this dissertation is built on a preliminary study which focuses on modeling player experience in Infinite Mario Bros ([Pedersen et al. \[2010\]](#)). In that work, generated content is represented as a vector of four content features. Our first contribution here lies in introducing a more advanced content generator with six features capable of creating more variations of game content along other interesting level design. The key innovation is in introducing the use of grammatical evolution for content representation and generation. To the best of the author's knowledge, this is the first time grammatical evolution is used for automatic creation of game content.

To assist the analysis of the content generated by the different generators and demonstrate each generator's weaknesses and strengths, we adopt an

expressivity analysis framework, inspired by the work of [Smith et al. \[2010\]](#). We implement the same expressivity measures proposed by [Smith et al. \[2010\]](#) and we identify three new ones. We further introduce a new method for visualizing the expressive range of a generator that allows comparing it with content spaces created by other generators.

- **New types of features:** The work presented in the literature mainly focuses on modeling the relationship between one type of features of player behavior and subjective player affects ([Martinez et al. \[2009\]](#); [Pedersen et al. \[2010\]](#); [Yannakakis and Hallam \[2007\]](#)). The features used are mostly game-play features ([Pedersen et al. \[2010\]](#)) or objective measures of player experience calculated from physiological signals ([Martinez et al. \[2009\]](#); [Yannakakis and Hallam \[2007\]](#)). In this dissertation, we present, for the first time, the combination of visual characteristics and gameplay features as indicators of player reported experience. Key contributions can be found in (1) the collection of a large corpus of visual and behavioral data of 58 subjects and the use of features extracted from this data to model player experience, and (2) the fusion of visual and behavioral cues for predicting player experience.
- **New methods for feature representation and extraction:** When constructing models of player experience, frequencies of game items and player actions are usually employed for capturing the in-game interaction. This representation method has been widely used and resulted in accurate estimators of player behavior. However, we believe that models with higher prediction accuracies and more expressive power can be constructed by utilizing other forms of content and behavior representation. For this purpose, along with frequencies of items and events, we extract sequences of game content and player behavior and implement sequence mining methods for extracting patterns of players' behavior. In this thesis, we analyze sequences extracted from one modality (game content or player behavior) and bimodal sequences capturing players' behavior in reaction to specific game events.
- **Synthesizing the different approaches for content personalization:**

Closing the affective loop in games has been the focus of many studies recently (Pedersen et al. [2010]; Yannakakis and Togelius [2011]; Yannakakis [2009a]; Yu and Trawick [2011]), but, to the best of our knowledge, there is no complete implementation of a data-driven approach for a well-known game. In particular, scientific contributions can be found in (1) the use of the player experience models as measures of content quality and (2) the implementation of methods that search the content space for eliciting a particular experience for a specific playing style. Moreover, in order to allow for efficient online adaptation, an analysis is performed to set the frequency on which content should be adjusted. A novel combination of several computational intelligence techniques was required to implement and test the adaptation framework.

Our goals have been accomplished by integrating the different approaches of automatic content creation, feature representation, extraction and selection, player experience modeling, and content space search. As an overall contribution, this thesis presents a complete framework that integrates both sophisticated player modeling technologies and efficient search-based procedural content generation algorithms (Togelius et al. [2010d]).

1.7 List of Papers

This thesis is based on experimental research that has already been published in 15 peer-reviewed scientific papers, including four journal articles and one book chapter. This section lists the papers on which the thesis is based, along with the chapter numbers these papers are discussed.

It is worth noting that this dissertation is based on only 9 out of the 15 papers published in order to maintain the focus and the coherency of the key contribution of the thesis. All papers are available for downloading from the author's web site: <http://noorshaker.com>. The papers are as follows:

1. Noor Shaker, Stylianos Asteriadis, Georgios Yannakakis and Kostas Karpouzis. Fusing Visual and Behavioral Cues for Modeling User Experience in Games, IEEE Transactions on System Man and Cybernetics, Special

1.7. LIST OF PAPERS

- Issue on Modern Control for Computer Games, 2012. (under revision). Extensively discussed in Chapters 8 and 9.
2. Noor Shaker, Georgios Yannakakis and Julian Togelius. Crowd-Sourcing the Aesthetics of Platform Games, *IEEE Transactions on Computational Intelligence and AI in Games*, 2012. (to appear). Extensively discussed in Chapters 8 and 9.
 3. Noor Shaker, Georgios Yannakakis, Julian Togelius, Miguel Nicolau and Michael O’Neill. Evolving Personalized Content for Super Mario Bros Using Grammatical Evolution, in *Proceedings of Artificial Intelligence and Interactive Digital Entertainment (AIIDE 12)*, 2012. Extensively discussed in Chapters 10 and 11.
 4. Noor Shaker, Miguel Nicolau, Georgios Yannakakis and Julian Togelius and Michael O’Neill. Evolving Levels for Super Mario Bros Using Grammatical Evolution, in *Proceedings of the 2012 IEEE Conference on Computational Intelligence and Games (CIG 2012)*, 2012. Extensively discussed in Chapters 5 and 6.
 5. Noor Shaker, Georgios Yannakakis and Julian Togelius. Towards Player-Driven Procedural Content Generation, in *Proceedings of Computing Frontier Conference*, 2012. Extensively discussed in Chapter 7.
 6. Noor Shaker, Georgios Yannakakis and Julian Togelius. Digging deeper into platform game level design: session size and sequential features, in *Proceedings of EvoGames: Applications of Evolutionary Computation, Lecture Notes on Computer Science*, 2012. Nominated for best paper award. Extensively discussed in Chapter 10.
 7. Noor Shaker, Stylianos Asteriadis, Georgios Yannakakis and Kostas Karpouzis. A Game-based Corpus for Analysing the Interplay between Game Context and Player Experience, in *Proceedings of the 2011 Affective Computing and Intelligent Interaction Conference (ACII 2011)*, 2011. Extensively discussed in Chapter 8.

8. Noor Shaker, Georgios Yannakakis and Julian Togelius. Feature Analysis for Modeling Game Content Quality, in Proceedings of the 2011 IEEE Conference on Computational Intelligence and Games, 2011. Awarded best student paper. Briefly discussed in Chapter 10.
9. Noor Shaker, Georgios Yannakakis and Julian Togelius. Towards Automatic Personalized Content Generation for Platform Games, in Proceedings of Artificial Intelligence and Interactive Digital Entertainment (AIIDE 10), 2010. Extensively discussed in Chapters 10 and 11.

1.8 Outline of This Dissertation

The dissertation is organized into chapters as follows:

Chapter 2 reviews the state-of-the-art of the different research fields related to the work presented in this dissertation. An extensive review is presented for different areas related to affect recognition and procedural content generation.

Chapter 3 presents a literature review of the tools used throughout the dissertation.

Chapter 4 introduces the testbed game used for our experiences and analysis along with its different variations.

Chapter 5 describes three versions of content generators for our testbed game. More specifically, three approaches for generating game content based on heuristics, parameterized features and grammatical representation have been discussed.

Chapter 6 introduces a framework for analyzing the space of content covered by each generator along six expressive measures and presents a visualization method for comparing the expressive ranges of different generators.

Chapter 7 describes the methodological approach followed in this thesis for modeling player experience. In particular, this chapter includes: neuroevolutionary preference learning as a player experience modeling technique; feature extraction for collecting sets of features that cover variant aspects of game content and player behavior; feature selection for reducing the size of the feature space and model optimization for adjusting the model topology for best performance.

Chapter 8 presents the crowd-sourcing experiments conducted to collect data from human players. The different datasets gathered are introduced each with the specific sets of content and player behavior features. Feature extraction techniques are also described.

Chapter 9 demonstrates the experiments conducted to construct the player experience models from the different datasets. The chapter presents a thorough analysis of linear and non-linear relationships between game content, player behavioral parameters and reported player affects. The use of features from one modality and bimodal features for modeling player experience are also discussed.

Chapter 10 illustrates the adaptation framework implemented for personalizing game content. The frequency of adaptation is analyzed and two data-driven adaptation methods, based on exhaustive search and global search approaches, are presented utilizing the player experience models constructed for evaluating game content.

Chapter 11 investigates the efficiency of the player-driven procedural content generation framework. The methodology is tested using AI agents and human players having different playing styles.

Chapter 11 summarizes the thesis main achievements and contributions and discusses the proposed methodology's current limitations. Moreover, potential solutions that might embrace these drawbacks are presented. In addition, future research steps beyond the limits of this dissertation are discussed.

1.9 Summary

This chapter presents the motivations and objectives of this dissertation. The main questions we are trying to answer in this thesis are presented and the main challenges we consider in terms of player experience modeling and online content generation are discussed. We illustrate the framework we propose for closing the affective loop in games and we give a brief description of its components and our implementation. We highlight our key contributions as categorized in five main areas: *crowdsourcing player experience*, *new methods for content generation*, *new types of features*, *new methods for feature representation and extraction* and *synthesizing the different approaches for content personalization*. We present the

published papers that constitute the work presented in this dissertation and we give a summary of each chapter of the thesis.

2

Related Work

This chapter provides an extensive review of literature on research areas related to the work presented in this thesis. More specifically, we start with an overview of emotion theories and its suitability for computational domains. We review studies on empirical measurements of emotion based on physiological signals, self-reports and multimodal affect recognition interfaces and we present example work from each field. We then present the work done on building computational models of aesthetics with a special focus on the research conducted in computer games. And we finally discuss the procedural content generation framework with its well-known variations, namely: search-based and experience-driven procedural content generation.

2.1 Theories of Emotion

Interest in how to understand, identify, capture, synthesize and influence human emotion has led to research within many disciplines such as psychology, neuroscience, medicine and sociology. Recently, research on emotion has increased significantly in the computer science field due to the technical advancement and the increasing demands for more immersing and engaging experience.

Among the many domains that witness increasing interest in user emotion such as e-commerce, e-learning, news reading, web 2.0 services, and human-computer interfaces; computer games might be the most promising and interest-

ing area due to their rich environment, their capability of delivering immersion experiences and their unique ability to elicit emotion.

Modeling human emotion has a long history. Some of the well-known work in the field includes the component process model framework of emotion proposed by the psychologist Scherer (Scherer [1984]). In his component-processing model of emotion, Scherer suggested five essential components for an emotional experience; psychological arousal, motor expression, behavior preparation, cognitive processes and subjective feeling. An emotional experience is the result of the coordination and the synchronization of all of these components for a short period of time (Scherer [1984, 2002, 2005]). The psychological arousal includes the changes in physical processes such as body temperature and heart rate. Motor expression is the reflection of emotion in processes that people share with others such as facial expressions and gesture. Subjective feeling is the individual experience of an emotional state and the fact that a subject can verbally express her feeling. Behavior preparation states the implications of emotion on the ongoing behavior. Cognitive processes emphasize the effect of emotion on attention and memory (Scherer [2002, 2005]).

Based on his framework, Scherer showed how the different psychological theories of emotion vary quite strongly with respect to focus on specific components and phases in the emotion process (Scherer [2002]).

There are numerous theories of emotion (Cannon [1927, 1931]; Ellsworth [1994]; Pribram and Melges [1969]; Solomon [1980]). However, there is no gold-standard method of how to measure emotion, and in an ideal world of science, one would need to capture all different components of emotion such as the continuous changes in the nervous system, facial and vocal expression, body movements and the subjective experienced feeling state (Scherer [2005]). Theories of emotion, however, remain complex and not easily applicable to computer science. After all, they provide inspiration rather than implementable models (Davidson et al. [2002]; Scherer [2005]; Sundström [2005]).

Despite the difficulties in capturing and measuring emotion, this research area remains extremely interesting for researchers who have studied and developed many methods that allows the incorporation of emotion within digital applications.

2.2 Computational Models of Emotion

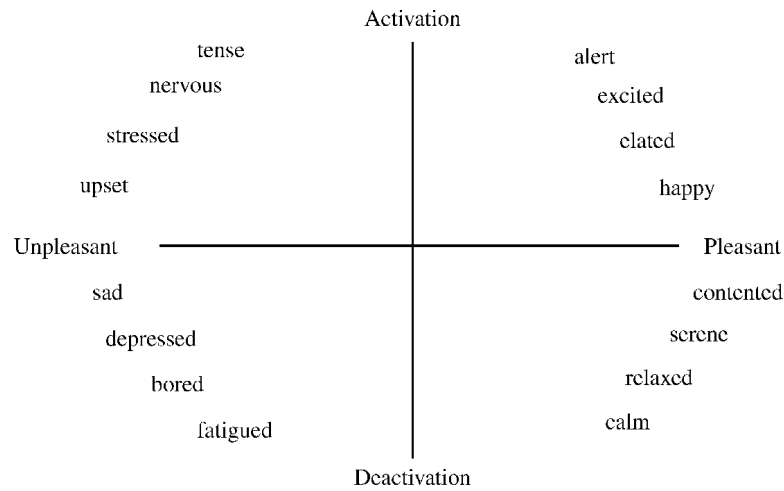
Computational models of emotions are typically inspired by a general theoretical framework of behavioral analysis and/or cognitive modeling (Yannakakis and Togelius [2011]).

Appraisal theory, some times referred to as person-environment relationship (Lazarus [1991]), is probably the most predominant theory of emotion. The central key component in this theory is appraisal, and hence, most of the studies have focused on analyzing the relationship between appraisal variables and elicited emotion (Ortony et al. [1990]; Scherer and Ellgring [2007]; Smith and Scott [1997]).

There are many variations of the appraisal theory and many computational models have been derived from it. In the following discussion we focus on the ones that are well-known and widely used and we give more emphases on those that have been used in game studies.

Ortony et al. [1990] and Elliott [1992] developed a computational model of emotion synthesis, also known as OCC model, that has been adopted by many computational systems (Bartneck [2002]; Conati [2002]; Dias and Paiva [2005]; Reilly [1996]). This model specifies 22 emotion labels that are mapped to appraisal variables. Emotions were defined as positive or negative valence reactions to situations consisting of events, objects and agents. The subject is pleased or displeased with an event, approve or disapprove an action done by an agent and like or dislike aspects of an object. The subject goals, preferences and desires define the valence of her emotional reaction.

The dimensional description model of emotion was proposed as an alternative to the categorical models where latent dimensions are used to characterize affective states (Greenwald et al. [1989]; Russell and Mehrabian [1977]; Watson et al. [1988]). Activation, evaluation, control and other aspects are typically used as dimensions. The evaluation dimension measures how a human feels, while the activation dimension measures the likelihood of taking an action under a given emotional state (Zeng et al. [2009]). A well-known example of dimensional models is the circumplex model of affect by Russell [1980] where emotions were represented as combinations of arousal and valence (Figure 2.1).



Source: Feldman-Barrett and Russell (1998)

Figure 2.1: The circumplex model of affect.

The Belief-Desires-Intentions (BDI) model of emotion, inspired by the work done by Bratman [1999] on human practical reasoning, is another widely used model for emotional displays. This model is mostly used for modeling agent behavior. BDI is a logical model that allows defining and reasoning about BDI agents. *Beliefs* represent the information the agent has about the world which can be incomplete or even incorrect (Rao et al. [1995]; Wooldridge [2000]). *Desires* are the motivational states of the agent and they consist of all the goals the agent would like to accomplish. *Intentions* are a subset of desires the agent has chosen to commit to.

2.2.1 Emotions in Games

There are also other game-specific theories about player emotion. Such theories include the work done by Malone [1981] on identifying what is “fun” in a game and what are the factors for engaging gameplay. In his theory, Malone defines three categories that he claims summarize the factors that make games fun: challenge, fantasy, and curiosity. The theory of flow proposed by Csikszentmihalyi [1991] is given a game-specific interpretation, namely *game flow* (Sweetser and Wyeth [2005]), and has been used as a model for evaluating player enjoyment. According to this theory, for a game to be engaging, a certain level of challenge should be

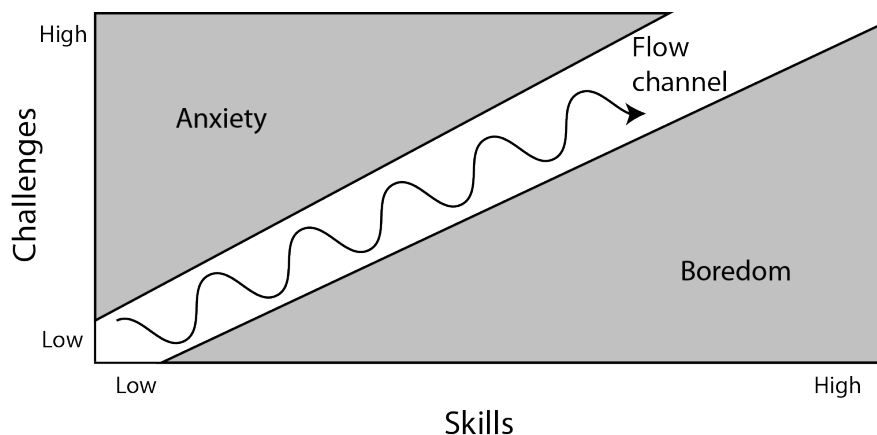


Figure 2.2: The Flow channel (Csikszentmihalyi [1991]).

presented to the player that matches his skills. If the game is too challenging, the player becomes frustrated and if it is too easy the player becomes bored. Keeping the two in balance is the key to the flow experience. Figure 2.2 presents the basic concept of the flow theory. Koster's [2004] theory of fun is based on the concept of learning. For Koster, a game is a learning process and for a game to be fun, it should maintain a balance between the challenges presented and the player's learning rate. Lazzaro [2004] identified four keys for entertaining game-play experience: hard fun, easy fun, altered states and socialization. Read and MacFarlane [2000] proposed three dimensions of fun; endurance, engagement, and expectations. Chen [2007] suggested that the game should keep players of different skill in flow by adapting to various playing styles.

2.3 Affective Computing and the Affective Loop

In computer science, affective computing is the study and development of methods that give the computers the ability to recognize and induce emotion and enable them to interact with humans in human-like ways (Picard [1995]). This requires successfully inferring users' affective state, constructing accurate models of users' affect and correctly expressing emotion.

Estimating affective and cognitive states in conditions of rich human-computer interaction, such as in games, is a field of growing academic and commercial

interest. Closing the *affective loop* (Sundström [2005]) is one of the ultimate aims of the research carried out in the field of affective computing (Leite et al. [2010]; Picard [1995]). The aim of the affective loop idea, is to “couple the affective channels of users closely to those of interactive applications, so that the user’s emotions are influenced by those emotions expressed by or through the application, and vice versa” (Fagerberg et al. [2003]).

Sensing and recognizing players’ emotion have received most attention, while less emphasis is placed on emotion modeling (Hudlicka [2008]). In the following sections we explore the techniques from AI and HCI for recognizing user affective states with a special emphasis on their applicability within computer games.

2.4 Affect Recognition

The last three decades have witnessed increasing interest in automatic human affect analysis (Zeng et al. [2009]). The analysis of the vocal emotion has a long history starting with the work done by Williams and Stevens [1972]. The early attempts to automatically analyze facial expressions were in 1978 by Suwa et al. [1978]. Several authors conducted extensive surveys of work in the machine analysis of affective expressions (Cowie et al. [2001]; Fasel and Luetttin [2003]; Pantic and Rothkrantz [2003]; Sebe et al. [2005]; Tao and Tan [2005]; Zeng et al. [2009]).

There is an abundance of studies presented in the bibliography dealing with the problem of user state estimation during Human-Computer Interaction (HCI) (Duric et al. [2002]; Kapoor et al. [2007]; Lisetti and Nasoz [2002]; Maat and Pantic [2007]). Recent advances on computer vision techniques under uncontrolled conditions have allowed the proposal of techniques incorporating notions such as body and head movements (Asteriadis et al. [2009]), eye gaze (with eye gaze usually necessitating specialized hardware, such as infra-red eye trackers (Jennett et al. [2008])) and facial expressions (Ioannou et al. [2007]). Typical works are those reported in Castellano et al. [2009] and Sanghvi et al. [2011], where the authors utilize Bayesian networking on gaze, postural and contextual data for detecting user engagement with a robot companion posing various expressions (van Breemen et al. [2005]). These attempts, however, may be unsuitable for use

in gaming context because the interpretation of measures used to infer emotion is influenced by the situational context (Gilleade and Dix [2004]; Schachter [1964]).

2.4.1 Affect Recognition in Games

In the domain of games, affect induction is an essential part, since most games can be tweaked in order to make the player experience more expressive and, thus, produce multimodal data that can be analyzed and classified. Kaiser et al. [1998] employed Scherer’s appraisal dimensions (Scherer [1984, 2002, 2005]) in an attempt to analyze emotional episodes. Scheirer frustrates the user on purpose in a general HCI framework in order to produce and record rich affective data (Scheirer et al. [2001]) and Katsis et al. [2008] put this approach to use in the context of car racing games, a popular game paradigm. Wang and Marsella [2006] produced EVG (Emotion eVoking Game), a dungeon role-playing game used to induce emotions related to discrete emotion labels (boredom, surprise, joy, anger and disappointment).

Measuring affect using physiological signals usually requires specialized hardware, which is often expensive and hard to calibrate. As a result, related approaches may be efficient in terms of recognizing player affect, but are extremely problematical to deploy in mass scales and for commercial uses. Nevertheless, physiological data offers a number of advantages over other modalities such as its insusceptibility to social masking of emotions (Kim [2007]) and its reliability making it still favorable in some research areas (Kim and André [2008]; Kim et al. [2004]; Picard et al. [2001]).

2.4.1.1 Objective Measures of Affects

Affect estimation approaches based on processing acceleration data, typically from mobile phones or accelerometer-equipped controllers (e.g. Nintendo’s Wii-mote) or video sequences taken from low-end cameras (e.g. cameras mounted on top of the users’ screen or Kinect sensors, typically sold for Microsoft’s Xbox 360 platforms, but available for desktop computers, as well) utilize hardware that most gamers already possess and do not impose any additional requirements, such as moving in confined spaces, since gamers carry controllers with them

and do not usually move away from their screen or TV while playing. [Buttussi et al. \[2007\]](#) uses acceleration features to deduce motions and actions, besides physiological, in the framework of a fitness game, while [Istance et al. \[2009\]](#) and [Nacke et al. \[2010\]](#) utilize eye-gaze as a means of alternative game control. One of the issues of such approaches is what [Almeida et al. \[2011\]](#) refers to as the ‘Midas touch’ problem, where eye gaze vectors are constantly used to issue commands, regardless of whether the user actually intends to do so or merely looks around at the game interface or is producing irrelevant fixations and saccades. To overcome this, several researchers focus on gamer attention and engagement, as a higher-level cognitive concept, based on eye gaze: [Seif El-Nasr and Yan \[2006\]](#) utilizes a commercial head-mounted eye tracker to identify points on a computer screen and then objects in the game world that attract the user’s attention, while [Isokoski et al. \[2009\]](#) and [Smith and Graham \[2006\]](#) use eye gaze to control virtual game characters. However, these approaches lie in-between those described before, since they do rely on visual features, but require dedicated eye-tracking hardware to produce them. [Kaiser et al. \[1998\]](#) do rely on automatic visual estimation, but concentrates on emotion labels, in order to produce an emotion-rich corpus, and does not delve into game-related concepts such as flow and incorporation. [Melzer et al. \[2010\]](#) investigated the relationship between body movement and affect while playing the Nintendo Wii game *Manhunt 2* ([Rockstar London and Rockstar Leeds and Rockstar Toronto \[2007\]](#)) that encourage rapid aggressive gesture. Their study showed that higher negative affect is observed when body movement is incorporated while playing the game compared to when the game is played using standard controller. In a similar study done by [Isbister et al. \[2011\]](#), the result showed significant correlation between various movement and reported increase of arousal.

The use of context itself is also a very important factor for predicting one’s current state within the frame of gameplay. Within this view, mouse pressure and accelerometers positioned on the player’s back are used in [Van den Hoogen et al. \[2008\]](#). The authors also employ information coming from chair sensors and mappings are created to player’s self reports, as well as actual game difficulty levels. A Gaussian process classification is employed by [Kapoor et al. \[2007\]](#) for detecting moments of frustration in a person-independent scenario. Children

2.4. AFFECT RECOGNITION

were asked to deal with a problem-solving activity, and a multiple-sensor setup was installed. Sykes [2003] conducted an experiment on 10 people, measuring the amount of their finger pressure for creating mappings of arousal to difficulty levels. Hoque et al. [2012] classified elicited expressions of frustration and delight using facial feature tracker and prosodic speech features. Conati [2002] used probabilistic models of users affect that integrates information on the subject skin conductance, heart rate and eyebrow position to improve the effectiveness of educational games through building emotionally intelligent assistant agent. In the work done by McDuff et al. [2011, 2012], the authors analyzed crowd-sourced facial responses data collected over the web and showed different relationships between head gestures, facial expressions, demographics and self-reports in a step towards building a framework for crowd-sourcing emotional responses.

Despite the advancement in psychological research to study theories of emotion and the several attempts to define the mapping between causes and emotional states, recognizing emotions from nonverbal behavior such as facial and vocal expression and physiological indicators remains a hard task in practice (Conati [2002]). This is mainly because of the multimodal nature of emotion which make sensing, recognizing and modeling emotion a hard problem (Hudlicka [2008]).

2.4.1.2 Subjective Measures of Affect

Self-report is currently a widely used measure of affect, where people are asked to freely describe their experience or rate their feeling on a Likert scale. Despite their simplicity in inferring user emotions, these methods have been successfully implemented in a wide range of applications.

There are two main approaches for collecting emotional data through self-reports. In the free response measurement of emotion, researchers usually prepare a set of questionnaires and ask participants to explicitly express their emotional state, as opposite to forced choice response measurement (Scherer [2005]). Two methods were proposed to collect information about users' affect using the forced choice approach (Scherer [2005]); in the first method, the discrete emotions approach, a categorization of discrete emotional states with scales of nominal, ordinal, or interval characteristics is presented to participants. Users are usually

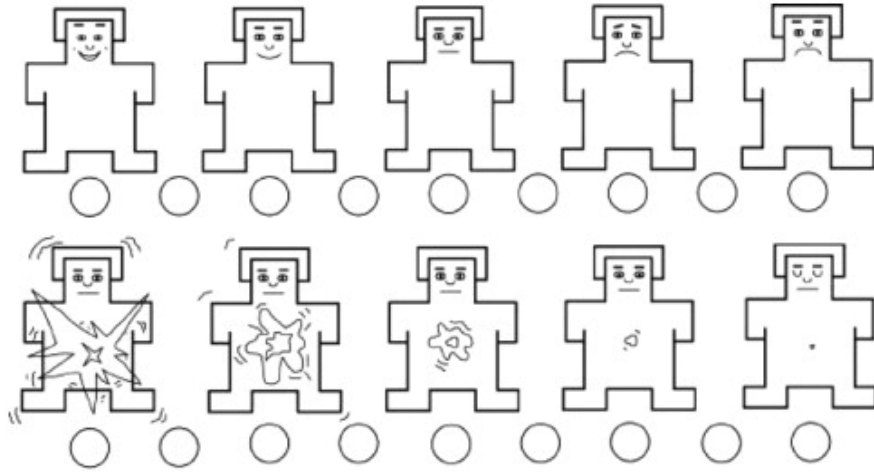


Figure 2.3: The self-assessment manikin (SAM) protocol to assess affective dimensions valence (top panel) and arousal (bottom panel) (Bradley and Lang [1994]).

asked to (1) check terms that best describe the emotion experienced (nominal scale), (2) indicate on a scale whether a respective emotion was experienced a little, somewhat, or strongly (ordinal scale), or (3) use an analog scale to indicate how much an emotion was experienced.

In the second approach, the dimensional approach, participants are usually asked to express their feeling in the valence-arousal space by answering questions about valence (positive–negative), arousal (calm–excited), and tension (tense–relaxed) (Scherer [2005]). The Self-Assessment Manikin (SAM), proposed by Bradley and Lang [1994], is an example tool designed to assess participants emotional experience along the valence-arousal dimension (Figure 2.3). There are a number of drawbacks for each of these methods and there is no universal agreement for which is better (Scherer [2005]; Yannakakis and Togelius [2011]). A recent study conducted by Yannakakis and Hallam [2011b] showed that player self-reports of preferences are consistent and that there are significant in higher order of reporting effects when subjects report via a rating questionnaire compared to the ones observed with preferences.

2.4.1.3 Fusing Modalities for Affect Recognition

According to [Tao and Tan \[2005\]](#), multimodal systems of emotion promises a better emotion detection rate than applications based on one modality. [Figure 2.4](#) presents an example of a multimodal emotion detection system that integrates different modalities. [Tijs et al. \[2008\]](#) use self-reported affect in the 2D space of valence vs. arousal space and mainly physiology-based emotion-related features to distinguish between a boring, frustrating and enjoying game mode while [Rani et al. \[2005\]](#), and [Mandryk and Atkins \[2007\]](#) experiment with challenge during gameplay. [Mandryk et al. \[2006\]](#) reported correlations between self-reported measures of boredom, challenge, frustration and fun and psychophysiological measures of users playing a hockey computer game against a computer or a friend. The In-Game Experience Questionnaire (iGEQ) ([Ijsselsteijn et al. \[2008\]](#)) has been used together with psychophysiological measures, such as Heart Rate, EMG in several studies ([Drachen et al. \[2010\]](#); [Nacke and Lindley \[2010\]](#)) to investigate the correlation between subjective and objective measures in a first-person shooter game. [Mcquiggan et al. \[2008\]](#) created predictors of self-reports of affects based on statistical features extracted from psychological responses (e.g. heart rate and skin conductance) in conjunction with players' self-reports. The authors also used context information such as the visited positions of the avatar and the cursor in a 3D learning environment. [Martínez and Yannakakis \[2010\]](#); [Yannakakis et al. \[2010\]](#) used statistical features about player behavior (such as distance to enemies) and features derived from heart rate, skin conductance and blood volume pulse (average, standard deviation and first and second absolute differences) to construct models of self-reported preferences in a 3D prey/predator game. While these studies, among some others ([Drachen et al. \[2010\]](#); [Mandryk and Atkins \[2007\]](#); [Mandryk et al. \[2006\]](#)) used a combination of context-sensitive (gameplay behavior) and context-free (psychological) features, none of them fused features from different modalities on the time-series level by considering sequential patterns across modalities. The only work done in this area is the recent study conducted by [Martinez and Yannakakis \[2011\]](#) who extracted frequent event sequences across different modalities of user input (physiological signals and gameplay data) using frequent sequence mining techniques and showed that sequential patterns can be

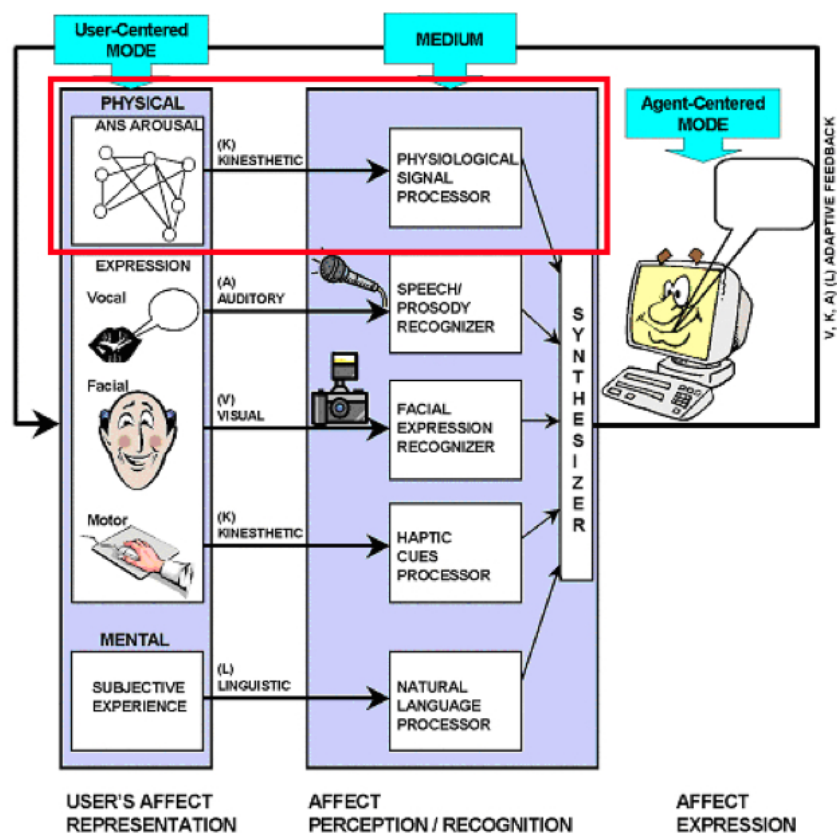


Figure 2.4: An example of a multimodal emotion system (Nasoz et al. [2003]).

used to construct more accurate predictors of user affects than models built on frequency features from one modality.

The comparative affect analysis framework for assessing user's affective state, first introduced by Yannakakis and Hallam [2006]; Yannakakis et al. [2006], has been used in several other attempts to model player affects while interacting with games. The framework proposes a methodology for modeling user affect by letting participants play two variants of a video game and report their preferences along a number of preselected dimensions of emotions. The authors have further tested this framework in an interactive playground using a combination of self-reported preferences and heart rate signals. Tognetti et al. [2010] implemented this framework to estimate player enjoyment preference from physiological signals and subjective self-reports in a car racing game. Martinez et al. [2009] studied correlations between psychophysiological measures (i.e., heart rate, blood volume

pulse, and electrodermal activity) and self-reported values of fun, frustration, and boredom in a simple predator-prey simulation game. The same framework has also been used to model player experience from gameplay data and reported preferences in the 2D platform game Super Mario Bros (Pedersen et al. [2009]). The work presented in (Pedersen et al. [2009, 2010]; Tognetti et al. [2010]; Yannakakis and Hallam [2009, 2006]) shown that accurate estimators of players' affect can be constructed from self-reports.

2.5 Computational Aesthetics

The field of computational aesthetics has recently received increasing interest. When it comes to computer games, the aim of computational aesthetics is to enrich and enhance playing experience (Browne et al. [2012]). To the best of our knowledge, there is no clear definition of computational aesthetics in the literature. As discussed in Section 1.6, in this dissertation, we view aesthetics in games from players' perspective based on how they play the game. We are trying to devise a data-driven approach that can automatically extract game design patterns from existing games.

2.5.1 Theories of Computation Aesthetics

Many analyses of computer games can be found in the literature, both in terms of game mechanics and from a player perspective based on how the player can interact with the game. A number of researchers have attacked this problem from a top-down perspective, that is, by creating theories of the aesthetics of game content and game play based on introspection or qualitative research methods. For example, Malone [1981] proposed that computer games are “fun” when they have the right amount of challenge and evoke curiosity and fantasy, and Magerko et al. [2008] proposed an adaptation framework based on a predefined set of learning styles (refer to Section 2.2 for a comprehensive list of the theory-driven models of emotion).

Such theories are in general too high-level and vague about key concepts to be implemented in algorithms, though some attempts have been made to create

computational models based on them (Togelius et al. [2006]; Yannakakis and Hallam [2005]).

2.5.2 Patterns in Game Design

Other authors have tried to identify more specific and concrete elements of game design and game content that contribute to player experience, so called “patterns in game design”; Björk and Holopainen [2005] are in an ambitious ongoing effort cataloguing hundreds of such patterns, whereas other authors discuss patterns in content design for individual genres, such as first-person shooters. For example, Hullett and Whitehead [2010] analyze some key patterns in first-person shooter games, such as sniper positions and open arenas and discuss how they contribute to player entertainment. In the work done by Moura et al. [2011] and Milam and Seif El-Nasr [2010], a system that visualizes players’ behaviors to allow analysts to easily identify patterns and design issues was presented. Jennings-Teats et al. [2010] showed how player experience could be altered by presenting sequences of level segments ranked by their difficulty and presented to the player according to her behavior. Smith et al. [2008] analyzed platform game levels and proposed a hierarchical ontology for such levels where cells contain rhythm groups which in turn consist of components such as platforms, collectibles and switches. The authors further hypothesize about how certain design choices might affect player experience. It was assumed that short and uneven rhythm groups may result in levels that are more challenging while longer rhythmic sections demand sustained concentration. These principles were eventually incorporated into the Tanagra level generator, which can create levels with rhythmic structure but does not include methods for judging the aesthetics of completed levels (Smith et al. [2010]).

If we find the previously mentioned theories accurate, we can then create *theory-driven* models of game aesthetics. However, even if the theories are correct and sufficiently extensive to allow prediction of player experience in a wide range of situations, they would also need to be quantitative in order to be incorporated within an algorithm, something most current theoretical efforts to understand game aesthetics are not. They would also need to be grounded in measurable quantities. For example, theories based on design patterns would need to be

accompanied by algorithmic ways of detecting and locating such patterns.

The alternative, complimentary approach is to create *data-driven* (bottom-up) models of game aesthetics based on collecting data about games, game content and player behavior.

2.5.3 Categories of Computational Aesthetics

According to Browne et al. [2012], the work done on computational aesthetics in games can be categorized into three main areas: *aesthetics as player experience*, *aesthetics as player emotion* and *aesthetics as style*.

2.5.3.1 Aesthetics as Player Experience

The work done in aesthetics as player experience category revolves around games as an entertaining medium and, therefore, aims at increasing player engagement by capturing and modeling the relationship between player behavior and game context (Browne et al. [2012]). There have been a growing number of studies in the past decade dedicated to this area of research. Gow et al. [2012] described an unsupervised approach for modeling player style based on log data of playing sessions. In the work done by Delalleau et al. [2012] a matchmaking strategy for an online multiplayer game is presented based on data collected about player behavior and preferences. The authors showed that fun, rather than balance, is the important factor in matchmaking systems. An extensive review on player experience modeling is given in Section 2.7.6.

2.5.3.2 Aesthetics as Player Emotion

Capturing and reflecting players' emotional state is the main focus of the work done on aesthetics as player emotion (Browne et al. [2012]). An example work of studies in this category includes the research done by Plans and Morelli [2012] who proposed an experience-driven procedural music generation framework. The authors use a gameplay metric as an indicator of player emotional state that is, in turn, used as a fitness function for adaptive music composition. Savva et al. [2012] implemented a method for automatic recognition of players' emotion from their body movement and investigated whether this information can be

used as a measure of an aesthetic experience. Pedersen et al. [2010] present an approach for modeling player experience from gameplay data, content features and reported player states. Models for predicting reported player experience across six dimensions of emotional states were constructed. The authors further discussed how these models can be used to close the affective loop in games.

2.5.3.3 Aesthetics as Style

Computational aesthetics in the third category is viewed as “the visual presentation and elegance that can be computationally modeled” (Browne et al. [2012]). The research in this area includes the work done by Liapis et al. [2012] who proposed an approach for personalized generation of pleasing spaceship designs based on user’s visual preferences. Similar studies were conducted by Hastings et al. [2009] and Risi et al. [2012]. In the work done by Hastings et al. [2009], the authors use an evolutionary algorithm to automatically evolve personalized weapons for the galactic arms race game based on the content the player liked. Risi et al. [2012] proposed a technique for evolving aesthetically pleasing flowers images and shapes by allowing the user to explore the search space based on their preferences. A step in a different direction was taken by Browne [2012] who explored how a set of game rules can be “elegant”. The author demonstrates how such an intangible concept can be empirically measured and proposed metrics that can be used to evaluate games.

2.6 Procedural Content Generation

Procedural Content Generation (PCG) is a term increasingly used in the production of different types of media. PCG refers to the process of automatically generating content using algorithmic techniques and it has seen a lot of success recently with a wide use in movies and games. The city of Pandora from the Avatar movie (Figure 2.5) featured extensive use of different PCG techniques to generate textures, plants and ground cover. The Lord of the Rings film featured the use of a computer program to automatically create character animation in battles and another program for character rendering (Figure 2.6).



Figure 2.5: Snapshot from Pandora city in the Avatar film.

In the following sections, we focus on exploring the use of PCG techniques in video games domain since this relates to the work presented in this dissertation.

2.7 Procedural Content Generation in Games

Video games rely on different aspects of content such as terrain, maps, levels, rulesets, sounds, stories and mechanics. Manual content generation is expensive (Takatsuki [2007]), time and storage consuming. A research direction that has received increased attention recently is the automatic generation of game content. Procedural Content Generation has been used to generate game content via algorithmic means with or without human designer interference.

2.7.1 Motivation

One of the early reasons for using PCG is to overcome memory limitations of home computers that constrained the space available to store game content. *Elite* (Braben and Bell) is one of the games that solved this problem by storing the seed numbers used to procedurally generate eight galaxies each with 256 planets



Figure 2.6: Snapshot from The Lord of The Rings film.

each with unique properties. A classic example of the early use of PCG is the early eighties' game *Rogue*, a dungeon-crawling game in which levels are randomly generated every time a new game starts (Figure 2.7). Automatic generation of game content is not an easy task; Rogue-like Dwarf Fortress (Adams [2006]) type of games can automatically generate compelling experiences, but they lack any visual appeal. Generating interesting games that emerge the player requires efficient handling of many aspects including the computational power, the aesthetics consideration, creativity of the design and the playability of the generated content (Hendrikx et al. [2011]; Kelly and McCabe [2007]; Smelik et al. [2009]).

2.7.2 Examples in Commercial Games

Procedural content generation has witnessed increasing attention in commercial games. *Diablo* (Blizzard North [1997]) (Figure 2.9) is an action role-playing hack and slash video game featuring procedural generation for creating the maps, the type, number and placement of items and monsters. PCG is one of the central mechanic used in *Spore* (Maxis [2008]) where the design the players create is animated using procedural animation techniques (PCG Wiki). These personalized creatures are then used to populate a procedurally generated galaxy. Figure 2.8 presents two example creatures created by different players. *Civilization IV* (Fi-

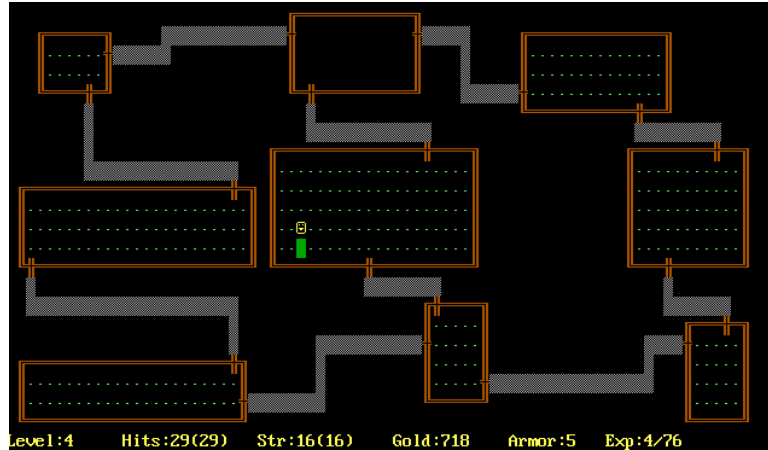


Figure 2.7: Snapshot from the Rogue video game (adopted from wikipedia.org).

raxis Games [2005]) is a turn based strategy game that allows unique gameplay experience by generating random maps. *Minecraft* (Mojang [2011]) is one of the recent popular indie games featuring extensive use of PCG techniques to generate the whole world and it's content (Figure 2.10). *Spelunky* (Yu and Hull [2009]) is another notable 2D platform rogue-like indie game that utilizes PCG to automatically generate variations of the game levels (Figure 2.11). *Tiny Wings* (Illiger [2011]) is yet another example of a 2D game featuring procedural terrain and texture generation system giving the game a different look with each replay.

2.7.3 PCG Middleware

Various PCG techniques such as *SpeedTree* and fractal generation have been used recently by game designers to generate game entities such as terrain and game characters. *Euphoria* (NaturalMotion [2007]) is a game animation engine that was developed to automatically generate real-time animation allowing more action diversity. *SpeedTree* (Interactive Data [2011]) is a middleware used to procedurally generate wide variety of trees. *CityEngine* (Mueller et al. [2011]) is a 3D modeling software that uses procedural modeling techniques for creating cities and buildings.



Figure 2.8: Two example creatures created in The Spore game (adopted from gamingprecision.com).

2.7.4 Types of PCG

There are a number of independent facets that define the type of a PCG technique:

- Online versus offline: PCG techniques can be used to generate content online, as the player is playing the game, allowing the generation of endless variations, making the game infinitely replayable and opening the possibility of generating player-adapted content, or offline during the development of the game or before the start of a game session. The use of PCG for offline content generation is particularly useful when generating complex content such as environments and maps. *Left 4 Dead* (Valve Corporation [2008]) is a recently released first-person shooter game that provides dynamic experience for each player by analyzing player behavior on the fly and alter the game state accordingly using PCG techniques (Booth [2009]). *NERO* (Stanley et al. [2005]) is an example of the use of AI techniques to allow the players to evolve real-time tactics for a squad of virtual soldiers. *Forza Motorsport* (Microsoft Game Studios [2005]) is a car racing game where the Non-Player Characters (NPCs) can be trained offline to imitate the player driving style and can later be used to drive on behalf of the player.
- Necessary versus optional: PCG can be used to generate necessary game



Figure 2.9: Snapshot from Diablo (adopted from wikipedia.org).



Figure 2.10: Snapshot from Minecraft (adopted from mojang.com).

content that are required for the completion of a level, or it can be used to generate auxiliary content that can be discarded or exchanged for other content. An example of optional content is the generation of different types of weapons in first-person shooter games or the auxiliary rewarding items in Super Mario Bros (Nintendo Creative Department [1985]).

- Random seeds vs. Parameter vectors: The generation of content by PCG can be controlled in different ways. The use of random seed is one way to gain control over the generation space, and another way is to use a set of parameters that control the content generation along a number of dimensions. Random seeds were used when generating the world in *Minecraft* (Mojang [2011]) which allow regenerating the same world if the same seed is used (Minecraft Wiki). A vector of content features was used in Shaker et al. [2010] to generate levels for *Infinite Mario Bros* (Persson) that satisfy

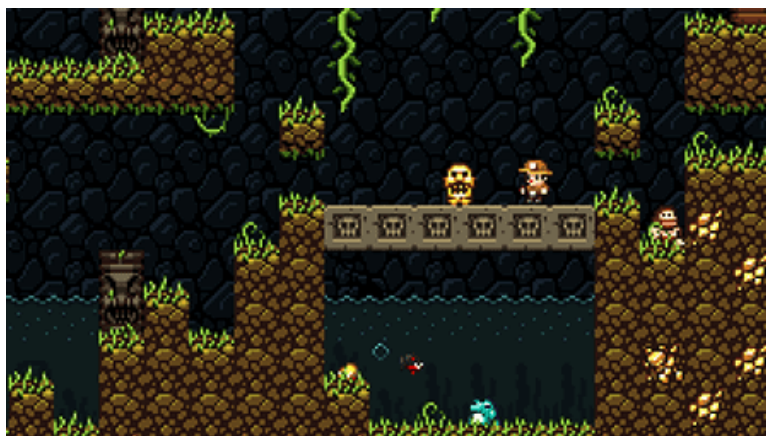


Figure 2.11: Snapshot from Spelunky (adopted from tig.wikia.com).

the feature specification.

- Generic versus adaptive: Generic content generation refer to the paradigm of PCG where content is generated without taking player behavior into account as opposite to adaptive, personalized or player-centered content generation where player interaction with the game is analyzed and content is created based on player's previous behavior. Most of the attempts that can be found in commercial games tackle PCG in a generic way, while adaptive PCG has been receiving increasing attention in academia recently. Yannakakis and Togelius [2011] provide an extensive review of player-driven PCG). *Left 4 Dead* (Valve Corporation [2008]) is an example of the use of adaptive PCG in a commercial game where an algorithm is used to adjust the pacing of the game on the fly based on player's *emotional intensity*. In this case, adaptive PCG is used to adjust the difficulty of the game in order to keep the player engaged (Booth [2009]). Adaptive content generation can also be used with another motive such as generating more content of those the player seems to like. This approach was followed in the *Galactic Arms Race* (Hastings et al. [2009]) game where the weapons presented to the player are evolved based on her previous weapon use and preferences. Figure 2.12 presents examples of evolved weapons for different players.
- Stochastic versus deterministic: Deterministic PCG allows the regeneration

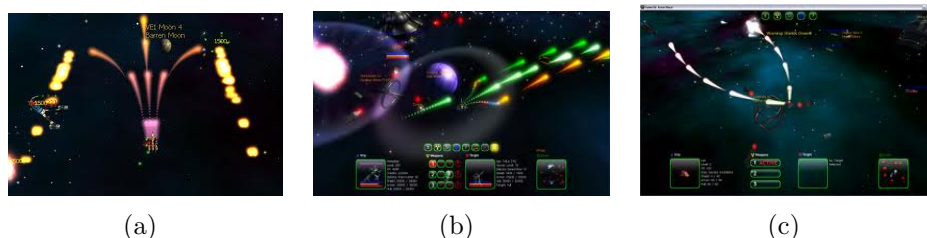


Figure 2.12: Three example weapons created in the Galactic Arms Race game for different players.

of the same content given the same starting point and method parameters as opposite to stochastic PCG where recreating the same content is usually not possible.

- Constructive versus Generate-and-test: In constructive PCG, the content is generated once and modifications are not permitted, e.g. rough-like games. Generate-and-test PCG techniques, on the other hand, go through the loop of generate-test for a number of times until a satisfactory solution is generated. *Yavalath* (Browne [2007]) is a two players board game generated completely by a computer program using the generate-and-test paradigm (Browne and Maire [2010b]).
- Algorithmic versus mixed authorship: Up until recently, PCG has allowed limited input from game designers who usually tweak the algorithm parameters to control and guide content generation while the main purpose of PCG remains the generation of infinite variations of playable content (Adams [2006]; Blizzard North [1997]; Browne and Maire [2010b]; Yu and Hull [2009]). A new interesting paradigm, however, has emerged recently focusing on incorporating designer and/or player input through the design process. In the mixed-initiative paradigm, a human (designer or player) cooperate with the algorithm to generate the desired content. *Tanagra* (Smith et al. [2010]) is an example of a system where the designer draws part of a 2D level and a constraint satisfaction algorithm is used to generate the missing parts while retaining playability. Another example is *SketchaWorld* framework (Smelik et al. [2010]) which is an interactive procedural sketching

system for creating landscapes and cityscapes where designers can manually edit and tune the generated results while the virtual world model is kept consistent. Liapis et al. [2012] utilizes players' choices as a fitness function for evolving personalized spaceship designs.

In the following sections, we explore the state-of-the-art of PCG and we focus on the methods and games developed in this research field. Emphasis is given on the procedural generation of different game aspects such as maps, levels, rulesets, stories and dialogs while less weight will be put on decorative assets such as textures and sound effects since they are not directly linked to the work presented in this dissertation. A special focus is given to two paradigms of PCG; Search-based Procedural Content Generation and Experience-Driven Procedural Content Generation since they provide the umbrella under which most of the work on PCG, including this work, is developed.

2.7.5 Search-based Procedural Content Generation

Search-based procedural content generation (SBPCG) (Togelius et al. [2010d]) is the field of PCG that uses global stochastic search algorithms and fitness functions to measure the quality of the generated content. The SBPCG approach has been used extensively for evolving different aspects of game content: levels for platform games (Jennings-Teats et al. [2010]; Pedersen et al. [2010]; Shaker et al. [2010]; Sorenson et al. [2011]), tracks for car racing games (Cardamone et al. [2011b]; Loiacono et al. [2011]; Togelius et al. [2006]), the distributed evolution of weapons in a space shooter game (Hastings et al. [2009]), maps for real-time strategy (RTS) games (Togelius et al. [2010b]) and for first-person shooter games (Cardamone et al. [2011a]) and rulesets for mini-games (Smith and Mateas [2010]). The generation of 2D mazes was explored by Ashlock et al. [2011, 2006]. In Johnson et al. [2010] self-organization capabilities of cellular automata are used to generate, in real time, cave-like maps. Spaceship designs were evolved in Roberts and Lucas and Liapis et al. [2011]. Mahlmann et al. [2012] used a genetic algorithm to generate balanced rules for card games. Unit types for strategy games were evolved in Mahlmann et al. [2011]. Giannatos et al. [2011] improved interactive story by suggesting new events within the story world using evolutionary optimiza-

tion. Giannatos et al. [2012] investigated generating stories for suspense using a combination of AI planning and evolutionary optimization methods. Martin et al. [2010] investigated the use of interactive evolution to design 3D building structures. Hom and Marks [2007] proposed automatically designed game rules to evolve a balanced board game. Togelius and Schmidhuber [2008] evolved rule-sets for simple Pacman-style games. Automatic game generation was explored by Browne and Maire [2010a] using evolutionary techniques to automatically generate and evaluate playable board games. The generation of complete playable games was explored by Cook et al. [2012] in a system called *ANGELINA* that uses co-operative co-evolution to automatically evolve simple platform games. Kersemakers et al. [2012] investigated the use of SBPCG techniques to generate content generators for a 2D platform game.

2.7.5.1 Content Representation and Quality

Content representation and quality measures are two vitally important issues that should be considered when automatically generating game content. The form of representation chosen plays an important role in the efficiency of the generation algorithm and the space of content the method will be able to cover. The choice of proper representation also depends on the type of problem one is trying to solve. In (Shaker et al. [2010]), a vector of integers representation of the content space is used; a vector of four content features was employed as a representative of selected features for each level in *Infinite Mario Bros* (Persson) and an exhaustive search method was implemented to search the content space. Models of players experience were utilized as a measure of content quality. In a later study on the same game (Shaker et al. [2012]), the structure of the levels for the same game was described in a Design Grammar that was employed by Grammatical Evolution (O'Neill and Ryan [2001]) which is used to evolve level design. The fitness function used measures the appeal of the resulted design for specific player preferences (Shaker et al.) (an extensive discussion and analysis of the work done in these papers are presented throughout the thesis). A similar representation was proposed in Sorenson et al. [2011], also on the same game, where levels were described as a list of design elements placed in 2D maps, but in this study

standard genetic algorithms in combination with constraints satisfaction were used to evolve and measure content quality.

In another study for evolving tracks for a car racing game, [Cardamone et al. \[2011b\]](#) used a set of control points that the evolved track has to cover and Bezier curves were employed to connect these points and ensure smoothness, a method inspired by the work done on the same game domain by [Togelius et al. \[2007\]](#). The two studies, however, employed different fitness functions: while interactive evolution is used by [Cardamone et al. \[2011b\]](#) by asking users to score the generated tracks, a fitness function that aims at maximizing measured entertainment for individual players was used by [Togelius et al. \[2007\]](#). In another study by [Loiacono et al. \[2011\]](#), the focus is given to evolve tracks with a large degree of diversity.

Multi-objective evolution was used by [Togelius et al. \[2010b\]](#) to evolve maps for *StarCraft* ([Blizzard Entertainment and Mass Media \[1998\]](#)). Two forms of representation were exploited: an indirect representation was used for searching (an array of different map elements) and a direct representation for quality testing. Different fitness functions that measure a set of desired characteristics of a good map were proposed to evaluate the maps generated. Four different map representations, having different levels of directness, were implemented in [Cardamone et al. \[2011a\]](#) to generate maps for a first-person shooter game. A fitness function that measures the average fighting time for each generated map, as an indicator of an interesting design, was used.

2.7.6 Experience-Driven Procedural Content Generation

[Yannakakis and Togelius \[2011\]](#) proposed a framework for closing the affective loop in games. The framework, called Experience-Driven Procedural Content Generation (EDPCG), defines an approach in which game content is generated according to player experience. EDPCG is synthesized by four main components (Figure 2.13) that communicate to optimize player experience: (1) a *Player Experience Modeling* (PEM) component that defines the relationship between game content and player experience (measured by cognitive or affective state responses, gameplay behavior or both); (2) a *Content Quality* component which measures

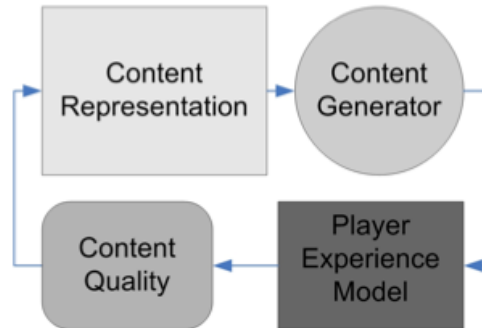


Figure 2.13: The main components of the experience-driven procedural content generation (Yannakakis and Togelius [2011]).

the quality of the generated content based on the PEM; (3) a *Content Representation* component that defines an approach for representing game content in a way that maximizes efficiency, robustness and performance; and finally, (4) a *Content Generator* component which explores the content space, given the content representation, searching for the content that optimizes player experience as reported by the PEM.

The main novelty of EDPCG over traditional search-based approaches (presented in Section 2.7.5) is that it utilizes player experience models for the assessment of content quality.

Optimization of game aspects based on empirically derived models has so far been focused on dynamic game balancing or Dynamic Difficulty Adjustment (DDA) that uses an automatic algorithm for changing parameters, scenarios and behaviors of NPCs. Different techniques for different game genres have been investigated. Research into dynamic scripting focuses on dynamic game balancing (Lee and Jung [2006]; Spronck et al. [2004, 2006]) but is only suitable for games that are scripted or imply storytelling. Lee’s and Jung’s [2006] work on dynamic scripting for a shooter game utilizes a Gaussian Mixture Module that models the players reaction pattern. Spronck et al. [2004, 2006] focused on dynamic scripting using reinforcement learning to control the movement of the NPC. Andrade et al. [2005a,b] also used reinforcement learning to modify NPC behaviors. Studies on a modified version of the Pac-Man game (Yannakakis and Hallam [2005, 2004b]) focus on the NPC cooperative behavior using an online neuroevolution learn-

ing mechanism; the proposed methodology was further examined by applying it to predator/prey game (Yannakakis and Hallam [2004a]). These attempts are mainly focused on adjusting the game difficulty by adapting the NPCs behavior assuming that challenge is the most important factor contributing to enjoyable gaming experiences. They did not, however, change the game environment or adjust the difficulty of the game level during play (except the work done by Yannakakis and Hallam [2004a]), and no particular emphasis is given to the content of the game and its impact to the player’s affective state.

The attention has been shifted in the last decade with more work being published focusing on adapting other aspects of game content. Most of these attempts, however, adopt a qualitative fitness function based on theories of player experience (refer to Section 2.2 for an overview of the computational models of emotion). Example studies include Mahlmann et al. [2011, 2012]; Sorenson and Pasquier [2010]; Togelius and Schmidhuber [2008]; Togelius et al. [2006, 2010c], most of which were previously discussed in Section 2.7.5.

Few attempts can be found on incorporating players’ emotions into the game in a closed-loop manner where player’s emotion is actively manipulated to ensure engagement (Hudlicka [2008]).

In the following sections we investigate the player experience modeling and the content quality components of the EDPCG framework as content representation and generation have already been reviewed under the SBPCG section (Section 2.7.5).

2.7.6.1 Player Experience Modeling

Constructing models of player experience is the first step towards realizing the affective loop in games. Houlette [2003] was the first to explicitly define player modeling as models of individual players derived from data collected from the interaction between the player and the game. These models can then be used by game AI to adapt itself. Later, in the work done by Smith et al. [2011], the authors defined four independent facets for describing a player model: the *scope* of application and who is being distinguished in the model, the *purpose* of the model which define how the model is to be used, the *domain* of modeled details

and what it describes, and the *source* of a model’s derivation or motivation. In this thesis, we take the view of player experience modeling described by Yannakakis and Togelius [2011]. The authors of the “*Experience-Driven Procedural Content Generation*” framework identified four classes of approaches for PEM in games: *subjective PEM* that can be derived from explicit player reports; *objective PEM* which relies on other modalities of player response such as physiological signals; *gameplay-based PEM* in which player actions within a game environment is considered for constructing the models. Different PEM approaches can be combined to construct *hybrid PEM*. In Section 2.4.1, we surveyed the work done in modeling players’ affect using subjective, objective and hybrid measures. In the following, we explore the research attempts to construct gameplay-based models of player experience.

2.7.6.1.1 gameplay-based PEM: Several attempts can be found in the literature on constructing models based on gameplay data collected from the interaction between the player and the game. Studies by Yannakakis and Hallam [2006] shown that artificial neural networks (ANN) and fuzzy neural networks can be used to construct an estimator of player satisfaction based on metrics of player experience better than a human-designed one. Further work in this direction (Pedersen et al. [2010]) showed that players’ reported experience states can be estimated up to a good degree using neuroevolutionary models constructed from the in-game interaction.

Constructing estimators of player experience based on gameplay data can be classified into three main categories: *model-based*, *model-free* and *hybrid* between the two (Yannakakis and Togelius [2011]).

- **Model-based:** Model-based approaches typically rely on theoretical framework of behavior and/or cognitive analysis such as the computational models of emotion presented in Section 2.2. Some examples of the work done using this approaches include the work of Togelius et al. [2006] and Hastings et al. [2009] that have been previously discussed. Other work includes the study done by Olesen et al. [2008] who adjust the challenge of a real-time strategy game to meet player skills by collecting information about player behavior and tactical strategies. In (van Lankveld et al. [2008]), the

progress of the player and the amount of damage she has sustained is employed as an indicator of skill in a turn based, side-scrolling arcade game adopting the incongruity measure from psychological literature. Most of the work presented previously on dynamic difficulty adjustment and game balancing can be seen as model-based, gameplay-based PEM.

- **Model-free:** Model-free PEM, on the other hand, tries to predict player actions and intentions or identifies patterns of player behavior. Machine learning techniques are usually employed to achieve this goal. Some work in this area include the study conducted by [McGlinchey \[2003\]](#) who used offline learning through Self Organizing Map (SOM) neural network to learn the characteristics of individual players while playing a game of Pong. [Wong et al. \[2009\]](#) also used (SOMs) to cluster the playing style in a shooter game. In ([Drachen et al. \[2009\]](#)) a SOM is also used to classify playing behavior data in Tomb Raider: Underworld ([Crystal Dynamics et al. \[2008\]](#)). Neural Gas and Bayesian imitation learning algorithms are used in [Thureau et al. \[2004, 2005\]](#) to learn and imitate human players' movement in the computer game Quake II ([id Software et al. \[1997\]](#)). The study conducted by [Martínez et al. \[2010\]](#), showed that player types as identified by SOM can be used to better model player experience in a 3D prey/predator game. [Ortega et al. \[2012\]](#) investigated and compared several methods (including neuroevolution, dynamic scripting, rule-based evolutionary computation and grammatically evolved behavior trees) to train models to imitate human playing style in Super Mario Bros ([Nintendo Creative Department \[1985\]](#)) based on data collected about player actions. [Weber and Mateas \[2009\]](#) crowd-sourced gameplay data from thousands of replays and employed data mining and machine learning techniques for opponent modeling based on the data collected in a real-time strategy game.

2.7.6.2 Personalized Content Generation

The player experience models constructed, adopting any of the approach presented previously, can then be used as an evaluation function for game content that can be optimized to maximize player experience. According to the ED-

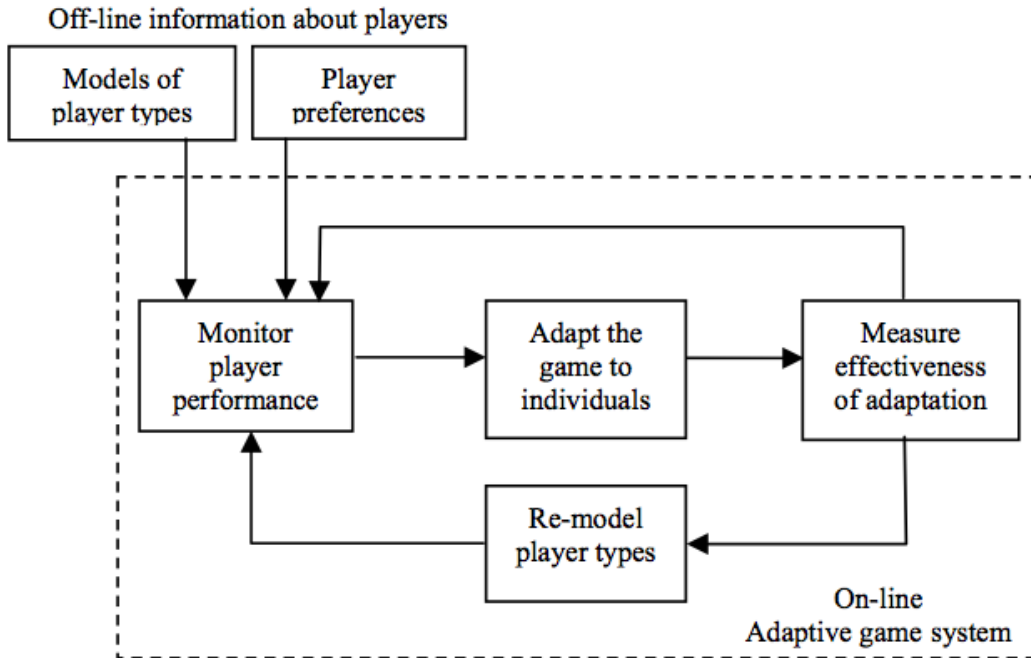


Figure 2.14: The adaptive game system diagram (Charles and Black [2004]).

PCG framework (Yannakakis and Togelius [2011]), a set of requirements should be fulfilled for a system to successfully realize the affective loop: “The game should be tailored to individual players’ affective response patterns; the game adaptation should be fast, yet not necessarily noticeable; and the affect-based interaction should be rich in terms of game context, adjustable game elements and player input”. A similar adaptation framework was proposed by Charles and Black [2004] where the authors suggested a continuous measure of adaptation efficiency through the use of constructed models of player experience as presented in Figure 2.14. The authors raise two important issues of game adaptation “we need to know when to adapt the game to a player... we should monitor if our adaptation has been effective or appropriate”. Similar requirements for adaptive systems were also proposed by Gilleade and Dix [2004] “we need to consider the motivation of the users: why they want to play, their experience and skills: how able are they to play, and detection: how to identify when change is necessary.”

The literature on personalized and player-adaptive PCG is so far scarce. Few attempts emerged recently focusing on adapting game content using computa-

tional models of player emotion built from the interaction between the player and the game. Typical examples of such work include the studies conducted by Hastings et al. [2009]; Pedersen et al. [2010]; Togelius et al. [2007]; Yannakakis et al. [2010] and Liapis et al. [2012] presented previously. Other studies include the work done by Avery et al. [2011] on evolving towers and creeps for a tower defense game where a new path is generated to reflect different playing styles. The authors adopted the same content personalization principle proposed by Hastings et al. [2009] by generating towers through an interactive evolutionary process: the towers the players choose to place are used to generate new personalized collection of towers to choose from. Tijs et al. [2008] indicated that emotionally adaptive games adjust their mechanics to optimize players' gaming experience. In their experiment, game speed is manipulated by analyzing the relationship between game mechanics and players' emotional state measured through self-reports and physiological signals. Saari et al. [2009] introduced adaptation in the context of a first person shooter (FPS) game, where player affect is measured via psychophysiological measures. Kazmi and Palmer [2010] proposed an adaptation mechanism based on players action in a FPS game where different in-game mechanisms are triggered in response to particular player actions.

The awareness of the importance of automatic personalized adaptation has also recently increased in commercial games with games such as *Pro Evolution Soccer* (Konami [2007]) featuring adaptive AI system, called Teamvision, that learns and adapts the team tactics' according to an individual's style of play. In *Lego Star Wars II* (Traveller's Tales and Robosoft Technologies [2006]), adaptive difficulty is added as an optional feature which affects the amount of LEGO studs (the part LEGO bricks that connects them together) the player loses upon death according to how well she plays.

2.7.6.3 Assessing the Quality of the Personalized Content

Once personalized game content is generated, an evaluation mechanism should be implemented to assess content quality and to validate the efficiency of the adaptation approach. *Direct*, *simulation-based* and *interactive-based* functions can be used for quality assessment (Yannakakis and Togelius [2011]). Direct evaluation

functions map features extracted from the content generated to a content quality value. This approach is the easiest to implement and the fastest to evaluate.

Simulation-based evaluation functions utilize AI agents that play through the content generated and estimate its quality. Statistics are usually calculated about the agents behavior and playing style and used to score game content. The type of the evaluation task determines the area of proficiency of the AI agent; if content is evaluated on the basis of playability, that is the existence of a path from the start to the end in a maze or a level in a 2D platform game, then AI agent should be designed that are excel in reaching the end of the game. On the other hand, if content is optimized to maximize particular player experience, then an AI agent that imitates human behavior is usually adopted. An example study that implement human-like agent for assessing content quality is done by [Togelius et al. \[2007\]](#) where neural network-based controllers are trained to drive like human players in a car racing game and then used to evaluate the generated tracks. Each track generated is given a fitness value according to playing behavior statistics calculated while the AI controller is playing.

Interactive-based functions rank content based on the in-game interaction. [Hastings et al. \[2009\]](#) implemented this approach by evaluating the quality of the personalized weapons evolved implicitly based on how often and how long the player chooses to use these weapons.

2.8 Summary

In this chapter the state-of-the-art of affect recognition while interacting with a digital interface, computational aesthetics and procedural content generation in computer games were presented. Subsequently, the current research on constructing computational models of emotions based on different emotional theories applied in the two main domains: human-computer interaction and computer games was discussed and its potential was revealed. In addition, the literature review on procedural content generation with its two main overlapping subareas; search-based and experience-driven procedural content generation is extensively reviewed.

The work presented in this chapter provides the background on which the stud-

ies and experiments presented in this thesis are built. Players' affective/cognitive states are captured following the self-reporting scheme where players are asked to report their emotional states in a forced choice response measurement following the pairwise preferences protocol proposed by Yannakakis and Hallam [2006]. Visual cues features, in terms of head movements, as well as player gameplay characteristics are employed to build hybrid computational models of player experience. Based on these models, the aesthetic considerations of level design are analyzed. An online procedural content generation methods are implemented to close the affective loop and realize the EDPCG framework. The approach followed to search for appropriate content for a given player can be categorized as online adaptive stochastic constructive approach. Simulation-based evaluation function is utilized to assess content quality while content is represented using parameterized and indirect grammar-based representations. Content generation is achieved using search based and evolutionary based approaches.

3

Tools

This chapter presents the algorithms and techniques adopted to successfully tackle the player experience modeling, content representation and generation, and game adaptation problems. Each section presents techniques used to tackle a specific subproblem.

3.1 General AI Techniques

The following sections present an introduction to a number of AI techniques used in the later sections.

3.1.1 Evolutionary Computation

Evolutionary computation (EC) ([Eiben and Smith \[2008\]](#)) is a subfield of artificial intelligence inspired by biological mechanisms of evolution. Problem solving in evolutionary computation is achieved by exploring the space of potential solutions and applying biologically inspired operations to improve these solutions. The fundamental metaphor of EC relates the power of evolution in nature to a particular style of problem solving, that of trail-and-error (also known as generate-and-test) ([Eiben and Smith \[2008\]](#)).

Evolutionary approaches have proven to be very efficient when exploring large search spaces with many local optima and when handling problems with non-exact, yet measurable, objective functions.

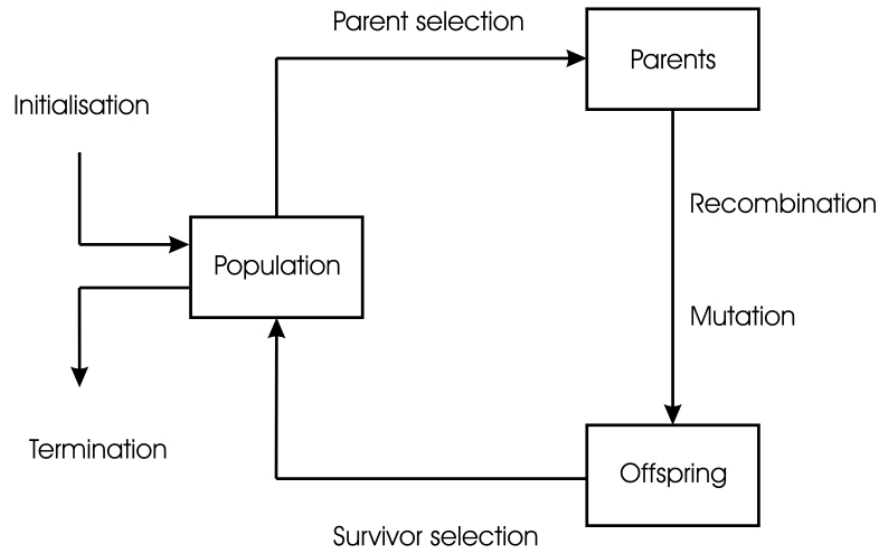


Figure 3.1: The general workflow of an Evolutionary Algorithm (Eiben and Smith [2008]).

There are several approaches of evolutionary computation: *evolutionary programming* (Fogel et al. [1966]), *genetic algorithm* (Holland [1975]), *evolution strategies* (Rechenberg [1973]) and *genetic programming* (Koza [1990, 1994]; Koza et al. [2005]). Of particular interest to the work presented in this dissertation are genetic algorithms, genetic programming and grammatical evolution. The algorithms involved in all of these fields are termed *evolutionary algorithms* and each field is considered a subarea belonging to the corresponding algorithm variant (Eiben and Smith [2008]).

Evolutionary Algorithms (EAs) work by evolving a randomly initialized population toward better solutions. Each individual in the population (called a chromosome, a genotype or a genome) is evaluated and given a fitness value. Genetic operators are then applied on a number of selected individual based on their fitness. The resulted offsprings are then used as the next generation and the whole process is repeated. The algorithm terminates when either a maximum number of generations is reached or when the population reaches a satisfactory fitness level. A diagram of an EA is presented in Figure 3.1 while a general scheme of an EA is given in Figure 3.2.

There are two fundamental forces that form the basis of evolutionary systems

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

Figure 3.2: The general scheme of an Evolutionary Algorithm (Eiben and Smith [2008]).

(Eiben and Smith [2008]):

- Variation operators (recombination and mutation) which facilitate creating the necessary diversity and thereby assist novelty, while
- selection acts as a force improving quality.

The various dialects of EC follow the same scheme presented in Figure 3.2 and differ only in technical details such as the representation of a candidate solution. While the candidates in genetic algorithms are represented by strings over a finite alphabet, real-valued vectors, finite state machines and trees are used in evolution strategies, evolutionary programming and genetic programming, respectively (Eiben and Smith [2008]).

3.1.1.0.1 Components of Evolutionary Algorithms: A particular variation of EA can be defined by a specification given to a number of EA components (Eiben and Smith [2008]). The most important components are:

- **Representation:** The first step in defining an EA is to identify the mapping between the problem and the problem solving space. In EAs, the term *phenotype* is usually used to refer to the representation of a solution in the

original problem context. The encoding of an individual solution within EA is called a *genotype*. Using this terminology, representation can be defined as specifying the mapping from the phenotypes onto a set of genotypes.

- **Fitness function:** In EAs, the evaluation (fitness) defines what improvement means and based on that it assigns a quality measure to genotypes.
- **Population:** The population is the pool of possible solutions. Defining a population usually means specifying the number of individuals in it. This number is mostly constant throughout the evolution process. During the course of evolution, the population adapts and improves while individuals remain static.
- **Parent selection:** Individuals in EAs are distinguished based on their quality using parent selection. This mechanism allows better individuals to become parents of the next generation and, thereby, ensures quality improvement. Probabilistic selection is usually implemented for parent selection with better individuals having a higher chance of becoming parents. However, in order to escape local optimum and prevent greedy behavior, a small selection chance is usually given to individuals with low quality. There are several methods for parent selection. Some example techniques include roulette wheel selection, Boltzmann selection, tournament selection and rank selection.
- **Variation operators:** New individuals are created from old ones by means of variation operators. A slight modification of an individual can be obtained using a *mutation* operator. In Mutation, the new individual, also known as an *offspring*, is the result of a stochastic operation on a parent in which the values of one or more chosen genes are changed (see Figure 3.4). On the other hand, *crossover* is a binary stochastic variation operator that merges information from two parents producing one or two offsprings. Crossover is based on the principle that by mating two individuals with different but desirable features, offsprings which combines both of these features can be produced. There are several variations of the crossover operator such as one-point crossover, two-point crossover and

3.1. GENERAL AI TECHNIQUES

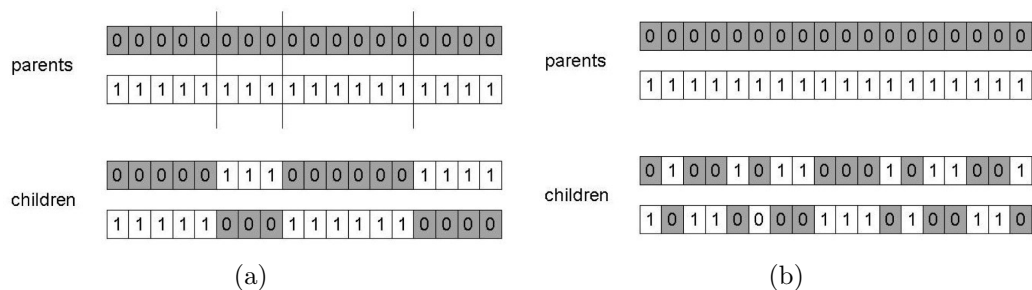


Figure 3.3: An example of n-point crossover (a) and a uniform crossover (b).

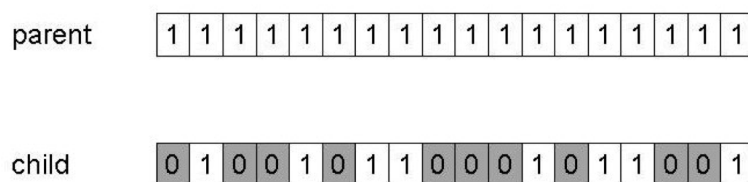


Figure 3.4: An example of a mutation operator.

uniform crossover. Figure 3.3 presents an example of a n-point and uniform crossover.

- Survivor selection:** After creating the offsprings of the selected parents, survivor selection, also known as replacement, is called to choose the individuals to be survived to the next generation based, mostly, on their fitness value. This is usually applied by selecting the top segment of a ranked population.

3.1.1.1 Genetic Algorithms

A Genetic Algorithm (GA) is an evolutionary approach that uses search heuristics to generate solutions to solve optimization problems. GAs became very popular through the work of Holland (Holland [1975]) who demonstrated the successful application of GAs in an adaptive system.

A simple GA system is characterized by a binary representation, a fitness proportionate selection, a mutation of low probability and a genetically inspired recombination (Eiben and Smith [2008]). A GA typically implements a similar

scheme to the one presented in Figure 3.2.

3.1.1.2 Genetic Programming

Genetic Programming (GP) is a form of GA with two fundamental differences: while a GA evolves fixed-length strings, a GP adopts a variable-length representation of individuals where each individual is considered a computer program. Evolution in GP is applied directly to the solution unlike other GA methods which use an indirect representation of the potential solution.

Individuals in GP are usually represented with a tree-based structure and the same genetic operands used in GA are applied to breed high-quality solutions. Mutation in GP is often not used at all while crossover is usually the only variation operator (Eiben and Smith [2008]). Individuals' evaluation is performed based on their abilities to perform a given task.

GP has a wide range of applications, and recently, GP has demonstrated its potential in developing solutions that are competitive to those generated by humans (Bentley [1999, 2000]; Koza et al. [2003, 2005]; O'Neill and Brabazon [2001]; O'Neill et al. [2010]; Takagi [2001])

3.1.1.3 Grammatical Evolution

Grammatical Evolution (GE) is an evolutionary algorithm based on GP (O'Neill and Ryan [2001]). The main difference between GE and GP is the genome representation; while a tree-based structure is used in GP, GE relies on a linear genome representation.

The population of the evolutionary algorithm is initialized randomly consisting of variable-length integer vectors; the syntax of possible solution is specified through a context-free grammar. GE uses the grammar to guide the construction of the phenotype output from these strings. As in GA, the genotype-to-phenotype mapping and the fitness calculation are repeated for every individual in the population. The genetic operators are then applied to the individuals with the best fitness resulting in a new population. The evolution process continues and the population evolves towards better solutions.

GE employs grammars written in Backus Naur Form (BNF). Each chromo-

3.1. GENERAL AI TECHNIQUES

some is made up of codons. Each codon in the string is used to select a production rule from the BNF grammar. A complete program is generated by selecting production rules from the grammar until all non-terminal rules are mapped. The resulted string is evaluated according to a fitness function to give a score to the genome.

Because of the use of a grammar, GE is capable of generating anything that can be described as a set of rules such as mathematical formulas (Tsoulos and Lagaris [2006]), programming code, game levels (Shaker et al. [2012]) and physical and architectural design (Byrne et al. [2011]; O'Neill et al. [2009]).

GE has been used intensively for automatic design (Byrne et al. [2011]; Hemberg and O'Reilly [2004]; Hornby and Pollack [2001]; O'Neill and Brabazon [2008]; O'Neill et al. [2009]), a domain where it has been shown to have a number of strengths over more traditional optimization methods.

3.1.1.3.1 Backus Naur Form Backus Naur Form (BNF) is set of production rules used to express a grammar. A BNF grammar $G = \{N, T, P, S\}$ consists of terminals, T , non-terminals, N , production rules, P and a start symbol, S . Non-terminals can be expanded into one or more terminals and non-terminals by applying the production rules. An example BNF is given in Figure 3.5.

Each individual in GE consists of a variable number of integer strings that are used in the genotype-to-phenotype mapping. These integer strings are typically evolved with a GA and then used to choose production rules when a non-terminal with more than one outcome is encountered. To better understand the genotype-to-phenotype mapping, we will give a brief example.

Consider the grammar in Figure 3.5 and the individual genotype integer string (4, 5, 8, 11). We begin the processing of an individual from the start symbol $\langle exp \rangle$. In this case there are three possible productions, to decide which production to choose, we use the first value in the input genome and apply the mapping function $4\%3 = 1$, where 3 is the number of possible productions, i.e. the second production is chosen, and $\langle exp \rangle$ is replaced with $(\langle exp \rangle \langle op \rangle \langle exp \rangle)$. The mapping continues by using the next integer with the first unmapped symbol in the mapping string, the mapping string then becomes $(\langle var \rangle \langle op \rangle \langle exp \rangle)$ through the formula $5\%3 = 2$. At this step, $\langle var \rangle$

-
- (1) $\langle \text{exp} \rangle ::= \langle \text{exp} \rangle \langle \text{op} \rangle \langle \text{exp} \rangle$
 | $(\langle \text{exp} \rangle \langle \text{op} \rangle \langle \text{exp} \rangle)$
 | $\langle \text{var} \rangle$
 (2) $\langle \text{op} \rangle ::= + \mid - \mid * \mid /$
 (3) $\langle \text{var} \rangle ::= X$

Figure 3.5: Illustrative grammar for generating mathematical expressions.

has only one possible outcome and there is no choice to be made, hence, X is inserted without reading any number from the genome. The expression becomes $(X \langle op \rangle \langle exp \rangle)$. Continuing to read the codon values from the example, individual's genome $\langle op \rangle$ is mapped to $+$ and $\langle exp \rangle$ is mapped to X through the two formulas, $8\%4 = 0$ and $11\%3 = 2$, respectively. This results in the expansion $(X + X)$.

During the mapping process, it is possible for individuals to run out of genes, in this case GE either declares the individual as invalid by assigning it with a penalty fitness value or it wraps around and reuses the genes. To guarantee a deterministic genotype-to-phenotype mapping, GE insures that whenever an individual is mapped the same output is generated.

3.1.2 Artificial Neural Networks

Artificial Neural Networks (ANNs) are mathematical models inspired by the biological neural networks. ANNs constitute a well-known approximation method and in most cases ANNs are considered adaptive systems in the sense that they change their underlying structure according to the information flow through them.

Several papers can be found in the literature on neural networks and their applications. The reader may refer to (Haykin [1999]; Hertz et al. [1991]; Rumelhart et al. [1985]) for a thorough description of ANNs. In the following discussion we will limit our description to the particular classes of neural network that are used throughout this thesis, namely Single-layer Perceptron (SLP) and Multi-layer Perceptron (MLP).

In this dissertation, ANNs are mainly used as approximation methods to find the unknown relationship between a set of input features and the outputs. The use of ANNs for function approximation has been investigated intensively in the

3.1. GENERAL AI TECHNIQUES

literature. They have been proven to be able to approximate any function given enough number of processing elements (Cybenko [1989]; Hornik et al. [1989]; Kolmogorov [1963]).

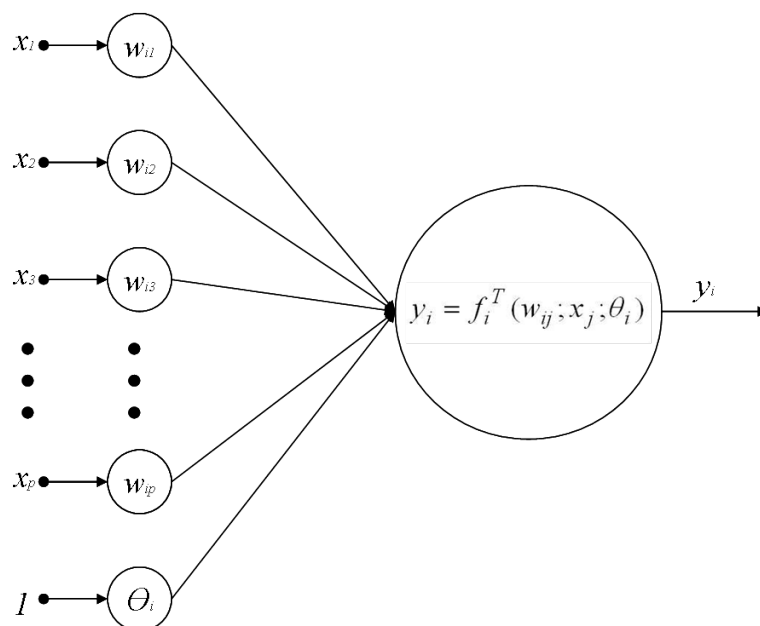


Figure 3.6: The artificial neuron.

3.1.2.1 Single-layer Perceptron

Single-layer perceptron (Rosenblatt [1958]) is the simplest kind of neural network consisting of a single layer of output neurons. Such networks can be trained more quickly and are easier to analyze than more complex MLP networks. However, SLPs are only capable of learning linearly separable problems and hence they have only been used as linear approximators. The structure of such networks can be seen in Figure 3.6. The network consists of an input layer connected to a node in the next layer, which is also the output layer. This node takes the weighted sum of all its inputs and apply a transfer function of the from

$$y_i = f_i^T(w_i; x_i; \theta_i) \quad (3.1)$$

where θ_i is a classification threshold and

$$f_i^T = \begin{cases} 1 & \text{if } w_i x_i + \theta_i > 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.2)$$

3.1.2.2 Multi-layer Perceptron

The MLP (Rumelhart et al. [1986]) is the most common type of neural network used in applied research. It can be seen as a generalization of SLP with more computational power and an ability to solve non-linearly separable problems. MLPs consist of two or more layers of nodes (neurons) employing non-linear activation functions. It has been proven that a neural network with such a structure is capable of approximating any numerical function, as long as enough number of hidden neurons is provided (Bishop [1995]).

The basic building blocks of MLPs are the *neuron* and the *connection*. Neurons are structured in a graph in which each node is connected to other nodes via directional connections. The ANN graph is usually structured in layers; the first layer is where the input is placed and hence it is called the input layer, this layer is followed by one or more hidden layers and finally an output layer. Each node in the hidden layer employs a weighted sum function on all of its inputs. A *transfer (activation) function* is then applied on the resulting summed value. More formally, each node i employs a function f_i^T of the form

$$y_i = f_i^T \sum_{j=1}^n (w_{ij} x_j + \theta_i) \quad (3.3)$$

where y_i is the i^{th} neuron's output; n is the number of inputs to the neuron i ; x_j is the j^{th} input to the neuron; w_{ij} is the connection weight between neuron i and neuron j ; and θ_i is the threshold of the neuron. The choice of the transfer function highly affects the functionality of the neural network and is crucial for its universal approximation capacity.

Learning in neural networks is achieved by adjusting the connection weights of the ANN so that it performs certain tasks. Several approaches have been followed to train MLPs; the three key approaches are via supervised learning,

unsupervised learning and reinforcement learning, each correspond to solving a particular learning task.

3.1.3 Evolving Artificial Neural Networks

Evolutionary neural networks (Stanley and Miikkulainen [2002]; Yao [1993]) is the research field that emerged from combining evolutionary algorithms with neural networks. The name refers to the use of evolutionary algorithms to set the weights for neural networks and it sometimes refers to evolving the network topology. This method has been widely used to automatically generate efficient ANNs. The practice of evolving neural networks is called *neuroevolution*. Because neuroevolution relies on a fitness function to measure the performance of the network, the method can be used to train networks to approximate a function in a reinforcement learning manner.

For the experiments presented in this dissertation, we consider the simplest form of neuroevolution, namely an MLP whose weights are evolved. Assuming we are evolving MLPs with a number of c weight, the key steps of the method are as follows:

1. A population of N individuals is initialized, each individual is a vector of c real numbers which represent the connection weights of the ANN.
2. Each individual (ANN) is evaluated according to a fitness function that assesses the performance of the ANN.
3. Parents are selected (via a selection strategy), offspring are generated via genetic operators (any variant of mutation and crossover) and members of the population are replaced (following any replacement strategy).
4. If the termination condition is not satisfied, go to step 2.

3.1.4 Preference Learning

Preference learning has received increasing attention in the machine learning literature in recent years (Fürnkranz and Hüllermeier [2010]). The ranking problem

has been categorized into three main types, namely label ranking, instance ranking and object ranking (Förnkrantz and Hüllermeier [2010]). We focus on object ranking in this dissertation. Within object ranking, the goal is to learn a ranking function $f(\cdot)$ that produces a ranking of a given subset of objects given their pairwise preferences. More formally, given a set of instances Z and a finite set of pairwise preferences $x_i \succ x_j; (x_i, x_j) \in Z \times Z$, find a ranking function $f(\cdot)$ that returns the ranking of this set Z . Here, $x_i \succ x_j$ means that the instance x_i is preferred to x_j .

Various methods have been presented in the literature for the task of object ranking. Methods based on large-margin classifiers (Bahamonde et al. [2004]; Fiechter and Rogers [2000]; Herbrich et al. [1998]), Gaussian processes (Chu and Ghahramani [2005]; Gervasio et al. [2005]), and neuroevolution (Yannakakis et al. [2009]) have been investigated to learn the ranking function.

Neuroevolutionary preference learning is used in this dissertation due to its powerful approximation capability and its efficiency in modeling player experience in similar problems to the one at hand (Martinez et al. [2009]; Pedersen et al. [2009, 2010]; Yannakakis et al. [2009]). In this thesis we chose to focus on implementing one technique since we are more interested in realizing a complete implementation of the affective loop in games than exploring other possible methods.

In the following sections, we describe the features selection method used to extract relevant features from each object instance and the neuroevolutionary preference learning methodology proposed to learn the ranking function.

3.1.5 Feature Selection

Feature selection is an important step within a machine learning process since one would desire the model to be dependent on as few features as possible, both to make the analysis easier and to make the result more generalizable and easier to incorporate into future applications. Moreover, not all features are relevant for predicting preferences, and applying feature selection improves learning quality by eliminating inappropriate features. In this thesis, Sequential Forward Selection (SFS) (Whitney [1971]) is used to select the relevant subset of features for

3.1. GENERAL AI TECHNIQUES

predicting preferences (Yannakakis et al. [2008]).

SFS is a bottom-up search procedure where one feature is added at a time to the current feature set. The feature to be added is selected from the subset of the remaining features such that the new feature set generates the maximum value of the performance function over all candidate features for addition. The process is terminated when an added feature yields lower validation performance compared to the one obtained without it.

As the description of SFS points out, the method is a variant of hill-climbing, hence, it does not necessarily result in the optimal set of features since it does not explore all possible combinations.

The method has been successfully applied to wide variety of feature selection problems where it yields minimum feature subsets with high performance (Pedersen et al. [2009]; Yannakakis et al. [2009, 2008]). A comparative study have been conducted to test the performance of the method against other feature selection methods in a similar setup to the problem presented in this work (Pedersen et al. [2009]). The comparison results showed that SFS outperforms the other methods (such as n best individual feature selection, sequential floating forward selection and perceptron feature selection) both in finding the minimum subset of relevant features and in the classification accuracy. The benefits of SFS on picking feature set with minimal effort (compared to e.g. exhaustive search or global metaheuristic search) which yield highly performing preference models are the key motivations for using SFS in this thesis.

3.1.6 Neuroevolutionary Preference Learning

Neuroevolutionary preference learning (Yannakakis et al. [2009]) is used to learn the ranking function that approximates the relationship between selected features and reported preferences. Since there are no prescribed target outputs for the learning problem, standard backpropagation loss functions (e.g. mean squared error) are inapplicable. Learning is achieved via artificial evolution in this thesis.

In neuroevolutionary preference learning, a genetic algorithm (GA) evolves an artificial neural network (ANN) so that its output matches the pairwise preferences in the data set. The input of the ANN is a set of features that have been

extracted from the data set. The GA implemented uses a fitness function that measures the difference between the reported preferences and the relative magnitude of the model output. A sigmoid-based fitness function has been adopted as its shape has been optimized for maximum model performance. The ANN topology is fixed, and the GA chromosome is a vector of ANN connection weights.

The evolutionary procedure (Yannakakis et al. [2006]) used can be described as follows: a population of N networks is initialized randomly. Initial real values for their connection weights are picked randomly from a uniform distribution. Then, at each generation:

Step 1 Each member (neural network) of the population gets two d -tuples (where d is the number of input features) one for A and one for B , where A and B are two instances associated with a pairwise preference, and returns two output values, namely $y_{j,A}$ and $y_{j,B}$ for each pair j of the dataset. When the $y_{j,A}$, $y_{j,B}$ values are consistent with the actual reported preference of subject j then we state that there is an ‘agreement’ between the output and the reported preferences. In the opposite case, we state that there is a ‘disagreement’.

Step 2 Each member i of the population is evaluated via the fitness function f_i (which defines the loss/error function of the problem):

$$f_i = \sum_{j=1}^N \begin{cases} g(d_j, p_k) & \text{if agreement} \\ g(d_j, p_l) & \text{if disagreement} \end{cases} \quad (3.4)$$

where $d_j = y_{j,A} - y_{j,B}$ and $g(d_j, p) = 1/(1 + e^{-pd_j})$ is the sigmoid function, p_k and p_l are assigned empirically.

Step 3 A roulette-wheel selection scheme is used as the selection method.

Step 4 The Montana and Davis crossover (Montana and Davis [1989]) is applied to the selected parents for generating two offspring. Gaussian mutation occurs in each gene (connection weight) of each offspring’s genome.

The algorithm is terminated when the total number of generations is reached or when the population reaches a satisfactory fitness level.

3.2 Sequence Mining

Sequence data mining is a technique used to identify frequent subsequences of patterns in a dataset of samples. Sequential pattern mining was first introduced by Agrawal and Srikant [1995] and has since then been applied to a wide range of applications including user modeling, e-commerce and business intelligence.

In the work presented in this dissertation, we are interested in finding frequently occurring patterns within a dataset. One particular type of dataset is of particular interest for this work. This type is what is usually called *temporal and time-series database*, in which each temporal data item is associated with a corresponding time attribute.

Several approaches can be found in the literature for mining sequential patterns (Mannila and Toivonen [1996]; Oates et al. [1997]; Srikant and Agrawal [1996]; Zaki [2001]). Two algorithms for frequent itemset mining have been implemented to find frequent sequence patterns within a dataset of sequences, namely Apriori and Generalized Sequential Patterns (GSP). The Apriori algorithm (Agrawal and Srikant [1994]) is used to mine single-dimensional sequences occurring in a relatively small dataset. Mining sequences across multiple time series of data within a large dataset is achieved via the GSP algorithm (Srikant and Agrawal [1996]). GSP allows us to mine variable-length sequences and introduces a few more tunable parameters, making it well suited for mining sequences of multiple dimensions.

In the following sections we provide a list of definitions relevant for frequent subsequence mining and give a brief description of the two algorithms.

3.2.1 Definitions

A *data-sequence* is a sample of a sequential dataset where each sequence consists of a number of events, each one associated with a time stamp. The events are ordered by increasing time.

A *sequence pattern*, l_i , is a non-empty set of simultaneous events denoted by $\langle e_0e_1e_2\dots e_n \rangle$ where e_i is an event. A data-sequence supports a sequence pattern if and only if it contains all the events present in the pattern in the same order but not necessarily consecutive.

A minimum support, min_{sup} , is the minimum number of times a pattern l_i has to occur in the data-sequences to be considered frequent. If the number of occurrences of l_i in the data-sequences exceeds the min_{sup} , we call l_i a *frequent pattern* and in this case, the fraction of data-sequences that support l_i is referred to as *support count*, sup_{count} .

3.2.2 Apriori Algorithm

Most of the algorithms for sequence mining are based on the Apriori property proposed by Agrawal and Srikant [1994]. Apriori uses a bottom-up approach to find frequent subsets of patterns by extending one item at a time employing breadth-first search and hash tree structure to efficiently count candidate sets. The main apriori property states that “All nonempty subsets of a frequent itemset must also be frequent”.

Suppose we want to find frequent patterns of size k in a database, the Apriori algorithm is decomposed into five main phases:

1. Find frequent items of size 1, L_1 .
2. Candidate generation, L_{n+1} : given a dataset of frequent items of size n , generate the candidates of size $n + 1$ by joining L with itself.
3. Use the apriori property to prune the infrequent items of L_{n+1} .
4. Scan the database and count the support for each candidate in L_{n+1} .
5. Remove those candidates from L_{n+1} such that their min_{sup} is smaller than a user-defined threshold.
6. If $n + 1 < k$, let $n = n + 1$ and go to step 2.

The main disadvantage of the algorithms that depend on the apriori property is the computation cost. The algorithm has to keep track of the support count for each subsequence for testing the apriori property in each iteration. This leads to a large search space. Without introducing methods for narrowing the search space, the applicability of the algorithm on large databases becomes infeasible.

Several variations of the apriori algorithms have been proposed with solutions to reduce the computational cost (Garofalakis et al. [1999]; Han et al. [2000]; Lin and Lee [2002]; Srikant and Agrawal [1996]; Zaki [2001]). In the following section, we describe two of the well-known variations that are used throughout the work presented in this dissertation.

3.2.3 Sequential Pattern Discovery: SPADE

Sequential PAttern Discovery using Equivalence classes (SPADE) is an algorithm for frequent sequence mining (Zaki [2001]). The dataset in SPADE is represented using a vertical id-list $\langle itemset - id, timestamp \rangle$ format where each sequence is associated with a list of items in which it occurs. Sequence mining is then performed by growing a supsequence one item at a time using Apriori candidate generation. Candidate generation is done in the main memory and uses a lattice-theoretic approach to decompose the search space search while simple two join operations are typically used to generate candidate sequences (Zhao and Bhowmick [2003]). Frequent sequences can be efficiently found using intersections on id-lists. By utilizing id-lists, the method reduces the number of databases scans, and therefore also reduces execution time.

The main steps in SPADE are the following:

1. Compute the frequencies of 1-sequences, L_1 in an Apriori-like way.
2. Count the sequences of length 2, L_2 . This is done by counting the number of sequences for each pair of items using a bidimensional matrix.
3. Construct a lattice from all the 2-item sequence found.
4. Search the lattice for generating the sequences of length n , L_n . This is done by joining the $n - 1$ sequences using their id-lists. The size of the id-lists is the number of sequences in which an item appears. If this number is greater than min_{sup} , the sequence is a frequent.

The id-list for each item is produced using breadth-first or depth-first search. The algorithm terminates when no more frequent subsequences can be found. SPADE usually makes only three database scans to extract the frequent patterns.

The main drawback of the lattice theory is that the lattice can grow very big and cannot fit in the main memory when dealing with large datasets. SPADE solves this problem by partition the lattice into disjoint subsets called “equivalence classes” that can be loaded into the main memory and searched separately (Mabroukeh and Ezeife [2010]). In this dissertation, we use a simplified version of SPADE to find frequent patterns when the dataset size is relatively small and therefore there is no need to implement the equivalence classes.

3.2.4 Generalized Sequential Patterns

The GSP algorithm (Srikant and Agrawal [1996]) solves the sequence mining problem based on an apriori algorithm (Agrawal and Srikant [1994]) with a number of generalizations.

Using GSP, we can discover patterns with a predefined minimum support, define time constraints within which adjacent events can be considered elements of the same pattern, and specify a time window for events from different modalities to be considered as synchronous events.

GSP generalizes the basic definition of frequent sequential pattern by introducing two relaxation schemes:

- Sliding window: This generalization allows the items of a pattern to be contained in the union of the items belonging to different time-series. According to this relaxation, a sequence $s = \langle s_i s_j \rangle$ — where s_i and s_j can be contained in different time-series — is allowed to be counted as a support for a subsequence c as long as the time difference between s_i and s_j is less than the user specified window-size, max_{win} .
- Time constraints: This relaxation specifies the time gap between consecutive events from one or two different time-series. Given a user-defined gap, max_{gap} , a data-sequence supports the pattern of two consecutive events $\langle s_i s_{i+1} \rangle$ if and only if s_i and s_{i+1} occur in the sequence of the specified order and with a time difference lower than the specified max_{gap} .

Based on these generalizations, GSP main distinctions from other standard apriori algorithms are in terms of candidate generation and frequent pattern

3.2. SEQUENCE MINING

generation. Similar to the basic apriori generation procedure, candidates of length k are generated based on frequent subsequences of length $k - 1$. GSP follows the same basic apriori principle; if a sequential pattern is frequent then its *contiguous* subsequences are also frequent. Given a sequence $s = \langle s_1, s_2, \dots, s_n \rangle$, and a subsequence c , c is a contiguous subsequence of s if any of the following conditions hold:

1. c is obtained from s by dropping an item from the first (s_1) or last (s_n) element.
2. c is derived from s by dropping an item from an element s_i with two or more items.
3. There exist a sequence q such that q is a contiguous subsequence of s and c is a contiguous subsequence of q .

The algorithm generates the candidates by joining two sequence sets of length $k - 1$ that have the same contiguous subsequences. Elements in the resulting set that have a contiguous subsequence whose support count is less than a user-defined threshold are eliminated.

The GSP algorithm is used in this dissertation for mining sequences since it allows more generalized frequent patterns to be found by exploring different max_{gap} . By using max_{win} , we can discover simultaneous events from two different modalities.

The max_{win} defines the threshold under which events from two different modalities can be considered as simultaneous events. The max_{gap} parameter is used to set up the time gap between two events to be considered as belonging to the same pattern. This parameter has a great impact on the number of frequent patterns that can be extracted. Correctly tuning this parameter has a large impact on the informativeness of the resulted patterns, especially when mining multimodal sequences (Martinez and Yannakakis [2011]).

3.3 Summary

This chapter provides a high-level description of the literature on several AI techniques. In some of the particular experiments these algorithms are used in their naive form, whereas in other experiments slightly modified versions are used. Those algorithms are described in detail in the relevant chapters, referring to the descriptions given in this chapter.

More specifically, Grammatical Evolution is used in Chapter 5 for generating game content for our testbed game. Genetic Algorithms are used for evolving ANN weights in Chapter 7 for constructing accurate models of player experience and preference learning is used as a training method. The inputs to the models are different types of features, some extracted using sequence mining techniques (see Chapter 8). Feature selection methods are employed to choose the minimum subset of relevant features for modeling player experience (see Chapter 7).

4

The Testbed Game

4.1 Platform Games

Platform games (or *platformers*), originated in the early 1980s, are an extremely popular genre of video games in which the player avatar has to jump on platforms and overcome obstacles. Jumping in platform games constitutes an essential gameplay mechanic, and it is sometimes coupled with other jump-related mechanics such as swinging or bouncing. Platformers may also include other forms of movement such as floating, walking, or running as well as other types of mechanics such as shooting and transporting items. The ultimate aim of a platform game is to reach the end of each level. Sometimes other auxiliary goals are presented such as collecting as many items as possible or killing enemies.

The early examples of platformers exhibit a static view of a 2D play field due to technical limitations. *Space Panic* (Universal [1980]) (Figure 4.1.(a)), which inspired many other games, is cited as the first platform game featuring a single screen with a static background where the player has to climb ladders, dig holes and kill enemies.

Donkey Kong (Nintendo EAD et al. [1981]) (Figure 4.1.(b)) was the first platform game offering the core mechanic of platformers; jumping. The game was released by Nintendo in 1981, and introduced *Mario* for the first time, the main character of the very popular game *Mario Bros.* The game was composed of different levels of vertical challenge presented in one screen each.

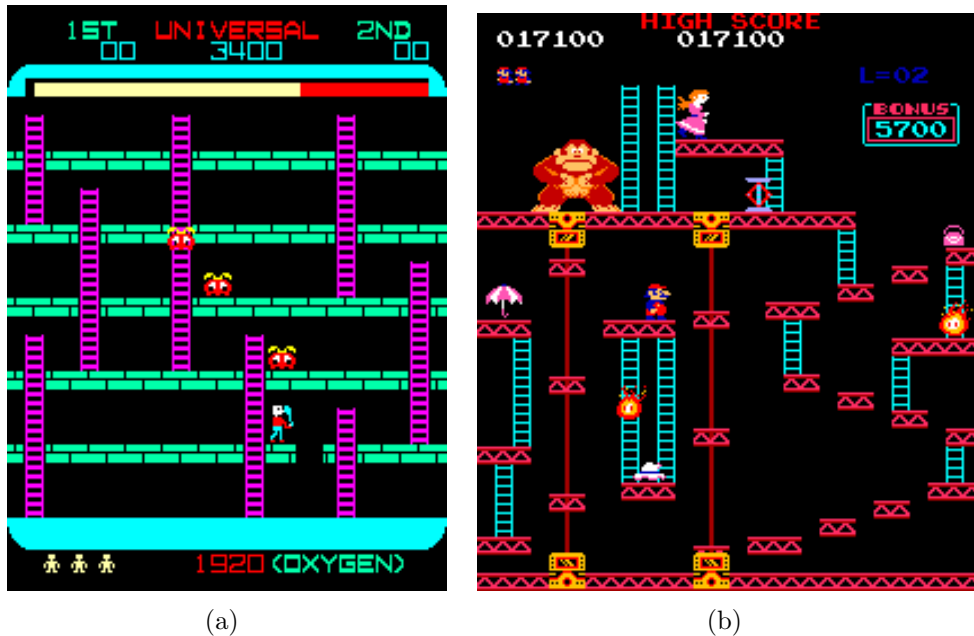


Figure 4.1: Two examples of 2D platform games. Sub-figures (a) presents a level in Space Panic, the first platform game (adopted from wikipedia.org). Sub-figures (b) presents a level from the platform game Donkey Kong (adopted from arcade-museum.com).

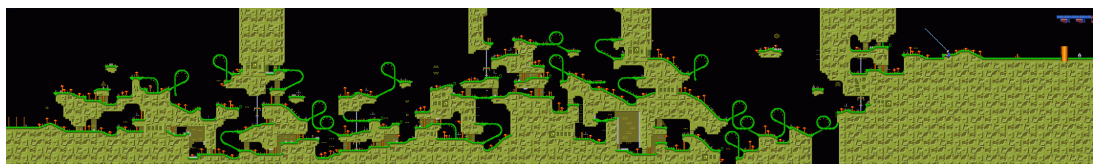
Platform games witnessed a wide spread and increasing popularity with the production of games featuring scrolling levels that span several screens in 1982. The same year witnessed an exponential increase in the platform game market when games started being available for home consoles. Several games using scrolling graphics have been developed starting from *Jump Bug* (Hoei Corporation [1981]), a platform shooter game, to *Pitfall* (Activision [1982]), the second best-selling game made for the Atari 2600 after *Pac-Man* (Namco [1980]), and *Impossible Mission* (Epyx [1984]) which was the inspiration for such games as *Prince of Persia* (Brøderbund et al. [1989]).

In 1985, the development of new systems gave platform games a boost in the market. Nintendo released its platform game *Super Mario Bros* (Nintendo Creative Department [1985]) which sold over 40 million copies. With the continuous advancement of console systems, new generations of platform games with more advanced features and options were produced. Some very successful plat-

4.1. PLATFORM GAMES



(a) A level from the platform game *Commander Keen* (adopted from commander-keen.com).



(b) A level from the platform game *Sonic & Knuckles* (adopted from soniccenter.org).

Figure 4.2: Two examples of 2D platform games.

form games were released, including *Commander Keen* (id Software and David A. Palmer Productions [1990]), *Earthworm Jim* (Shiny Entertainment and Playmates Interactive Entertainment [1994]), *Sonic 3 & Knuckles* (Sonic Team and Sega Technical Institute [1994]), *Super Mario World 2: Yoshi's Island* (Nintendo EAD [1995]) and *Rayman* (Ubisoft et al. [1995]). Example levels from two games are presented in Figure 4.2.

In 1994, a new sub-genre of platform game emerged with games featuring levels in 2.5D. In these games, the environment is rendered in 3D but retained 2D gameplay. Few games for this era have been released such as *Klonoa 2: Lunatea's Veil* (Namco [2001]) and *Viewtiful Joe* (Clover Studio [2003]).

Platform games entered its golden age with the invention of 3D platformers with new words in 3D and three dimensions gameplay. *Alpha Waves* (Infogrames [1990]) is considered the earliest example of 3D platformers released in 1990. Several successful games have been released including *Jumping Flash!* (Exact Co. Ltd. [1995]), *Bug!* (Realtime Associates [1995]), *Fade to Black* (Delphine Software Internationals [1995]), *Super Mario 64* (Nintendo EAD [1996]), *Rayman 2: The*

Great Escape (Ubisoft Montpellier et al. [1999]), *Jak and Daxter: The Precursor Legacy* (Naughty Dog [2001]), *Spyro the Dragon* series (Insomniac Games [1998]), *Super Mario Galaxy* (Nintendo EAD [2007]), *Mirror's Edge* (EA Digital Illusions CE [2008]) and *Sonic Generations* (Sonic Team and Dimps [2011]).

The work presented in this dissertation focuses on 2D platformer games domain, and in particular, Super Mario Bros. Platformers are well suited for the research carried on in this dissertation due to their popularity, rich level design and relatively simple game mechanics.

4.2 Super Mario Bros

Super Mario Bros (Nintendo Creative Department [1985]) is a 2D platform game developed by Nintendo and released in 1985. The main character in the game is *Mario* who is controlled by the player. The game also offers two-player cooperative play by allowing another player to control Mario's brother *Luigi*. The game held the record of the best selling video game until 2003 by having sold more than 40.23 million copies worldwide, and was the first in a series of many other successful Nintendo's games.

The game takes place in the Mushroom Kingdom, the objective is to survive through a number of 2D levels (there were eight different worlds with four levels each designed for the original Super Mario Bros) to save *Princess Peach Toadstool* who has been captured by *Bowser*, the king of *Koopas*.

The game features the appearance of enemies, obstacles and rewarding items. The gameplay in Super Mario Bros consists in moving Mario through the levels. The game provides relatively simple game mechanics; Mario has only four movement mechanics: walking, running, jumping, and ducking. The main goal of each level is to reach the flag pole at the end of the level, which means traversing it from left to right jumping over gaps (gaps are the pits in the game in which Mario may fall and die) and avoiding enemies. Enemies include *Goombas*, *Koopas*, *Bullet Bills*, *Koopa Paratroopas* and *Piranha Plants*, each with different strength, attack and defeat methods. Goombas, which look like walking mushrooms, are the physically weakest enemies and can be killed by stomping over them. Koopas, which look like turtles, come next in strength after goombas, they cover in their

4.2. SUPER MARIO BROS

shell when jumped on, the shell can then be kicked by Mario to kill other enemies. Koopa paratroopas are koopas with wings which they lose when they are attacked in the air, they exhibit a variety of flying patterns depending on their color; red koopas fly up and down or side to side in a set path while green ones bounce in the player's direction. Piranha plants are found in pipes, they attack by biting Mario and they can be defeated by shooting them or kicking a shell at them. Finally, bullet bills are shot out of bill blasters (cannons) and they can be defeated by stomping on them. Table 4.1 presents the different types of elements in SMB and their graphical representation.

Several rewarding items are scattered around the levels. *Coins* are placed around the level for Mario to collect; *Blocks* are marked with a question mark, and they reveal coins, *Super Mushroom* or *Fire Flower* when hit from below by Mario; *Brick Blocks* on the other hand, may hide an item or they can be empty, and they are breakable when hit by Mario from below if he is in *Super Mario* mode. Super mushrooms make Mario grow to *Super Mario* while fire flowers turn Mario into *Fire Mario*. While in super mode, Mario can survive one hit by enemies as well as being able to smash bricks. The fire mode allows Mario to shoot fireballs at enemies giving him the ability to attack remotely.

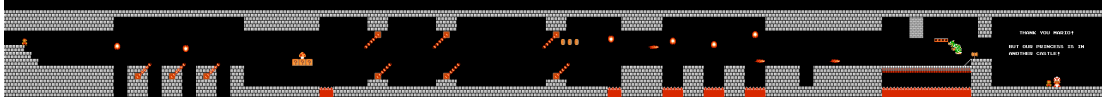
At the beginning of each game, Mario is given a certain number of lives and he may gain additional lives by picking up special items or performing specific actions. Mario loses a life when he is touched by an enemy in small mode, falls down a gap or runs out of time.

The ultimate aim of each level is to reach the flag pole at the end of it. Auxiliary objectives include collecting as many as possible of the coins, clearing the level as fast as possible, and killing as many enemies as possible. A comprehensive list of the different types of items and enemies is presented in Table 4.1. The graphical icons presented in the table are adopted from a public domain clone of the game, named *Infinite Mario Bros*, which is described in the next section. Two complete levels from the original Super Mario Bros game are presented in Figure 4.3.

Only some of the main rules and gameplay mechanics of Super Mario Bros are described above. In fact, no textual description can fully convey the full gameplay of a particular game. The game is still very popular and playable more than two

















(a)



(b)

Figure 4.3: Two example levels of the original Super Mario Bros game (adopted from ian-albert.com).

Table 4.1: Some elements that appear in Super Mario Bros and their graphical representation.

Element	Graphical Rep.	Element	Graphical Rep.
Mario		Bullet bill	
Super Mario		Bill blaster	
Fire Mario		Piranha plant	
Goomba		Coin	
Koopa		Block	
Koopatrols		Brick	
Super Mushroom		Fire flower	

decades after its release.

We chose Super Mario Bros game as the testbed for our research. Given our aims, this game provides unique properties. Firstly, the game is a commercial-standard platform game which adds to the generality and the scalability of our approach; secondly, the game is very popular and well known having a large audience of players which ease any online data collection experiments, thirdly; the game laid the groundwork and closely resembled many other 2D platform games and finally, an open source clone of the game is available.

4.3. INFINITE MARIO BROS



Figure 4.4: Snapshot from Infinite Mario Bros, showing Mario standing on horizontally placed blocks surrounded by different types of enemies.

4.3 Infinite Mario Bros

Markus Persson has published a public domain clone of Nintendo’s classic platform game Super Mario Bros. The clone, named *Infinite Mario Bros.* (IMB), features the art assets and general game mechanics of Super Mario Bros. but differs in level construction. Infinite Mario Bros is playable on the web, where Java source code is also available¹. While implementing most features of Super Mario Bros, the standout feature of Infinite Mario Bros is the automatic generation of levels. Every time a new game is started, levels are randomly generated by traversing a fixed width and adding features according to certain heuristics as specified by placement parameters. Table 4.1 presents the main elements of IMB.

In the experiments conducted in this dissertation we employed modified versions of IMB. In two modified versions we concentrate on a number of selected game level parameters that affect game experience, while in the third version, we exploited the content space with minimum restriction imposed on the content generator. Figure 4.4 presents a snapshot from a level generated by a modified version of IMB, Mario appears standing on horizontally placed blocks surrounded by koopas and goombas.

Infinite Mario Bros, along with its modifications, have been used relatively extensively as a testbed for research and as a testing environment for various AI techniques in e.g. reinforcement learning of proficient game-playing strategies (Togelius et al. [2009]), imitation of human playing styles (J. Togelius and Shake

¹<http://www.mojang.com/notch/mario/>

[2011]; Ortega et al. [2012]), player experience modelling (Pedersen et al. [2010]; Shaker et al. [2010]), procedural content generation (Mawhorter and Mateas [2010]; Shaker et al. [2010]; Sorenson et al. [2011]; Sorenson and Pasquier [2010]), and as a testing environment for various AI techniques (Bojarski and Congdon [2010]; Perez et al. [2011]). The game is also being used as a benchmark for the Mario AI Championship¹ (Karakovskiy and Togelius [2012]; Shaker et al. [2011]) and as a believability assessment tool (J. Togelius and Shake [2011]).

Infinite Mario Bros has been chosen because of the popularity of Super Mario Bros, the high similarity between the two, the availability of an open source clone of the game which makes development and data collection easier and because of the 2D design and game mechanics it provides which are similar to other games from the same genre.

The original Infinite Mario Bros source code has been heavily modified for the purpose of conducting the experiments and analysis presented in this dissertation. In Chapter 5, we describe the original IMB generator and the different modified versions derived from it.

¹<http://www.marioai.org/>

5

Content Generators

The design of game content is a creative activity that consumes a lot of resources in terms of time and money. Consequently, there has been increasing interest recently in automatic generation of game content with or without human designer interaction. Using these computational techniques, it could be not only possible to reduce development cost, but also to generate endless variations of content that provides a unique experience with every replay. The content could even be adapted to the preferences and skills of individual players.

In this chapter, we explore three different techniques for content generation for our testbed game, Infinite Mario Bros. The content space generated by each of these generators is analyzed and compared to the other generators' expressive space in Chapter 6. These generators are further used in the later chapters for the purpose of player experience modeling (Chapter 7), data collection (Chapter 8), and adapting game content (Chapter 10).

We first present the *Notch Generator* (Section 5.2) which is a heuristics-based generator that comes originally with IMB. Section 5.3 discusses the *Parametrized Generator* which is a heavily modified parametrized version of Notch generator, and finally, in Section 5.4, the *Grammatical Evolution Generator* is described for which grammatical evolution is employed to explore the content space generated according to a design grammar that specifies the rules of content generation.

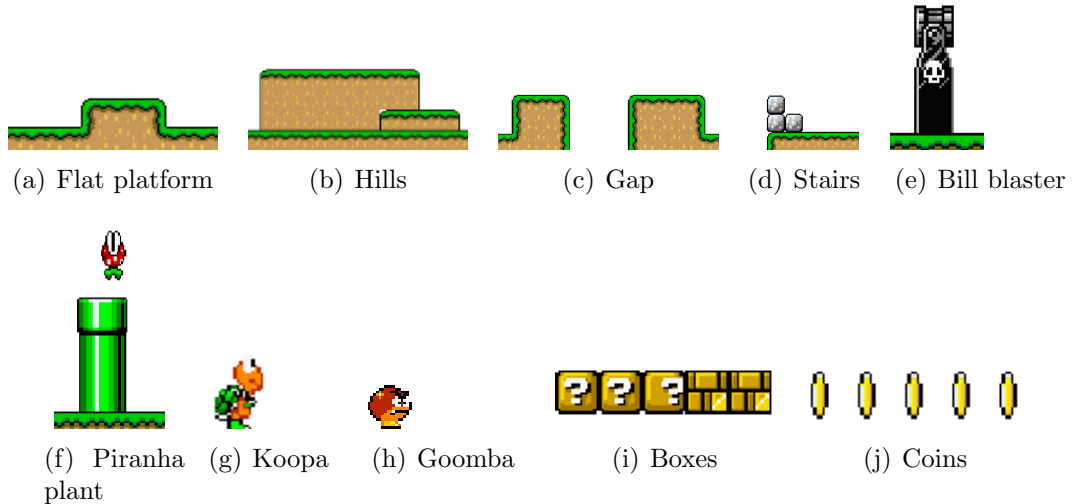


Figure 5.1: The geometric representation of the different chunks used for constructing Infinite Mario Bros levels.

5.1 Level Representation

The internal representation of the levels in Infinite Mario Bros is a two-dimensional array of objects, such as brick blocks, coins and enemies. In “small” state, Mario is one block wide and one block high.

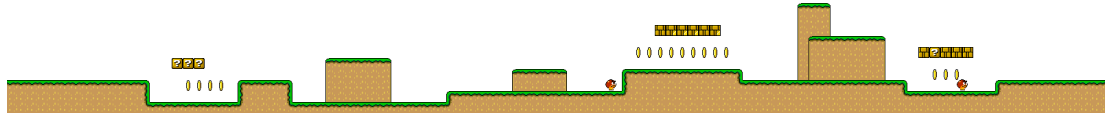
The levels are generated by placing a number of chunks in the two-dimensional level map. The list of chunks that has been considered in this work includes platforms, gaps, stairs, piranha plants, bill blasters, boxes (blocks and brick blocks), coins, goombas and koopas. Each of these chunks has a distinguishable geometry and properties. Figure 5.1 presents the different chunks that collectively constitute a level. We assume that the level initially contains a flat platform that spans the whole x-axis. This assumption ensures that all chunks in the resulted design will be connected and explains the need of defining gaps as one of the chunks.

Different approaches can be followed to place the chunks into the 2D map. In the following sections we will describe the three different approaches utilized for the work presented in this dissertation. The three content generators exhibit a wide range of variation in the methodology followed for level constructions, hence, a number of disparate experiments have been conducted to quantitatively evaluate the content generated by each generator.

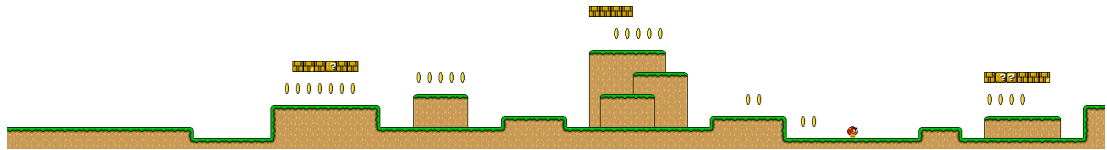
5.2 Notch Level Generator

The Notch level generator is the one written by Markus Persson and comes originally with the game. It constructs levels by incrementally placing different chunks according to certain heuristics. The level generation can be parameterized by defining the level of difficulty which affects the number of generated platforms, hills, gaps, bill blasters, piranha plants, enemies (koopas and goombas) and the type of enemies. The generator constructs endless variation of levels by using a different seed whenever a new game is started.

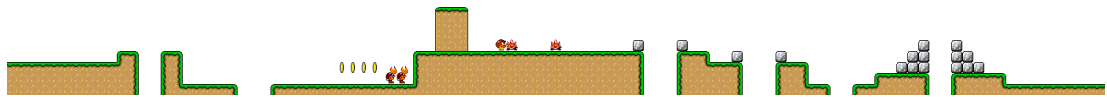
The level is constructed by traversing the level map from left to right and adding chunks. While the level is constructed and whenever a new chunk is to be placed, a random number is generated and compared with predefined heuristics defined for each chunk according to the level of difficulty specified. As a result of this comparison, the type of the chunk to be created is determined. The chunk's specific parameters, such as the width of a gap or the height of a platform, are controlled by another set of random numbers initialized by the seed. Reward items such as coins and blocks are added to each generated chunk according to the level seed and difficulty. Figure 5.2 presents two levels constructed by the Notch generator with varying seeds and difficulties. Levels 5.2.(a) and 5.2. (b) are generated with different seeds and a difficulty = 0, while level 5.2. (c) is constructed using a difficulty value that equals three. As can be seen, different seeds result in levels with varying structure even when the same level of difficulty is used. Levels with low difficulty are characterized by a low number of enemies—most of them being goombas, the weakest form of enemies—, a relatively high number of rewarding items, and no gaps. Levels constructed with high difficulty values, on the other hand, exhibit the presence of more challenging elements such as larger number of gaps of different width and more enemies with higher strength. These levels also present a small amount of rewarding items making them more challenging and harder to win.



(a) A level from IMB with a level difficulty = 0.



(b) Another level from IMB with a different seed and a level difficulty = 0.



(c) A level from IMB with a level difficulty = 3

Figure 5.2: Example levels from Infinite Mario Bros generated by the original Notch generator with different seeds and difficulty values.

5.3 Parameterized Level Generator

The parameterized generator is a heavily modified version of the original Notch level generator. The level generator of the game has been modified to generate content according to a number of predefined content features.

Throughout this dissertation, the term *Controllable features* is frequently used to refer to a set of chosen content features as these features are used to control the generation of content and are varied to make sure several variants of the game are played and compared for the purpose of player experience modeling.

Content features have been chosen in a way that permits meaningful exploration of the search space and possibilities of finding interestingly new design parameter configurations. The selection of particular controllable features was done with the intent to cover the features that have the most impact on player experience and which are common to most, if not all, platform games to elaborate on the generality of the proposed methodology.

5.3. PARAMETERIZED LEVEL GENERATOR

For the work presented in this dissertation, we survey two versions of the parameterized generator. In the first version, the generator is used to explore the content space specified by four content features, while in the second version, a larger set of six content features has been employed to control the content space.

5.3.1 Content Features

Two versions of the parameterized generator have been employed for the work presented in this dissertation.

5.3.1.1 Basic Parameterized Generator

In the first version, the Notch generator has been modified to generate content according to four content parameters:

- The number of gaps in the level, G .
- The average width of gaps, \bar{G}_w .
- The gaps entropy which measures the number of gaps appearing in a number of E equally-spaced segments of the level. The entropy of gap-placements G_e is calculated and normalized into $[0,1]$ via the equation:

$$G_e = \left[-\frac{1}{\log E} \sum_{i=1}^E \frac{g_i}{G} \log \left(\frac{g_i}{G} \right) \right] \quad (5.1)$$

where g_i is the number of gaps placed in segment i . If the gaps are placed in all E segments uniformly then $g_i = 1$ for all E segments and H_g will be 1; if all gaps are placed in one level segment, then $g_i/G = 1$ and H_g is zero.

- Direction switch S_w . This parameter defines the percentage of the level played in the left direction. If this parameter is set to zero, this means no direction switch and the player needs to traverse the level from left to right in order to reach the end of the level, as in the original Super Mario Bros. If $S_w > 0$ the level direction will be mirrored at certain switch points, forcing the player to move in the opposite direction until reaching the end of the level or the next switch.

These features were also presented in the Notch generator; however, they have been control by heuristics. Our modifications include constraining their generation, and hence, imposing various changes in the level generation mechanism.

Level construction has been done by assigning low and high values for each of these four content features and exploring all possible combinations of the resulting states.

5.3.1.2 Advanced Parameterized Generator

Another version of the parameterized generated has been investigated for the experiments conducted in this dissertation. In this version, six content features have been explored permitting more control and variation in the level design. The first two parameters are the same one used in the previous version of the parameterized generator due to the important role they play in platform game level design and because they have a great impact on player experience. The full list of parameters includes:

- The number of gaps in the level, G ;
- The average width of gaps, \bar{G}_w .
- The number of enemies, E . This parameter controls the number of Goombas and Koopas scattered around the level, affecting the level difficulty.
- Enemies placement, E_p . The way enemies are placed around the level is determined by three probabilities which sum to one.
 - Around horizontal boxes (blocks and/or bricks), P_x : Enemies are placed on or under a set of horizontal boxes (a number of blocks placed horizontally without connection to the ground).
 - Around gaps, P_g : Enemies are placed within a close distance to the edge of a gap.
 - Random placement, P_r : Enemies are placed on a flat space on the ground.

5.3. PARAMETERIZED LEVEL GENERATOR

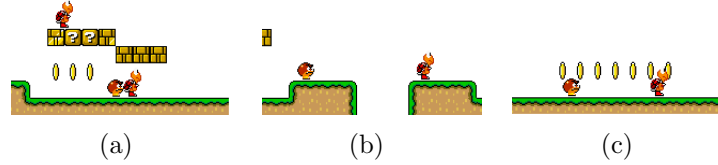


Figure 5.3: Enemies placement using different probabilities: high probability is given to placement around horizontal boxes, P_b (a), around gaps, P_g (b), and to random placement, P_r (c).

Figure 5.3 illustrates positioned enemies by giving different values for P_b , P_g and P_r . Figure 5.3.(a) shows enemies placed by setting P_b to 80%. Figure 5.3.(b) illustrates the result of setting P_g to 80%, and Figure 5.3.(c) is the result of $P_r = 80\%$.

In practice, this feature takes one of the three values 0, 1 or 2 specifying which of these probabilities should be assigned the highest value. For example, if E_p is 1, then the highest probability is given to P_g and a level is generated with $P_z = 10\%$, $P_g = 80\%$ and $P_r = 10\%$.

- The number of powerups, N_w : This includes super mushrooms and fire flowers that are placed hidden in boxes for Mario to collect and upgrade his state from small to super or from super to fire.
- The number of boxes, B . We define one variable to specify the number of the two different types of boxes that exist in Super Mario; *blocks* and *bricks*. Blocks usually contain hidden elements such as coins or powerups. Bricks may hide a coin, a powerup or simply be empty.

Two states (low and high) are set for each of the controllable parameters above except for enemies placement which has been assigned three different states allowing more control over the difficulty and diversity of the generated levels. An example level generated by one possible combination of the controllable features is presented in Fig. 5.4. Note that these two states are not the only possible values that can be assigned for these features, and more variations of content can be generated by increasing the number of states. However, for the work presented



Figure 5.4: An example level generated by the parametrized generator using six content features.

in this dissertation, we are interested in generating the most distinct levels and therefore only two states have been employed.

The generation of levels with specified values for all parameters is guaranteed by the generator; while generating the levels, and whenever an item is to be added, these parameters are checked and the item is placed accordingly.

5.4 Grammatical Evolutionary Generator

Grammatical evolution has been adopted to generate content for SMB because of the advantages it provides in the design domain over more traditional optimization methods (O’Neill et al. [2010]): it maintains a simple way of describing the structure of the levels; it enables an open-ended structure where the design and model size are not known a priori; it enables the design of aesthetically pleasing levels by exploring a wide space of possibilities since the exploratory process is not constrained or biased by imagination or known solutions; it allows an easy incorporation of domain knowledge through its underlying grammatical representation permitting level designers to maintain greater control of the output and makes it possible to easily generalize to different types of games and finally, GE, to the best of author’s knowledge, has not been exploited for game content creation previously.

As mentioned in the description of GE in Section 3.1.1.3, GE specifies the syntax of possible solutions through a context-free grammar. In the following section, we present the design grammar used by GE to specify the structure of IMB levels.

5.4.1 Design Grammar

The process in which levels are constructed is represented in the input grammar that GE uses in the construction of a solution (in this case a level design). Several methods for specifying the design grammar have been discussed during the development process, however, the grammar specified by GE is of context-free nature. This means that it is not possible to construct the levels by gradually scanning the level maps without introducing repeated patterns. To accommodate for this requirement, and to keep the grammar as simple as possible to ease the designer's interaction with the system; the solution proposed, inspired by the work of (Morel et al. [2005]), is to add a chunk to the 2D level array regardless of the positioning of the other chunks. With this solution, however, arise a number of conflicts in level design that should be resolved. Section 5.4.2 discusses this issue and the proposed solution in details.

A design grammar has been specified that takes into account the different chunks that collectively constitute a level. The list of chunks that has been considered includes: platforms, gaps, tubes, bill blasters, boxes, coins, and enemies (refer to Figure 5.1). In order to allow more variations in the design, we distinguish between two types of platforms; *obstruct-platforms* which block the path and enforce the player to perform a jump action (Figure 5.1.(a)), and *hills* that give the player the option to either pass through or jump over them (Figure 5.1.(b)). Platforms and hills of different types have been considered such as a blank platform/hill, a platform/hill with a bill blaster, and a platform/hill with a piranha plant.

The early version of the grammar that has been designed is presented in Figure 5.5. A level is constructed by placing a number of chunks each assigned with two or more properties, the x and y parameters specify the coordinates of the chunk starting point position in the 2D level array and are limited to the ranges [5,95] and [3,5], respectively. These ranges are limited by the dimension of the level map. The first and last five blocks in the x dimension are reserved for the starting platform and the ending gate, while the y values have been constrained in a way that insures playability (the existence of a path from the start to the end position) by placing all items in areas reachable by *Mario* by performing jumps.

```

<chunks> ::= <chunk> |<chunk> <chunks>
<chunk> ::= gap(<x>,<y>,<wg>)
           | platform(<x>,<y>,<w>)
           | hill(<x>,<y>,<w>)
           | blaster_hill(<x>,<y>,<h>)
           | tube_hill(<x>,<y>,<h>)
           | coin(<x>,<y>,<wc>)
           | blaster(<x>,<y>,<h>)
           | tube(<x>,<y>,<h>)
           | boxes(<x>,<y>,<wb>)
           | enemy(<x>,<y>,<we>)
<x> ::= [5..95]
<y> ::= [3..5]
<wg> ::= [2..5]
<w> ::= [3..15]
<wc> ::= [2..6]
<wb> ::= [2..7]
<we> ::= [1..7]
<h> ::= [3..4]

```

Figure 5.5: The first version of the grammar employed to specify the design of IMB levels.

The w_g parameter specifies the width of gaps that insures the ability to reach the other edge, w stands for the width of a platform or a hill, w_b defines the number of boxes, w_e determines the number of enemies, w_c defines the number of coins, and h indicates the height of a tube of the piranha plant or the height of a bill blaster. This height is also constrained to the range [3,4] assuring the possibility of jumping over tubes and bill blasters.

An example phenotype that results from the grammar in Figure 5.5 can be *hill(10, 4, 4)platform(74, 3, 4)tube(62, 4, 3)*. Because of the context-free nature of the grammar, the chunks generated in the phenotype are not necessarily ordered in x or y dimensions. Note that, as discussed in section 3.1.1.3, the genotype to phenotype mapping is a deterministic process guided by the grammar specified. This also includes the assignment of the parameters for each chunk since the parameters are also specified as part of the grammar.

An example of a resulting level is depicted in Figure 5.6. Visualizing samples of the outputs and thoroughly examining the design grammar reveal limitations

5.4. GRAMMATICAL EVOLUTIONARY GENERATOR

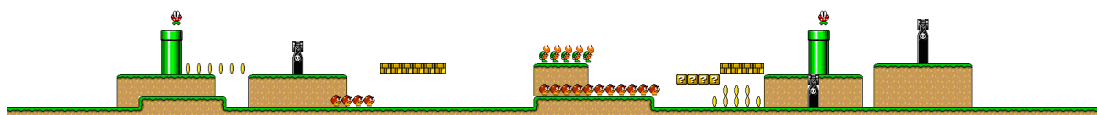


Figure 5.6: An example level generated by the first version of the grammar. The design illustrates a number of limitations in the grammar such as the placement of enemies and the generation of boxes.

in the design exposed by the grammar. The definition of gaps, piranha plants, and bill blasters in the grammar only specifies the width of the gaps and the height of the tubes and blasters. As a result of this definition, each one of these elements will be generated with equal-width platforms surrounding it (Figure 5.6). According to game designers, the width of the platform before and after these elements plays an important role in the gameplay experience and the level of difficulty. For example, the width of platform before a gap affects the difficulty of the game since speeding up is sometimes required to launch a wide jump to overcome a wide gap. Therefore, this parameter has been split into two parameters specifying the width of the platform before w_{before} and after w_{after} each of these chunks. Introducing these two parameters also accommodates for more control and variation in the design.

The other limitations concern the generation of boxes and enemies. The definition proposed in the grammar results in generation of groups of only bricks or only blocks. In IMB boxes are usually presented as groups of bricks and blocks collectively. For this to be allowed a refinement in the grammar has been made so that whenever a box is to be placed, a decision has to be made regarding the type of the generated box, the box can be either an empty brick, a brick containing a coin, a block with a powerup or a block with a coin. In order to maintain the grouping feature, boxes are generated in combination of a minimum of two boxes and a maximum of six, as listed by the superscripts depicted in Figure 5.7. The same argument holds for enemies and a similar solution has been adopted to allow for different types of enemies (koopas and goombas) to be introduced.

The final limitation relates to the placement of enemies. In the first version of the grammar, enemies are spawned in groups. To make sure enemies are always

placed on a platform, whenever an enemy is generated, an associated platform is created on which the enemy is placed. This forces groups of enemies of the same type to be always placed on a separate platform (Figure 5.6). To support more variabilities, the grammar has been improved to allow enemies of different types to be placed on any generated platform (around gaps, tubes, etc.). This has been accomplished by (1) constructing the physical structure of the level, (2) calculating the possible positions on which an enemy can be placed (this includes all positions where a platform has been generated) and (3) placing each generated enemy in one of the possible positions. The place on which the enemy is placed is determined by generating a random number by the grammar and mapping it on the list of possible positions after constructing the main structure of the level. This place has been defined as a parameter in the grammar to maintain the deterministic genotype to phenotype mapping.

The final version of the grammar that has been created to overcome all the limitations discussed can be seen in Figure 5.7.

5.4.2 Conflict Resolution

There are a number of conflicts inherent within the design grammar. According to the design approach, each chunk generated can be assigned any x and y values from the ranges $[5,95]$ and $[3,5]$, respectively, depending on the genotype without any restrictions. This means that it is very likely that there will be an overlap between the coordinates of the generated chunks. For example: *hill*(65, 4, 5) *hill*(25, 4, 4) *blaster_hill*(67, 4, 4, 4, 3) *coin*(22, 4, 6) *platform*(61, 4, 4) is a phenotype that has been generated by the grammar and contains a number of conflicts: e.g., *hill*(65, 4, 5) and *blaster_hill*(67, 4, 4, 4, 3) have been assigned the same y value, and overlapping x values; another conflict occurs between *hill*(25, 4, 4) and *coin*(22, 4, 6); as the two chunks also overlap on their $x - axes$.

To resolve these conflicts, a priority value has been manually defined and assigned to each of the chunks. Hills with bill blasters or piranha plants are given the highest priority followed by blank hills, platforms with enemies (bill blasters or piranha plants) come next then blank platforms and finally come coins and blocks with the lowest priority. The chunks in the generated phenotype are then

5.4. GRAMMATICAL EVOLUTIONARY GENERATOR

```
<level> ::= <chunks> <enemy>
<chunks> ::= <chunk> |<chunk> <chunks>
<chunk> ::= gap(<x>,<y>, <wg>,<wbefore>,<wafter>)
  | platform(<x>,<y>,<w>)
  | hill(<x>,<y>,<w>)
  | blaster_hill(<x>,<y>,<h>,<wbefore>,<wafter>)
  | tube_hill(<x>,<y>,<h>,<wbefore>,<wafter>)
  | coin(<x>,<y>,<wc>)
  | blaster(<x>,<y>,<h>,<wbefore>,<wafter>)
  | tube(<x>,<y>,<h>,<wbefore>,<wafter>)
  | <boxes>

<boxes> ::= <box_type> (<x>,<y>)2 | ...
  | <box_type> (<x>,<y>)6

<box_type> ::= blockcoin | blockpowerup
  | brickcoin | brickempty

<enemy> ::= (koopas | goombas)(<pos>)2 | ...
  | (koopas | goombas)(<pos>)10
<x> ::= [5..95]
<y> ::= [3..5]
<wg> ::= [2..5]
<wbefore> ::= [2..5]
<wafter> ::= [2..5]
<w> ::= [2..6]
<wc> ::= [2..6]
<h> ::= [3..4]
<pos> ::= [0..100000]
```

Figure 5.7: The final version of the grammar employed to specify the design of the level. The superscripts (2, 6 and 10) are shortcuts specifying the number of repetition.

arranged in descending order taking into account the chunk priority, coordinates and type. The resulted ordered phenotype is then scanned from left to right. While scanning, a conflict check is performed between the current chunk and the rest of the chunks coming later in the ordered list. Whenever two chunks overlap, the one with the higher priority value is maintained and the other is removed. Nevertheless, to allow more diversity, some of the chunks are allowed to overlap such as hills of different height (Figure 5.1. (b)), and coins or boxes with hills (hills here refer to all types of hills; blaster-hills, tube-hills and flat hills). Without this refinement, most levels would look rather flat and uninteresting. In the above example, priority ordering of the chunks gives the ordered phenotype: *blaster_hill*(67, 4, 4, 4, 3) *hill*(25, 4, 4) *hill*(65, 4, 5) *platform*(61, 4, 4) *coin*(22, 4, 6). After eliminating the conflicting chunks, the resulting phenotype becomes: *blaster_hill*(67, 4, 4, 4, 3) *hill*(25, 4, 4) *platform*(61, 4, 4) *coin*(22, 4, 6). Note that both *blaster_hill*(67, 4, 4, 4, 3) and *platform*(61, 4, 4) have been maintained although they have overlapping coordinates. This is because the two chunks are of different types that are allowed to overlap. The same argument holds for *hill*(25, 4, 4) and *coin*(22, 4, 6).

5.5 Summary

This chapter explores different approaches for generating content for the 2D platform game Infinite Mario Bros. The aim is to place the level building blocks, called chunks, into a two-dimensional level map. Three content generators have been investigated each with different parameters and generation methods. In the first generator, called Notch Generator (Section 5.2), a level is constructed according to a set of heuristics, these heuristics are parameterized by a random seed and a desired level of difficulty. The presence of randomness allows this generator to generate endless variation of content with each reply. The Parameterized Generator is a modified version of Notch generator. This generator constructs levels by specifying the values of a set of parameters, called content features. These features control the number and/or placement of certain content elements. Parameterizing the content space allows us to analyze the impact of certain content features on player experience and generate content according to user-defined

5.5. SUMMARY

preferences.

The third generator is an evolutionary-based generator in which grammatical evolution is utilized to evolve the design of the levels. The structure of the levels has been defined in a grammar that GE uses to construct levels. This chapter presented the process followed to implement the GE level generator.

Samples of each generator's outputs are presented. However, the samples don't reflect the capability of each generator and the space of content the generator covers. Therefore, it is very important to evaluate the content generated by each of these techniques and compare it against content generated by other techniques. Because of the large amount of content that can be generated, it is not humanly feasible to judge the results, and automatic evaluation becomes a necessity. In the next chapter, a framework for comparing content generated by different generators is presented. A number of expressivity measures are defined to test the generator's capabilities and the space of content the generator's output covers.

6

Expressivity Analysis

In most published papers on PCG, the focus is on the system design and implementation, and little if any emphasis is given to analyzing the space of possible content the generators can produce. While samples of the systems' output are sometimes presented, few studies include meaningful statistical measures of the systems' performance.

Analyzing the expressive range —the space of content a generator can cover— of a generator provides an important basis on which the generated content can be evaluated. Furthermore, defining the expressive range of a generator and being able to visualize the space of content the generator covers constitute an important step if we are to compare content generated by different generators. Exploring vast spaces of content can also support creativity in several ways, including finding artifacts and allowing a designer to swiftly visualize the results of a design idea. More importantly, visualizing the content space highlights the limitations in the generator's capabilities, reveals its strength and weakness and enables an in-depth analysis of the design choices and parameters and its impact on the generator's expressivity.

In this chapter, we present a framework for expressivity analysis for 2D platform games genre. We apply this framework to analyze the content space and investigate the expressive ranges of the three content generate presented in the previous chapter, namely, the Notch generator, the parameterized generator and the grammatical evolution generator.

6.1 Expressivity Analysis

According to [Smith \[2012\]](#), a generator’s expressive range “describes the variety and style of levels that the system can generate and how sensitive that variety is to the input parameters for the generator”. [Smith and Whitehead \[2010\]](#) suggested a framework for analyzing the expressive range of a level generator by defining a set of description metrics, collecting a large number of representative samples of the generator’s capabilities, visualizing the generative space, and finally analyzing the impact of the generator’s parameters on the generator’s expressivity.

The work presented in this chapter adopts this framework for analyzing the expressive range of the three generators presented in [Chapter 5](#) and extends it through defining more informative aesthetic measures of the generators’ expressivity and applying these measures to analyze and compare the expressive ranges of three level generators of the same game. This examination allows us to explore the widest possible range of output for each generator and highlights the differences among them.

6.2 Experimental Setup

To analyze the design and the expressiveness of the generators, several statistics have been extracted from 1000 levels generated by each different generator. All generated levels have the same height and 100 blocks width each.

The experimental parameters used to generate the levels are as follows:

- Notch generator: to allow a fair comparison between the levels generated by this generator and the other levels constructed by the other two generators, unique random seeds have been used to construct the levels and the difficulty of all generated levels is set to 2.
- The parameterized generator: the advanced parameterized generator is used to generate content since it allows more variation in level design than the basic version of the generator. Levels are generated by assigning low and high values for each content feature and exploring all possible combinations. The low and high values assigned for each content feature are the following:

the number of gapes $G = 2$ or $G = 6$, the gaps width $\bar{G}_w = 5$ or $\bar{G}_w = 15$ if two gaps were generated, and $\bar{G}_w = 15$ or $\bar{G}_w = 25$ if six gaps were generated. This distinction has been made in order to allow for more variations and to ensure the possibility of jumping over the gaps. Note that \bar{G}_w/G defines the maximum width of a gap. The number of enemies can be either $E = 3$ or $E = 7$; if enemies placement $E_p = 0$, the highest probability is given to the placement of enemies around gaps and in this case, $E_g = 80\%$; if $E_p = 1$, enemies are placed around blocks with the highest probability and $E_x = 80\%$ and finally, if $E_p = 2$, then the highest probability is given to the random placement of enemies and in this case $E_r = 80\%$. The number of powerups $N_w = 0$ or $N_w = 1$ and the number of boxes $B = 0$ or $B = 15$.

The generator is allowed to freely explore the other aspects of game content such as the number of bill blasters and piranha plants, the number of coins, the differences in platform height, and the number of hills.

- The grammatical evolution generator: the existing GEVA software (O’Neill et al. [2008]) has been used as a core to implement the needed functionalities. Since this is the first time GE was applied for platform game design, we wanted to test the applicability of GE to construct IMB levels, a relatively simple fitness function have been implemented. The main objective of the fitness function is to allow for exploring the design space by creating levels with an acceptable number of chunks permitting for rich design and variability. Thus, the fitness function used is a weighted sum of two normalized measures; the first one, f_p , is the difference between the number of chunks placed in the level and a predefined threshold that specifies the maximum number of chunks that can be placed. The second, f_c , is the number of different conflicting chunks found in the design. Apparently, the two fitness functions partially conflict since optimizing f_p by placing more chunks implicitly increases the chance of creating conflicting chunks (f_c).

Figure 6.1 presents two sample levels generated by the second version of the grammar. The GE parameters used to generate these levels are as follows: 10 generations with a population size of 100 individuals, the ramped half-and-half initialization method. The maximum derivation tree depth was

6.3. EXPRESSIVITY MEASURES

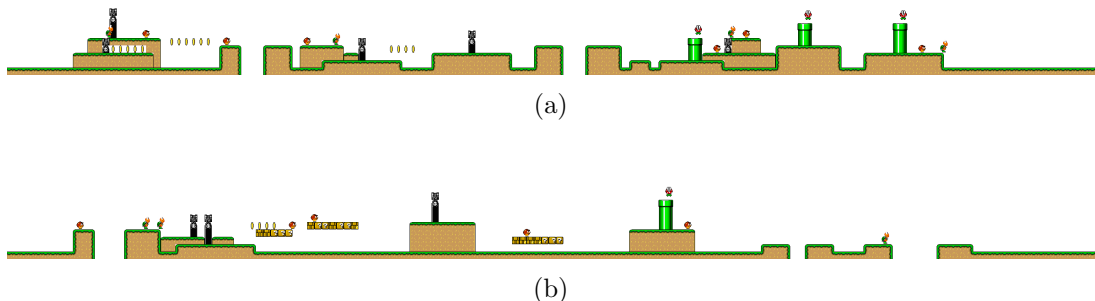


Figure 6.1: Two example levels generated by the GE-generator using the second version of the grammar.

set at 100, tournament selection of size 2, int-flip mutation with probability 0.1, one-point crossover with probability 0.7, and 3 maximum wraps were allowed. The levels generated illustrate the method ability to construct levels with varying structure; while level 6.1.(a) shows a rich design with many elements and a relatively high number of overlapping items, level 6.1.(b) display a rather flat structure containing less variation. Apparently, these two levels suit different playing styles since they exhibit different levels of challenge.

6.3 Expressivity Measures

To analyze the expressive range of each generator and compare the generators with each other, 1000 levels have been constructed by each generator using the experimental setups presented in the previous section.

6.3.1 Frequency Analysis

The frequency analysis is the simplest mean of presenting a generator’s capabilities and it is usually performed to draw a general picture of the generator’s expressive range. In this study, the statistical analysis of frequencies eight key statistical features has been performed. Figure 6.2 presents a comparison between the average values of these features that have been extracted from the

data of all levels across all generators. These features are the numbers of coins, boxes, powerups, enemies and gaps, the average gap width, as well as the enemy placement. All feature values are normalized to the range $[0,1]$ using max-min normalization. Note that for comparison purposes, the values are normalized across the content generated by all generators.

As can be seen from Figure 6.2, the GE generator appears to generate higher values for all aspects of game content except for the number and width of gaps. This might be the result of defining a rather high threshold for the total number of chunks that can be placed in the level when designing the fitness function. The generator appears to be biased towards generating a low number of gaps, a large number of enemies and boxes and placing enemies around boxes. The standard deviations are roughly comparable, though the GE generator appears to have less variation in enemy numbers and placement.

The Notch generator and the parameterized generator, on the other hand, appear to generate around the same number of boxes, coins, powerups, and gaps. The main differences between these two generators are in the number of enemies created and the width of gaps. A larger number of enemies (including piranha-tubes and bill blasters) and wider gaps have been generated in the parameterized levels compared with the ones generated by the random generator.

Note that, in the case of the parameterized generator, the average values obtained for most of the features extracted are predictable. This is because the content space generated by this generator is somehow constrained by the content features defined. Therefore, it is important to define more abstract expressivity measures that compare the generators' expressive range along other dimensions than the ones controlled.

The frequency analysis draws a picture of the generators' capabilities but a more in-depth analysis is required, if we are to examine the space of possibilities the generators' output covers and the density of the levels generated along different aspects of expressivity measures. For these reasons, we have defined several more complex level design metrics and employed them to evaluate the generated levels. In the following sections, we describe these measures and the results of applying them to examine the qualities of the generators' output. Two of these measures are similar to the ones proposed in [Smith and Whitehead \[2010\]](#).

6.3. EXPRESSIVITY MEASURES

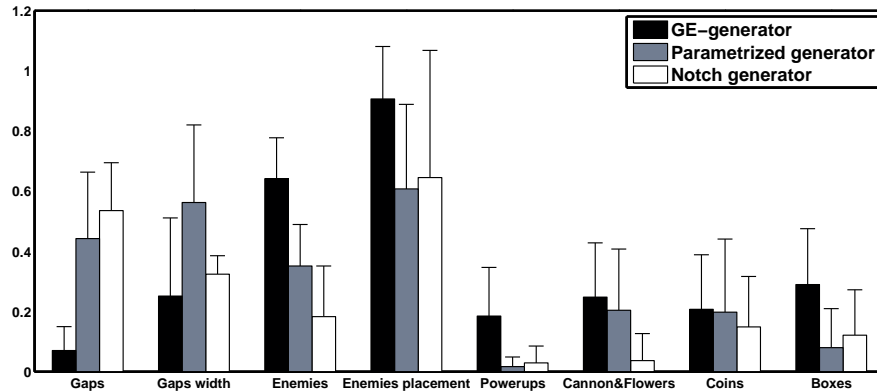


Figure 6.2: Average and standard deviation values of eight statistical features that have been extracted from all generated levels across all generators.

The rest of the measures introduced have been designed so that they provide a meaningful assessment for game designers in terms of level design and gameplay experience.

Since the expressivity of some of the generators have been constrained along some aspect of content generation (such as the parameterized generator), we tried to define expressivity measures that allow us to compare the generators' outputs along dimensions orthogonal to the ones directly controlled by the parameters.

In order to allow a fair comparison, the scores assigned for each measure have been normalized to $[0,1]$ along the levels generated by all generators using standard max-min normalization.

6.3.2 Linearity

Linearity measures how flat a level is. In IMB, linearity is affected by the existence of different types of hills along the level, as well as the differences in the platform height. A highly non-linear level is one with frequent changes in the platform height or one containing hills scattered around. A level with such characteristics requires the player to perform more jumps, gives him the possibility to reach higher places and/or presents more than one possible path to reach the end of the level. Two levels of very high and low linearity values are depicted in Figure 6.3.

We follow the approach proposed by [Smith and Whitehead \[2010\]](#) to measure

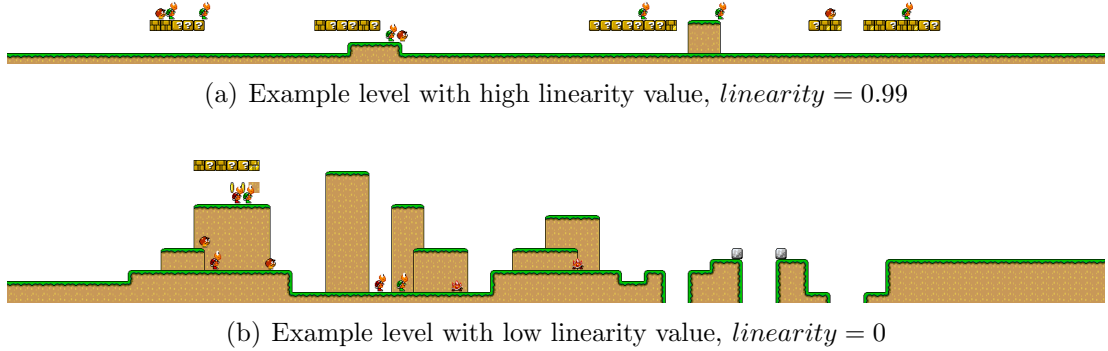


Figure 6.3: Two example levels with different linearity values.

linearity by calculating the deviations from the linear regression for each level. This has been performed by traversing the level from left to right and accumulating the values of the absolute differences between the center-point of the highest platform or hill and the corresponding point on a predefined line. The results are then uniformly normalized to $[0,1]$.

Figure 6.4 presents the average values of the linearity measure obtained from ranking the levels generated by all generators. The results show that the levels generated by the GE generator are, on average, less linear than the ones generated by Notch generator, which are in turn less linear than the ones generated by the parameterized generator.

6.3.3 Density

We defined a density measure that ranks the levels according to the summed density of segments. In IMB, hills of different height can be stacked on top of each other allowing *Mario* to reach higher places and introducing new patterns in the level design. The density is calculated by assigning a density value to each point along the width of the level according to the number of platform stacked at that point. The density of a level is the normalization of the sum of these values over the width of the level. Figure 6.5 presents three levels having extreme density values. Note that since normalization has been performed based on the

6.3. EXPRESSIVITY MEASURES

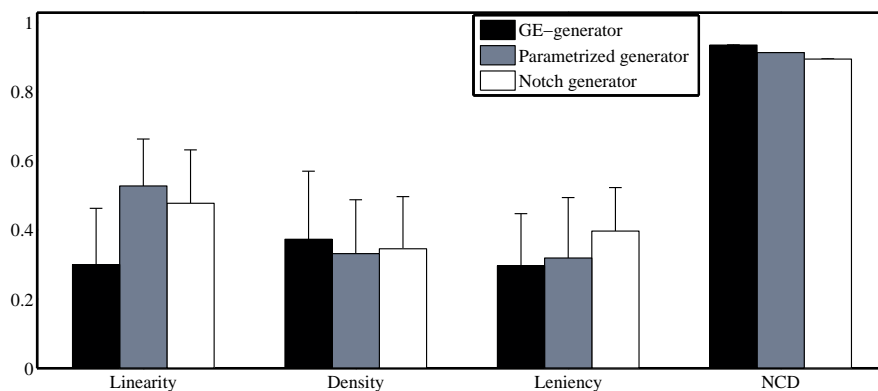


Figure 6.4: The average and standard deviation values for the expressivity measures for all generators.

density values obtained from all the levels generated, Figure 6.5.(c) is assigned a density value equals to 1 because it has the maximum density value of all the levels generated although it might be possible to manually generate level with higher density.

The density measure taken together with the linearity measure give an indication of the distribution of hills along the level. A level with a high density value can either contain hills scattered along the level or they can be stacked in one or more segments. Figure 6.5.(b) and Figure 6.5.(c) present two example levels with high density, yet having a very different distribution of hills and hence providing a very different aesthetics quality that the player experience. The linearity values assigned for these two levels, however, are 0.4 and 0.9 for the former and latter level, respectively, indicating a wide range of differences in the structure of the levels. The level with hills compressed in a small segment is assigned with a higher linearity value than the one with hills spread along the level since linearity takes into account only the highest platform at each position.

As can be seen from Figure 6.4, the GE-generator constructs levels with higher density than the parameterized and Notch generator. It's also worth noting that all generators construct levels with low average density (less than 0.5).

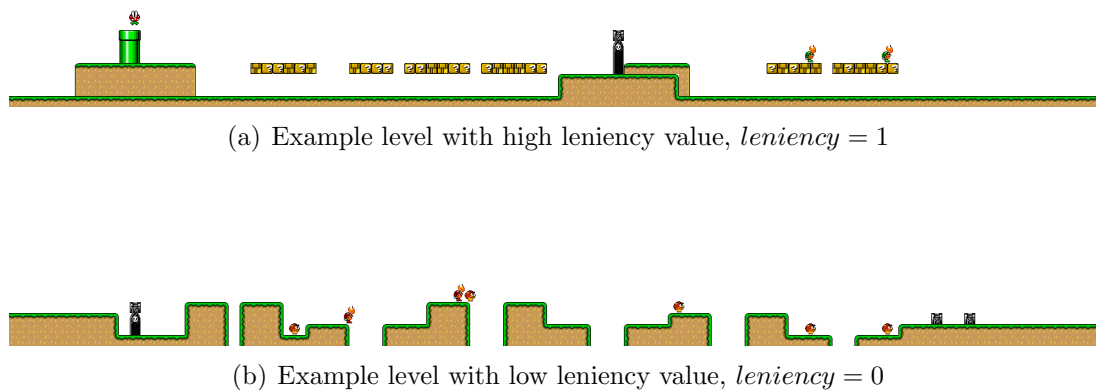


Figure 6.6: Two example levels of different leniency values.

The leniency of a level is the weighted sum of the leniency of each of the chunks presented in the level. The leniency values for all generated levels are normalized to $[0,1]$. Two levels with different leniency values are presented in Figure 6.6. Note that despite the fact that the level presented in Figure 6.6.(a) contains four enemies (two koopas, one bill blaster and one piranha plant), this level has been assigned a very high leniency value because 85% of the boxes presented in the level hide powerups.

The average leniency values obtained for the generators are presented in Figure 6.4. Notch generator constructs the most lenient levels followed by the parameterized generator, while the levels generated by the GE-generator are the least lenient.

6.3.5 Compression Distance

In order to measure the overall structural similarity between the outputs of each generator, we converted all levels into sequences of numbers representing the existence of different types of content items as well as changes in the level geometry.

The following content events have been considered when converting the levels into sequences:

- Increase/decrease in platform height

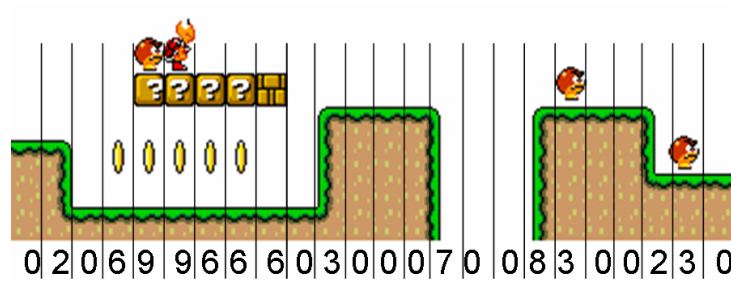


Figure 6.7: Snapshot from a level and the corresponding structure sequence representation.

- The existence/non-existence of enemies and rewarding items (coins or boxes)
- The beginning/ending of a gap

All content events considered along with their graphical representation are presented in Table 6.1. A sequence is generated by traversing the level from left to right and recording a value corresponding to content at each block. Figure 6.7 presents part of a level and its corresponding sequence representation.

The diversity of the resulting levels sequences for each generator is measured using the normalized compression distance (NCD) measure (Li et al. [2004]).









The results of applying this measure on each pair of the content sequences for each generator showed a high dissimilarity between the sequences; NCD was found to be higher than 0.6 in 93%, 91% and 89% of the cases for the levels generated by the GE-generator, the parameterized generator and Notch generator, respectively (Figure 6.4).

6.3.6 Sequential Patterns

The compression distance measure presented in the previous section provides a high level insight on the structural similarity between the levels since it compares them on the block level. More powerful and efficient sequence extraction and comparison methods are required if we are to dig deeper into the structural similarity between the levels. Therefore, sequence mining methods have been employed to extract meaningful sequence patterns about the structure of the levels.

6.3. EXPRESSIVITY MEASURES

Table 6.1: The content events considered when converting levels into sequences and their graphical representation.

Graphical Representation	Content Event
	Flat platform
	Increase in the platform height
	Decrease in the platform height
	The beginning of a gap
	The end of a gap
	An enemy
	A coin, block, or brick block
	An enemy with a rewarding item




The sequences obtained when converting each level into a sequence of numbers in the previous section are used to extract design patterns that occur frequently in the levels generated by each generator. The frequent sequence mining algorithm, Generalized Sequential Patterns (GSP) (Srikant and Agrawal [1996]) presented in Section 3.2.4, have been employed to find frequent sequence patterns within the dataset of structural sequences. GSP allows us to define a time constraints within which adjacent events can be considered elements of the same pattern. For example, using GSP, we can extract patterns such as (, , ) which indicates a decrease in the platform height followed by a gap although these events are not directly adjacent. For the experiments presented in this study, we consider frequent subsequences of length three that occur in at least 500 levels (half the size of the full dataset). The max_{gap} value used is 5, allowing two items occurring within five block difference to be considered as belonging to the same pattern.

Table 6.2 presents a subset of the frequent patterns extracted from the levels generated by each generator with the highest and lowest number of occurrences. Note that the lowest possible number of occurrences is the min_{sup} value which is set to 500 in this experiment. Note also that these patterns are not necessarily directly connected since they are allowed to occur within a five block distance.

We perform inter and intra analysis to investigate structural similarities based on the frequent patterns obtained from each generator. Different set of patterns have been extracted from each generator’s outputs. The number of frequent

Table 6.2: The five most occurring and five least occurring sequence patterns of length three in the levels generated by each generator. The numbers indicate the number of occurrences of the patterns in all sequences of levels.

Notch generator	Parameterized generator	GE-generator
:1000	:1000	:1000
:1000	:1000	:1000
:1000	:1000	:1000
:1000	:1000	:1000
:1000	:1000	:1000
:1000	:1000	:1000
:518	:509	:501
:519	:510	:502
:522	:512	:503
:535	:517	:503
:538	:518	:515

patterns extracted reflects the diversity of the content generated, the largest number of patterns have been extracted from the parameterized generator (89 frequent patterns) followed by the GE-generator with 87 frequent patterns while the number of frequent patterns found in the Notch generator is 83.

To investigate for inter-similarity between the generators' outputs, we calculate the intersections between the set of patterns extracted from the levels generated by each generator. The results obtained showed that 60% of the patterns extracted from the levels generated by the GE-generator and the Notch generator overlap while an intersection of 50% have been found between the GE-generator's patterns and the parameterized generator's patterns.

Surprisingly, the patterns extracted from levels generated by the parameterized generator overlap in only 28% of the cases with the patterns extracted from the levels generated by the Notch generator. This was unexpected since the parameterized generator has been implemented by biasing the random generator and hence we anticipated that these two generators would sustain structural similarity up to a good degree. To further investigate the similarity in more details, we counted the number of occurrences of the patterns extracted from each generator in the levels generated by the other generators.

The results obtained showed that despite the low intersection percentages between the frequent patterns extracted from the levels generated by the parameterized and Notch generator, these common patterns occur equally frequently in the levels generated by both generators while the patterns that overlap with the ones found in the levels generated by the GE-generator occur less often.

6.3.7 Histogram comparison

The expressive measures presented in the previous sections analyze the generators' output along one dimension. It is interesting, however, to be able to visualize the space of content a generator's output covers and the intensity of levels generated for each point in this space. Such an analysis gives a more detailed insight of the generator's capabilities and highlight the generator's strength and weakness which could potentially be used by a level designer to compare different generators or to adjust the generator's parameter to manipulate the output space according to the designer's preferences.

The expressive range of a generator can be analyzed by plotting the histogram that illustrates the distribution of the generated levels along the expressivity measures. The 1000 levels generated by each generator have been processed and ranked by the linearity, leniency and density measures. Figure 6.10, 6.9 and 6.8 present the expressive ranges obtained for the GE-generator, the parameterized generator and Notch generator, respectively.

Different distributions have been obtained for each measure across the generators. The GE-generator, as can be seen from Figure 6.10 and the parameterized generators (Figure 6.9) appear to be slightly biased according to the linearity measure; while the GE-generator constructs levels that are slightly non-linear, the parameterized generator appears to be generating more linear levels. On the other hand, both generators appear to be biased towards generating non-lenient levels. It is interesting to note, however, that the Notch generator (Figure 6.8) constructs levels with a distribution for the linearity that approximates the normal distribution around 0.5. This generator appears to be very biased towards generating averagely lenient levels (more than 80% of the levels have a lenience value between 0.3 and 0.5). Very small percentage of the levels generated by all

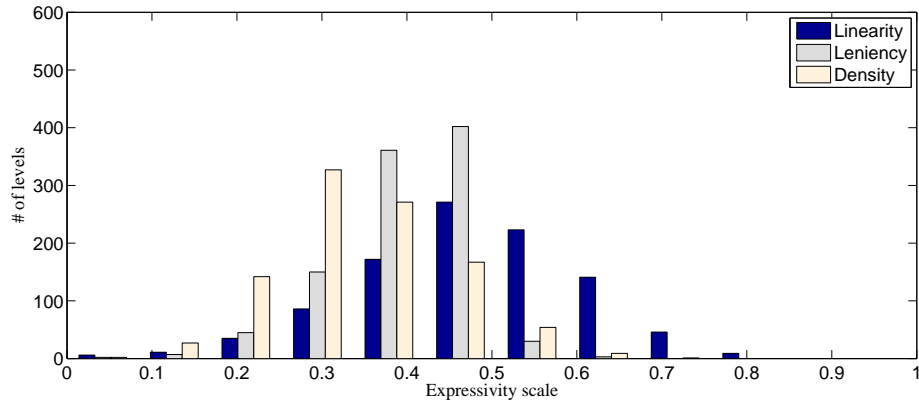


Figure 6.8: The histograms of the linearity, leniency and density measures for the 1000 levels generated by Notch generator.

generators fall in the extreme ranges of the expressivity measures.

We anticipated the bias towards generating linear levels by the parameterized generator since in IMB levels, the flat platform is the basic element when designing the levels and the addition of hills and the changes in the height are supplementary requirements in order to allow richer design diversity and gameplay experience. Also, this generator has been designed to generate levels according to a predefined set of features that resulted in highly condensed levels, leaving a few number of segments where a hill can be generated.

In order to be able to compare the expressive ranges for the three generators among each other, histograms have been created illustrating the density of levels for each expressivity measure along all generators as illustrated in Figures 6.11, 6.12 and 6.13.

Unsurprisingly, the parameterized and Notch generators appear to generate similar levels according to linearity compared to the levels generated by the GE-generator (Figure 6.11). This was anticipated since the parameterized generator is a modified version of Notch generator. However, the shift in the center of the distribution of the levels generated by the GE-generator along the linearity dimension, compared to the ones obtained from the parameterized and Notch generator, can be explained by the different methodology used by this generator when constructing the levels.

All generators appear to have similar distributions for the levels along the

6.3. EXPRESSIVITY MEASURES

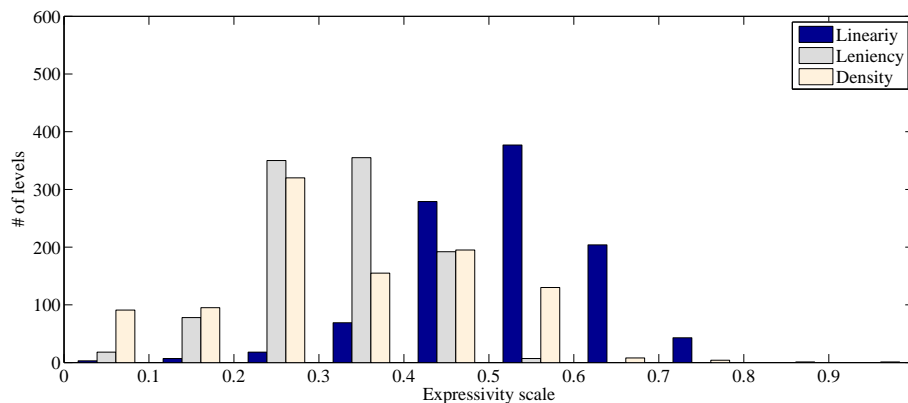


Figure 6.9: The histograms of the linearity, leniency and density measures for the 1000 levels generated by the parameterized generator.

leniency dimension as can be seen from Figure 6.12. Most of the levels generated by the three generators have a low to average lenient values while very few of them are lenient. Notch generator, between the other two generators, appears to be the most biased towards generating averagely lenient levels. This can be explained by the design choice of the difficulty parameter that enforces all constructed levels to have a difficulty value equals to 2. All levels constructed by the parameterized generator, on the other hand, have been generated with either three or seven enemies (as described in Section 6.2) resulting in a shift towards non-lenient levels. At least two enemies (koopas and/or goombas) are placed in the levels constructed by the GE-generator. The other types of enemies, such as bill blasters and piranha plants, have the same probability of occurrence in the levels as any other chunk. This explains the low leniency value assigned to most of the levels generated by the GE-generator.

The level distribution along the density dimension (Figure 6.13) vary among the three generators with all of them generating low to average density levels. The shift in the density values obtained from the levels generated by the GE-generator can be explained by a design choice which is implicitly imposed by the design grammar; the range of possible height for each chunk generated has been constrained in a way that the chunk will be reachable by *Mario*. The same argument for generating linear levels by the parameterized and Notch generator holds for explaining the shift towards generating levels of low density; the levels

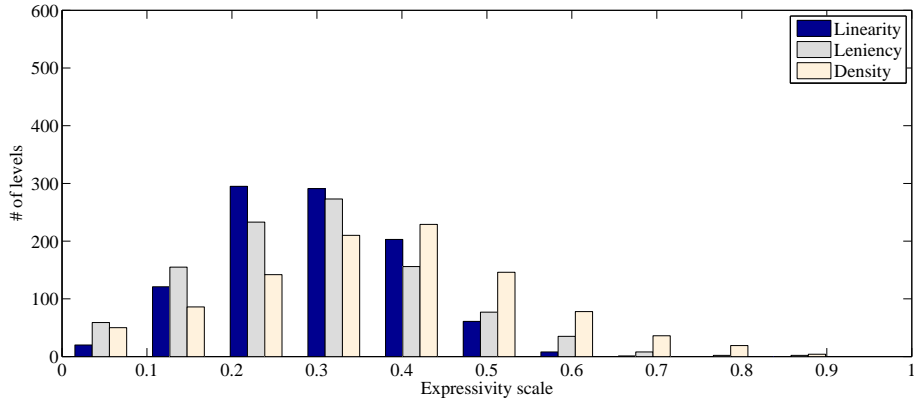


Figure 6.10: The histograms of the linearity, leniency and density measures for the 1000 levels generated by the GE-generator.

have been constructed with a flat ground as the basic infrastructure in both generators. The parameterized generator imposes more implicit constraints on density by enforcing each constructed level to satisfy a set of design parameters leaving little room for the other design elements than the one specified to be placed (note that hills are not one of the parameters considered by the parameterized generator and hence they occur less often than the other chunks).

The Notch generator appears to cover a narrower expressive range for all measures than the other generators. None of the generators was able to express a uniform distribution of levels along the expressivity measures defined. Nevertheless, it is not clear whether this is desirable and necessitates covering a wider range of player preferences.

The statistical analysis of these measures across all levels generated by each generator (Table 6.3) showed strong positive correlations between linearity and leniency for the levels generated by all generators, while strong negative correlations have been obtained between linearity and density and leniency and density.

The positive correlation between linearity and leniency can be explained by the interconnection between the content elements involved when measuring these scores. The presence of gaps and enemies (bill blasters and piranha plants), which mostly implies changes in the platform height, leads to generating levels with low linear and lenient score. The negative correlation between linearity and density, on the other hand, points out a bias in the generators towards generating levels

6.3. EXPRESSIVITY MEASURES

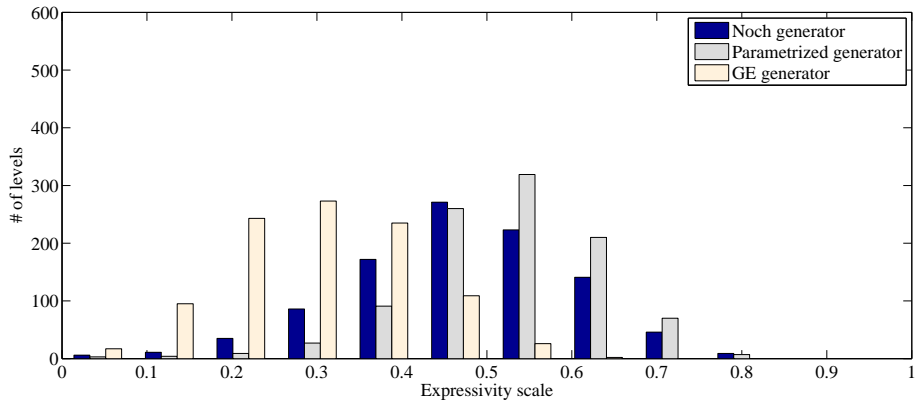


Figure 6.11: The histograms of the linearity measure for the 3000 levels generated by Notch generator, the parameterized generator and the GE- generator.

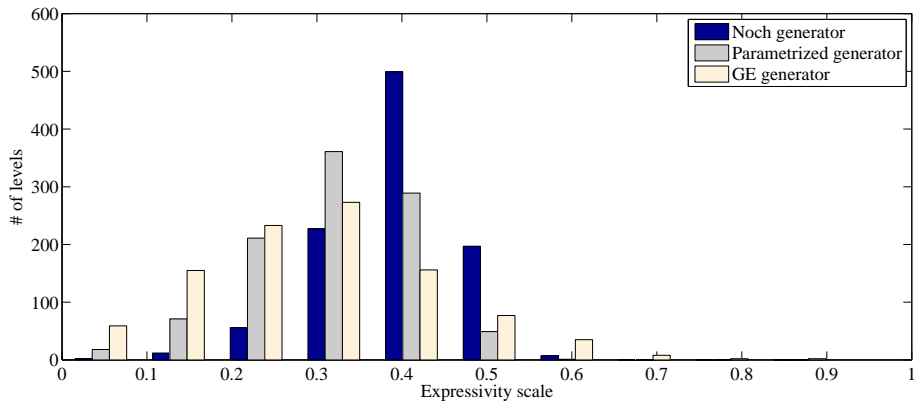


Figure 6.12: The histograms of the leniency measure for the 3000 levels generated by Notch generator, the parameterized generator and the GE- generator.

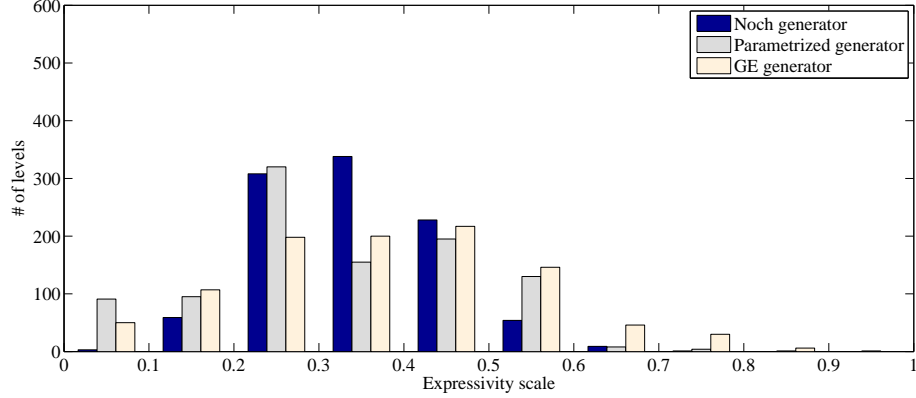


Figure 6.13: The histograms of the density measure for the 3000 levels generated by Notch generator, the parameterized generator and the GE- generator.

with hills spread along them rather than stacked on top of each other.

Table 6.3: Testing for correlation between the obtained scores for each measure across the three generators. The significant differences ($p - value < 0.01$) are presented in bold. The sign of the correlation is presented in parentheses.

	Notch generator	
	<i>Leniency</i>	<i>Density</i>
<i>Linearity</i>	$1.52 * 10^{-29}$	(-) $3.61 * 10^{-20}$
<i>Leniency</i>		(-) $4.32 * 10^{-51}$
	Parameterized generator	
	<i>Leniency</i>	<i>Density</i>
<i>Linearity</i>	$19.85 * 10^{-6}$	(-) $2.99 * 10^{-21}$
<i>Leniency</i>		(-) $4.88 * 10^{-11}$
	GE-generator	
	<i>Leniency</i>	<i>Density</i>
<i>Linearity</i>	$2.86 * 10^{-51}$	(-) $2.29 * 10^{-155}$
<i>Leniency</i>		(-) $6.15 * 10^{-23}$

6.4 Summary

Analyzing the expressive range of a content generator and being able to compare different content generators is vitally important for understanding the generators' capabilities, strength and weakness and for providing quantitative measures of the

generators' performance.

This chapter presents a framework for comparing content generated by different generators with different generation methods for the same game. A number of expressivity measures have been defined to test the generators' capabilities and the space of content the generators' output covers. The expressive range of each generator has been analyzed and quantitatively compared to the other generators by plotting the histograms of 1000 levels generated by each generator across the expressivity scales defined. The results obtained showed different characteristics of each generator and a wide variety in the space of content each generator covers.

The expressivity analysis highlights limitations in the expressivity of each generator. For example, the design grammar in the GE-generator is unable to generate levels with high density due to the height constraint defined in the grammar forcing the generated chunks to be placed within a predefined height limit to ensure playability. One possible solution is to define a constraint-free grammar and play-test the generated levels to check for the playability. This can be done automatically by exploiting AI agents that pass through the levels and check for possible path from the start to the end, and/or check whether all chunks generated are reachable. Another solution is to adopt context-sensitive grammar such as attribute grammars to control the parameter values of the solutions as they are being generated during the mapping process (O'Neill et al. [2004]).

The measures presented in this chapter provide a mean to compare content but covering a wider range along these measures doesn't necessarily mean better content quality. Designers' knowledge or player experience models, that map game content to players' reported affect, can be used as content quality measures to rank the content generated according to the gameplay experience it provides.

The generator's parameters highly influence its expressive range. For example, the fitness function and the design grammar used by the GE-generator can bias the search towards different kinds of maps. Analyzing the effect of design parameters on the generator's expressive range constitutes a future direction.

The ultimate aim of the work presented in this chapter is to provide a tool for players and game designers where they can explore the content space and swiftly visualize the impact of design choices on the generator's expressive range. There is plenty of room for future improvements including investigating new

methods for visualizing the expressive range, and giving real time feedback on the impact of changing design parameters on the content space. The framework presented for analyzing the expressive range of a generator can potentially be used by game designers or players to generate content with user defined expressivity parameters. This could be done by biasing the content generated according to these parameters.

7

Modeling Player Experience

An algorithm that could automatically judge how engaging or interesting a particular piece of game content is would be useful for several reasons. One strong reason is that such a method would help us to automatically or semi-automatically generate good content, another is that analysis of the algorithm could help us understand what players like in games. As players tend to vary significantly in their preferences it would further be useful to have an algorithm that, given information about a particular player, could predict the appeal of the game content for that player. Finally, having an algorithm that could observe a human playing a game and accurately judge what the human is experiencing as he/she is playing the game would also be useful, as this could allow us to adapt the game to the player, and also help us understand how human affect is expressed in behavior.

When constructing player experience models, one should identify relevant features from game content and gameplay that affect player experience. A large set of features has been extracted as presented in the previous chapter, and not all of these features are necessarily relevant for modeling player experience.

In this chapter, we present the process followed to construct models of players' experience. Neuroevolutionary preference learning is used for approximating the unknown relationship between selected subset of relevant features and players' reported affects. In other words, we use artificial evolution for shaping artificial neural networks (ANNs) whose output matches the reported (pairwise) preferences of the players.

7.1 Neuroevolutionary Preference Learning

For the player experience modeling problem under investigation, neuroevolutionary preference learning, as presented in Section 3.1.6 has been used. The mechanism attempts to approximate a function $f(\cdot)$ that predicts whether $g_a \succ g_b$ holds, where g_a and g_b represents particular instances of individual gameplay sessions, given the following inputs:

- A set of gameplay features extracted from the interaction between the player and the game, and a set of game content features.
- A set of n training instances $Z = \{g_i | i = 1, \dots, n\}$ comprising vectors of the measured values of those features for the different variants of the game played by various players.
- A set of m pairwise preferences $G = \{g_a \succ g_b | a, b = 1, \dots, m\}$ in which players reported which of the two game variants they preferred.

The following sections describe the feature selection method adopted to extract relevant features from each gameplay instances and the neuroevolutionary preference learning methodology proposed to learn the ranking function.

7.2 Feature Extraction

While playing the games, several features can be recorded to further be used as indicators of players' affects, performance, playing characteristics and style. The models of player experience can be defined by the set of features extracted from the gameplay session. In this dissertation we focus of models constructed from gameplay data (*gameplay-based PEM*), visual cues of players behavior (*objective PEM*), player's reported affects (*subjective PEM*) and *hybrid PEM* build from a combination of these features. The features extracted in each of these categories have been chosen in order to capture a wide range of variations of players behavior and to allow the construction of accurate estimators of player experience. These features as well as details about their extraction methodology are described in details in Chapter 8.

7.3 Feature Selection

Feature selection is a critical step for efficient knowledge discovery when handling a large amount of data. The selection of the relevant subset of features not only helps us reduce the dimension of the input space resulting in more accurate models that are easier to analyze, but it also eliminates noisy features that are irrelevant for the player experience modeling as well as improving the model's generalization capabilities.

Feature selection has been applied as a first step when modeling players' experience. The input space constitutes of the different combinations of the set of features explained in the previous chapter. The values of all features have been normalized to the interval [0,1] using the max-min normalization function.

The selection of the relevant subset of features for predicting different reported affective/cognitive states is achieved though Sequential Forward Selection (SFS) (Yannakakis et al. [2009, 2008]).

Neuroevolutionary preference learning (Section 3.1.6) is used to measure the performance of the subset of features selected by SFS. The method works in two steps: (1) feature selection and (2) feature space expansion. In the first step we train single-layer perceptrons (SLPs) as a mapping between selected features by SFS and reported preferences.

Since the data is assumed to be a very noisy representation of the unknown function between the input features and players' reported preferences due to the wide variety of playing styles and the high level of subjectivity of players' reports, we adopt more robust estimators of players' preferences by using more complex nonlinear functions such as MLP. Simple multilayer perceptrons are used to expand the subset of features selected using SLPs. The subset of features derived from SFS using SLP is used as the input of small MLP models containing one layer of two hidden neurons and SFS is used again to extract additional features from the set of remaining features allowing features with more complicated nonlinear relationships to be selected. An overview of the process followed is depicted in Figure 7.1.

The single neuron uses the sigmoid (logistic) activation function $g(e, p) = 1/(1 + e^{-pd_j})$ where d_j is the difference between the ANN output values for a pair

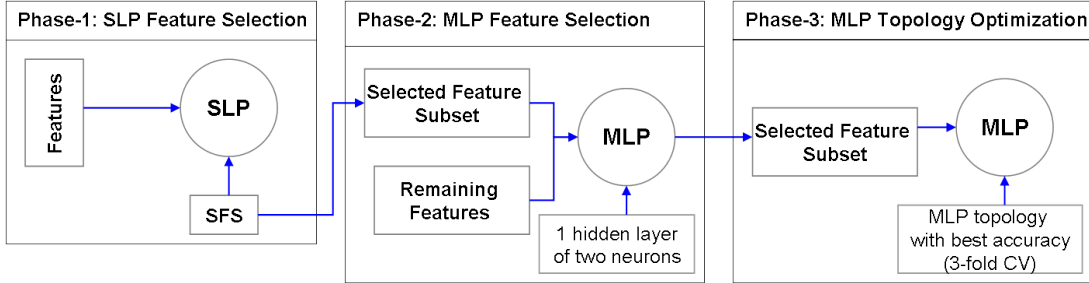


Figure 7.1: The three-phase player experience modeling approach followed.

j and $p = 30$ if there is an agreement between the network output and player’s preferences and $p = 5$ otherwise. Both the sigmoid shape and the selection of the p values are assigned experimentally after investigating their effects on the classification accuracy. Connection weights take values from -5 to 5 to match the normalized input values.

The data is randomly divided into thirds where training and validation data sets consisting of $2/3$ and $1/3$ of the data, respectively. The quality of a feature subset is determined by the average classification accuracy of the model in three independent runs using three-fold cross-validation method on the three independent training and validation sets.

7.4 Model Optimization

As mentioned earlier, the underlying function between gameplay, content features and reported players’ preferences is considered to be complex and robust estimators are required if we are to accurately model the features-affect relationship. The first critical step when constructing accurate estimators is to reduce the dimension of the input space and eliminate noisy features. This step has been achieved in the previous section by using SFS implemented through small MLPs of one hidden layer of two neurons. Although these MLPs were able to capture the non-linear relationship between selected features and players’ preferences with relatively low computational effort, this relationship might be more complex and requires more powerful MLPs with more sophisticated structure. Therefore, once all features that contribute to accurate simple MLP models are found we

optimize the topology of models using neuroevolutionary preference learning. We start with a simple MLP topology of one hidden layer of two neurons; we then increase the number of neurons up to ten by adding two neurons at each step. Further, we investigate MLPs with two hidden layers, with up to ten neurons in the first and second layer. Again, the number of hidden neurons starts at two and increases by adding two neurons at each step; this sums to 30 different MLPs topologies which are tested for each input vector.

Just as in the feature selection, the performance of each MLP is obtained through the average classification accuracy in three independent runs using 3-fold cross validation. Parameter tuning tests have been conducted to set up the parameters' values for neuroevolutionary user preference learning that yield the highest accuracy and minimize computational effort.

Unless otherwise mentioned, for the rest of this dissertation, neuroevolutionary preference learning uses a population of 100 individuals, and evolution runs for 20 generations. A probabilistic rank-based selection scheme is used, with higher ranked individuals having higher probability of being chosen as parents. Finally, reproduction was performed via uniform crossover, followed by Gaussian mutation of 1% probability. Several other setups have been examined and the one followed provided the best tradeoff between the computational efforts and the modeling accuracy.

7.5 Summary

In this chapter we introduced the framework for feature selection and player experience modeling that will be followed throughout this dissertation. We choose to focus on neuroevolutionary preference learning as a modeling approach due to its successful previous applications on similar problems where it has proved to construct estimators of players' experience that have been found to be more accurate than estimators constructed using a number of other approaches including large margin classifiers and Bayesian learning (Yannakakis et al. [2009]).

While applying other approaches for preference modeling and feature selection and exploring other methods that enable evolving the networks topologies along with their weights are very interesting research directions and might very well be

suited for solving the problems of player experience modeling, these approaches are not the key focus of this dissertation. This being said, further investigation and analysis of issues explored by this dissertation will with no doubt help in the development of future accurate player experience modeling and adaptation algorithms. The central attention of this work, however, is given to closing the affective loop in games for which accurately assessing player experience is one important aspect. Other aspects such as exploring features from different modalities as indicators of players experience, investigating different approaches for feature representation, defining the frequency of content adaptation, analyzing the impact of content features on player experience, constructing and evaluating an efficient adaptation framework are as essential for efficient adaptation and, therefore, will be explored and given attention in this work.

8

Data Collection and Feature Extraction

Several studies can be found in the literature on analyzing the relationship between game content and player experience. Most of these studies, however, have tackled this problem from a top-down perspective relying on theoretical rather than computational models of player experience. However, even if the theoretical models are empirically validated and sufficiently extensive to allow prediction of player experience in a wide range of situations, they would also need to be expressed quantitatively in order to be incorporated within an adaptation algorithm. They would therefore need to be grounded in measurable quantities.

The alternative, complimentary approach is to create data-driven (bottom-up) models based on collecting data about games, game content and player behavior, and correlating this data with data annotated with player experience tags.

The very first step towards constructing computational models of player experience and accurately adapting game content according to specific player needs is to collect data from players. We can then model the relationship between game content and player experience. This can be done through crowd-sourcing data from a wide range of players with different demographic backgrounds and variety of playing style and expertise.

Another critical issue is the identification of relevant features from game content and gameplay that affect this experience. Several approaches can be followed

on how to represent the features, what features to extract, how to extract them and how to relate them to specific affective/cognitive states. The selection of these features, the method followed to represent them and the choice of their extraction approach have a great impact on both the efficiency and the performance of the modeling.

In this chapter, we propose a protocol to collect data from players for the purpose of quantitatively analyzing and modeling player experience. The proposed protocol is followed to collect four datasets of different numbers of participants and variant types of features capturing different aspect of game content and playing behavior thus allowing an in-depth analysis of the gameplay experience.

8.1 Experimental Protocol

A game survey study has been designed to collect subjective affective/cognitive reports expressed as pairwise preferences of subjects playing different variants (levels) of the test-bed game, *Infinite Mario Bros*, by following the experimental protocol proposed by [Yannakakis et al. \[2009\]](#).

According to the protocol, each subject plays a set of two games. The games played differ in the levels of one or more of the content features. A detailed description of the procedure followed is as follows.

1. An introduction page presents the game to the player and contains information about the procedure that will be followed. The player is being told that during the session she will play two games and will be asked to answer a few questions about her game experience.
2. Then, a demographics page is presented which is used to collect the demographics data.
3. The player is introduced to the keys that can be used to control *Mario* and their functionalities.
4. After these introductory steps the player is set to play the first game (game *A*). The player is given three chances to complete the game level (i.e. she has three lives). If she fails in the first trial the game is reset to the starting

point and the player is set to try again. The game ends either by winning in one of the three trials “lives” or by failing in the third one.

5. After finishing game A , a second game (game B) is presented to the player and the player is set to play. The player is given three chances (i.e. three Mario lives) to complete the level and the same rules apply as in game A .
6. After completing a pair of two games A and B , the player is asked to report the preferred game for different emotional dimensions through a 4-alternative forced choice (4-AFC) questionnaire protocol.
7. The player then has the choice to either end the session or to continue. In the latter case, a new pair of two games is presented and the procedure is repeated starting from step 4.

Player experience models can be built on different types of data collected from players which, in turn, define different approaches to player experience modeling (Yannakakis and Togelius [2011]). In our study we rely on features extracted from the game content and data expressed by players themselves about their playing experience along with features of how they play the game (i.e. player behavioral features) and we construct our models based on this data.

Different datasets have been constructed following the same protocol described previously. The datasets differ in the generator employed to construct the levels used in the experiment, the features collected about game content and gameplay and the number of participants.

In the following sections we will describe the types of data have been collected from hundreds of players playing *Infinite Mario Bros*. Note that not all of these types have been recorded for all datasets; some of these types appear in some datasets while not in others.

8.2 Content Data

Direct features of game content have been recorded for all the levels generated in the experiments. As mentioned earlier, the direct content features are also named *Controllable* as they are used to generate the levels in the parameterized

generators and are varied to make sure several variants of the game are played and compared. These features are the same ones presented in Section 5.3 and their values have been recorded from all the levels generated.

In addition to the direct (controllable) features, the full structure of the levels has also been saved in some of the experiments permitting sequential content features to be extracted. The topology of the levels is later converted into sequences of numbers representing different types of game items and sequence mining techniques are applied to extract useful patterns from the resulted sequences. More information about sequence generation and pattern extraction is discussed in Section 8.6.2.2.

8.3 Gameplay Data

The same two types of features considered for content data have also been investigated for gameplay data. Several features have been directly extracted from the data recorded during gameplay in all experiments. These features represent frequencies of performing actions or interactions with game elements and the choice of these features is made in order to be able to represent the difference between large varieties of Infinite Mario Bros playing styles. Example features include how often the player jumped, ran, died, how much he spent moving left, and how many enemies he killed for the different type of opponents. These features cannot be directly controlled by the game as they depend on the player's skill and playing style.

In some experiments, different player actions and interactions with game items and their corresponding time-stamps have been recorded. These events are categorized in different groups according to the type of the event and the type of interaction with the game objects. The events recorded are the following:

- **Winning:** This event is generated when the player wins.
- **Losing:** This event is generated when the player loses. An extra attribute is associated with this event to define the type of object that causes the death. This attribute can take one of the following values: koopa, goomba, piranha plant, bill blaster or gap.

- Interaction with game items: This event is generated when the player collects items or interacts with intractable game objects. The event has an attribute that defines the type of the object that can take one of the values: coins or one of the different types of boxes (blocks or rocks).
- Interaction with enemies: This event is generated when the player kills an enemy using one of the possible defeating methods. The event has two attributes that define the type of the action performed to kill the enemy (stomp, shoot fire balls or unleash a koopa shell) and the type of enemy killed (koopa, goomba, piranha plant or bullet).
- Changing Mario mode: Mario can be in one of the following modes: small, super or fire. Whenever Mario mode is changed, an event is generated with the information about the type of the new mode.
- Changing Mario state: An event is generated whenever Mario changes his state between: moving left/right, jumping, running and ducking.

Each of the above-mentioned events is associated with a time stamp specifying the time within which the event occurred. For the last two events (changing Mario mode and changing Mario state) two time stamps, instead of one for the other events, are saved marking the start and the end time of the event.

Several methods can be applied to extract useful features from the above mentioned events. Feature extraction and pattern mining methods have been applied on the data and are detailed in Chapter 8.

8.4 Player Experience

We rely on self-reported annotations based on previous research in which very accurate player experience models of self-report affective/cognitive states were constructed (Pedersen et al. [2010]; Yannakakis and Hallam [2007]). However, a number of limitations are embedded in the players self-reporting experience modeling including noise due to learning and self-deception, disruption to game play experience, sensitivity to memory limitations as well as their low evaluation

bandwidth, providing information on the whole experiments rather than continuously throughout time (Yannakakis and Hallam [2011a]; Yannakakis and Togelius [2011]).

In order to minimize these effects we rely on annotated player experience data collected via a 4-alternative forced choice questionnaire presented after game sessions. The questionnaire asks the player to report the preferred game for three user states: *engagement*, *challenge* and *frustration*. The selection of these states is based on their extensive use in other game survey studies (Baker et al. [2010]; Byrne [2005]; Chanel et al. [2008]; Gilleade and Dix [2004]; Pedersen et al. [2010]) and our intention to capture both affective and cognitive/behavioral components of gameplay experience (Yannakakis and Togelius [2011]).

The questionnaire protocol is presented after each pair of games and gives the players the following alternatives:

- game A [B] was/felt more E than game B [A] (cf. 2-alternative forced choice);
- both games were/felt equally E or
- neither of the two games was/felt E .

where E is the player experience state under investigation.

8.5 Head Movement Features

Despite our attempts to minimize the self-reporting limitations, their affects are still undeniable. We took a step further to overcome these limitations by conducting an experiment to assess the estimation of players' affect while keeping the players engaged in the game and minimizing disruption. This has been done by introducing *head movement features*; a set of head movement parameters extracted for creating behavioral correlations to game events by analyzing video recording of players.

In our experiments, the above-mentioned protocol (see Section 8.1) has been followed. The only difference in the protocol of this experiment is a page added

just after step 3 informing the player that her game sessions will be video recorded and analyzed.

Subjects were seated in front of a computer monitor; the upper part of their body was monitored by a camera. In most cases players were left alone in the rooms they were playing in and, whenever this was not possible, everyone was asked not to distract them.

With this head movement data we desire to examine the relationship between a series of *head movement features* along with gameplay and content features and player experience tags.

8.6 Datasets

Four different datasets have been constructed based on different experiments conducted to collect data from players. The datasets differ in the number of participants and the types of data collected. In the following sections, a detailed description of each of these datasets is given along with the experimental protocol followed when constructing each particular dataset.

8.6.1 Dataset 1: Basic Parameterized Generator

A similar protocol to the one described in Section 8.1 has been employed to construct this dataset.

The basic version of the parameterized generator has been used to construct the levels (Section 5.3.1.1). The game sessions presented to players have been constructed using a level width of 320 Infinite Mario Bros units (blocks), the same size usually employed when generating levels for the original Super Mario Bros game. Two states, low and high are assigned to each of the content features in order to ensure variability and investigate the effect of the design choices on players' experience.

While varying the values of the content features used to control the content generation, the other content parameters such as coins, enemies, coin blocks, powerups and empty blocks are fixed to 15, 3, 4, 2, and 8 respectively.

A Java applet has been designed and released on the Internet to collect data

from players. Since four content features with two states each have been used to construct the levels, leading to $2^4 = 16$ different variants of the game, and — according to the protocol — the games are presented in pairs, the minimum number of experiment participants required so that all combinations are presented is determined by $C_2^{16} = 120$; i.e. the number of all combinations of 2 out of 16 game variants.

The data gathering resulted in a dataset of 654 game pairs played by 327 players, more than twice as many as the participants needed to play-compare all the possible combinations of pairs, meaning that each pair is presented at least twice in the resulted dataset.

Since this was our first experiment, the data collected was in its simplest form. The following sections present the three types of features extracted from each session.

8.6.1.1 Content Features

Direct features of game content have been saved for each level played. For this experiment, the values of the four content features employed by the generator while constructing the levels as presented in Section 5.3.1.1 have been recorded, namely, the number of gaps, average width of gaps, and gaps entropy, as well as a switching feature that defines the percentage of the level played in the left direction.

8.6.1.2 Gameplay Features

Gameplay characteristics are also represented as direct features of how the user plays the game. These features characterize each individual playing style and their values are unique to each player. The choice of the direct gameplay features is made in order to be able to capture the difference between a large variety of Infinite Mario Bros playing styles. The features presented in Table 8.1 are extracted from the gameplay data collected and are classified in five categories: time, interaction with items, interaction with enemies, death and miscellaneous.

8.6. DATASETS

Table 8.1: Features extracted from data recorded during gameplay.

Category	Feature	Description
Time	t_{comp}	Completion time
	$t_{lastLife}$	Playing duration of last life over total time spent on the level
	t_{duck}	Time spent ducking (%)
	t_{jump}	Time spent jumping (%)
	t_{left}	Time spent moving left (%)
	t_{right}	Time spent moving right (%)
	t_{run}	Time spent running (%)
	t_{small}	Time spent in Small Mario mode (%)
Interaction with items	t_{super}	Time spent in Super Mario mode (%)
	n_{coin}	Free coins collected (over all coins existent)
	n_{block}	Coin blocks pressed or coin bricks destroyed (over all blocks and bricks existent)
	$n_{powerups}$	Powerups pressed (over all powerups existent)
Interaction with enemies	n_{box}	Sum of all boxes pressed or destroyed (over all boxes existent)
	k_{flower}	Times the player kills a bullet ball or a piranha plant (over all bill blasters and piranha enemies existent)
	k_{goomba}	Times the player kills a goomba or a koopa (over all goombas and koopas existent)
	k_{stomp}	Opponents died from stomping (%)
Death	$k_{unleash}$	Opponents died from unleashing a koopa shell (%)
	d_{num}	Total number of deaths
Misc	d_{cause}	Cause of the last death
	n_{mode}	Number of times the player shifted the mode (Small, Super, and Fire)
	n_{jump}	Number of times the jump button was pressed
	$n_{misc.Jump}$	Difference between the total number of gaps and the total number of jumps
	n_{shoot}	Number of times the player shoots a fire ball
	n_{duck}	Number of times the duck button was pressed
n_{state}	Number of times the player changed the state between: standing still, run, jump, moving left, and moving right	

8.6.1.3 Player experience

After playing a set of two games in pair, players were asked to report the preferred game for three emotional dimensions; fun, challenge and frustration, through the 4-AFC. Note that in this experiment players were asked to report the preferred game for fun instead of engagement. Several studies reported in the literature used the word “fun” to capture children’s notion of an entertaining experience since the term seems to naturally fit the child’s environment (Read et al. [2002]; Yannakakis et al. [2008]). We decided to change the terminology from fun to engagement for two main reasons: (1) mostly adults have participated in our experiments, and (2) a recent study by Calleja [2011] showed that the notion of fun is too vague, as the term “fun merely implies a clustering of positive emotions surrounding an activity...The concept is as unhelpful to the designer as it is to the analyst. It is more productive to focus on a notion of engagement”. Engagement, as described by Calleja [2011] “engagement describes the player’s interest in engaging with the game. This is the most basic form of involvement”, seems to be the best alternative to fun, and therefore we use this term for the rest of the experiments presented in this dissertation.

8.6.2 Dataset 2: Advanced Parameterized Generator

The previous dataset enabled us to conduct successful experiments and analysis of the relationship between game content, player behavior and affective states as well as being able to conduct a preliminary study of game adaptation and generating personalized content as will be discussed in later chapters. It became apparent, however, that the dataset had some limitations. The number of controllable features (and the number of configurations of these features that were tested) was too small to permit meaningful exploration of the search space and possibilities of finding interestingly new design parameter configurations. Also, one of the controllable features (direction switching) turned out to be relatively uninteresting to explore in the context of the current game. The levels used in the first data set took about a minute to play each, which we judged was overly long given that we wanted our model to apply to the aesthetics of the moment, in order to enable online adaptation. Finally, and most importantly, we wanted to

8.6. DATASETS

record more detailed information about both levels and gameplay in order to see if we could find a way to predict player experience even better—to squeeze more information out of the data, as it was. We therefore embarked on collecting a new dataset, with more content features and more players.

A new experiment was therefore designed to construct a new dataset. The advanced version of the parameterized generator has been adopted to generate the levels for this experiment. Two states (low and high) are set for each of the controllable parameters except for enemies placement which has been assigned three different states allowing more control over the difficulty and diversity of the generated levels as discussed in Section 5.3.1.2. The total number of pairwise combinations of these states is $2^5 * 3 = 96$ combinations. This leads to a total of $C_2^{96} = 4560$ different game pairs to be played and compared. The total number of combinations, however, can be reduced to 40 by analyzing the dependencies between these features and eliminating the combinations that contain dependent variables. For example, the number of boxes, B , and enemies placement, E_p , are dependent variables since when $B = 0$, enemies cannot be placed around boxes and therefore the combination that differentiates from other combinations along only these two dimensions can be eliminated. After reducing the number of combinations to 40, the total number of pairs to be played in order for all different combinations to be presented and compared becomes $C_2^{40} = 780$ pairs.

The 40 different variants of the content features have been used to construct 40 different levels. Other features of the levels have been given fixed values such that the number of bill blasters and piranha plants is fixed to one, the type of background = overground, the number of coins = 7, the number of coins hidden in boxes = half the total number of boxes and the number of stairs around the gaps = half the number of gaps. All generated levels have been checked before starting the data collection in a way that assures their compatibilities with the intent parameters assigned.

The game sessions presented to players have been constructed using a level width of 100 Infinite Mario Bros units (blocks), which take roughly 30 seconds to play, about one-third of the size usually employed when generating levels for the game in the previous dataset. The selection of this length was due to a compromise between a window size that is big enough to allow sufficient interaction

between the player and the game to trigger the examined states and a window which is small enough to set an acceptable frequency of an adaptation mechanism applied in real-time aiming at closing the affective loop of the game.

A crowd-sourcing experiment has been conducted to collect the data. A Java applet containing the game was created and placed on a web page, which was then advertised over social networks, mailing lists and blogs. The applet is connected to an online SQL database that is used to collect data about game content, player’s behavior and reported experience. The database initially contains all possible pairs marked as “unplayed”. Whenever a game session starts, the software connects to the database and asks for an unplayed pair to load. Once two levels are chosen from the database, they are loaded and the player is ready to play. The protocol presented in Section 8.1 has been followed. Whenever a pair of two games is completed and its questionnaire is answered, the pair is marked as “played”. The list of played pairs is reset if there are no more pairs available in the database to play (all pairs were marked as “played”).

Since our main goal from designing this experiment is to record as many data as possible about the game content and player behavior, complete games were logged, including the levels and what actions the players took at which time, enabling complete replays and permitting the extraction of direct and sequential features.

A total number of 780 players participated in this crowd-sourcing experiment. Participants’ age covers a range between 16 and 64 years (31.5% females) while their location includes Denmark (46.11%), Greece (8.9%), Ireland (1.48%), USA (3.34%), Holland (0.74%), Finland (1.36%), France (0.37%), Syria (0.25%), Sweden (0.37%), Korea (0.12%), Spain (0.25%) or unknown (36.71%).

Direct features as well as sequential patterns have been extracted from the data recorded. In the following sections, we describe the feature extraction process in details.

8.6.2.1 Direct Features

Direct features represent frequencies of content items or players’ behavior occurring throughout the full game session. These features provide quantitative, com-

pressed information about the different types of interaction between the player and the game and can be directly extracted from the gameplay sessions. The content features considered are the same six features imposed by the generators while constructing the levels (Section 5.3.1.2), and the same direct gameplay features recorded in the previous dataset, as presented in Table 8.1, were extracted.

8.6.2.2 Sequential Patterns

We investigate another form of indirectly representing game content and the gameplay interaction by means of sequences which allows including features that are based on ordering in space or time. The direct features, presented in the previous section, provide a quantitative measure about different types of game content and playing style. Alternatively, analyzing sequences of game content and players' behavior yields patterns that might be directly linked to player experience. For example, we would like to extract features that encapsulate whether a player performed a particular action before or after encountering a specific in-game situation.

Modeling players' experience based on features extracted from sequential information provides a promising alternative for models constructed based on direct feature extraction, and by fusing these two types of representations, we anticipate constructing more accurate models of players' experience than those constructed on one of these form of data representation at a time.

In the following, we describe different criteria for extracting sequences from game content, gameplay, and the interaction between the two. We present two sequence mining approaches and further discuss different setups that can be used for mining the extracted sequences.

Table 8.2 presents the different possible approaches that can be followed to generate different types of sequences. The columns represent the different orders and frequencies at which information is logged. The rows represent what type of data is logged each time an event occurs. We will be distinguishing the following orders/frequencies, while acknowledging that even more fine-grained distinctions are possible:

- t_{small} : time step. Information is logged at a constant rate (e.g. once per

Table 8.2: The different types of sequences that can be generated. Columns present the type of event to be recorded, while rows present when to record the event. The combinations marked with an X are the ones investigated in this dissertation

	t_s	Block	Gameplay Event
A_h (Player Behavior)	X		X
C (Content)		X	X
M (both)			X

second), regardless of what the player does. This yields a sequence with a length proportional to the time step chosen and the time taken by the player to complete the level.

- **Block:** Information is logged once per block in the level, independent of the time taken by the player to traverse the level. This yields a sequence with a length equal to the width of the level.
- **Gameplay event:** Information is logged each time the player changes the command issued (pressing/releasing a button or changing direction) or something else happens (e.g. Mario changes the mode or stomps on an enemy).

The information logged can be game content (C), player (gameplay) behavior (A_h) or both game content and player’s behavior (M).
















We will focus the discussion on the five sequence types marked with an X in Table 8.2. Although we only investigate a few sequence types of all those available, our sample provides a variety of options that cover different aspects of playing experience.

Once we know what to sample and when, the question remains how to turn this information into sequences using a low-cardinal alphabet. Below, we discuss how to do this for levels and for gameplay traces.

8.6.2.2.1 Sequential Content Features Sequences capturing different information about level geometry have been extracted by converting the content of

8.6. DATASETS

the levels into numbers representing different types of game items. Three different representations of game content have been investigated. The full list of events considered as well as their graphical representation is presented in Table 8.3.

- Platform structure, P : A sequence is generated by comparing the height of each block across the level with the height of the previous block and recording the following values: 0 if no difference found (); 1 if there is an increase in the platform height (); 2 if there is a decrease in the platform height (); and, 3 and 4 to mark the beginning () and the ending () of a gap, respectively. Figure 8.1.(a) presents part of a level and the corresponding platform structure sequence representation.
- Enemies placement, EP : A bit-string sequence that represents the initial placement of enemies along the level has been generated for each level. A boolean variable is used to represent the existence (0) or non-existence (1) of enemies.
- Enemy and item placement, I : The term items refers to the coins and the different types of boxes scattered around the level. The existence and non-existence states for enemies and items have been combined together resulting in four different possible values: 0, 1, 2 and 3 corresponding, respectively, to non-existence of either enemies or items, the existence of an enemy () , the existence of an item () , and the existence of an enemy and an item () . Figure 8.1.(b) illustrates an example level segment where the above-mentioned four states are presented.
- Content corresponding to gameplay events, C_g : We explored another method in which game content at the specific player position is recorded whenever the player performs an action or interacts with game items. In this case, different content events are used: increase in platform height, ; decrease in platform height, ; existence of an enemy, ; existence of a coin, block or brick,  ; existence of a coin, block or brick with an enemy, ; and the beginning, , and the end , of a gap.

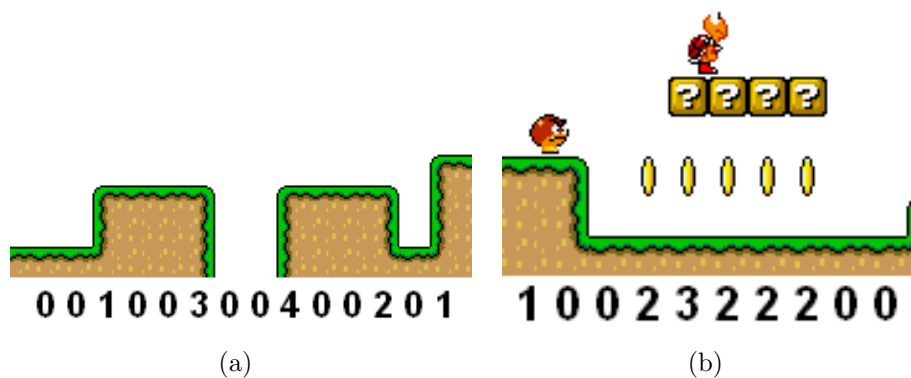


Figure 8.1: Snapshot from a level and the corresponding platform structure sequence representation, P (a), and enemies and items sequence representation, I (b).

Table 8.3: The different types of events considered when generating the sequences and their graphical representation.

Category	Graphical Representation	Description
Platform Structure		Flat platform
		Increase/decrease in the platform height
		The beginning/end of a gap
Enemies and Items		The existence of an enemy
		The existence of coin, block, or brick block
		The existence of enemy with a rewarding item
Gameplay	►, ◄, ▼	Moving right, left or duck
	↑	Jumping
	↑►, ↑◄	Jumping right or left
	R►, R◄	Running right or left
	R►↑, R◄↑	Running while jumping right or left
	S	Not pressing any key
	E _s	Stomping on an enemy
	U	Unleashing a koopa shell
	D	Changing Mario mode
W, L	Winning or losing the game	

8.6.2.2.2 Sequential Gameplay Features Sequences representing different players' behavior have been generated by recording key pressed/released events (*action event*) or interaction with items events. The action event might represent a simple action performed such as pressing an arrow key to move left or right; or

8.6. DATASETS

more complex player's behaviors that can be achieved by pressing a combination of keys at the same time (e.g. jumping over a big gap requires the player to press the run and jump keys together for a number of time steps). The following list describes the different events that have been considered.

- Pressing an arrow key to move right, left, or duck (\blacktriangleright , \blacktriangleleft , \blacktriangledown).
- Pressing the jump key, \uparrow .
- Pressing the jump key in combination with right or left key ($\uparrow\blacktriangleright$, $\uparrow\blacktriangleleft$).
- Pressing the run key in combination with right or left key ($R\blacktriangleright$, $R\blacktriangleleft$).
- Pressing the run and jump keys in combination with right or left key ($R\blacktriangleright\uparrow$, $R\blacktriangleleft\uparrow$).
- Not pressing any key, S .
- Winning the game, W .
- Losing the game, L .
- Killing an enemy by stomping, E_s .
- Unleashing a koopa shell, U .
- Changing Mario mode, D .

Figure 8.2 presents the graphical interpretation for most of the actions that can be performed and that are considered for the experiments presented in this dissertation.

We consider two time window t values for generating sequences: 0.5 sec ($A_{0.5}$) and 0.25 sec ($A_{0.25}$) meaning that an event will be registered every half or quarter of a second, respectively. We also consider sequences generated whenever the player switch the action, A . (Note that Infinite Mario is a fast-paced game in which the player could in theory perform an action every 1/24 sec).

The purpose of recording these events is that players' behavior and playing style can be analyzed by looking at events generated by each player and how frequent each of these events occurs. Generating a sequence combining these events

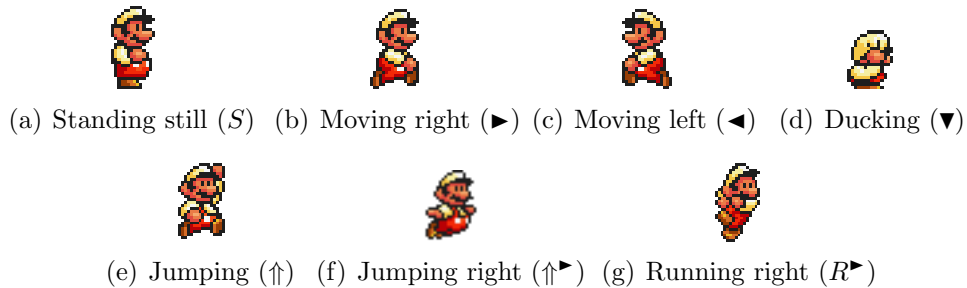


Figure 8.2: Graphical representation of the different actions that can be performed by the player.

in a timely manner provides an in-depth insight about more complex behavior patterns that might have an impact on players' experience.

The resulted sequences of players' behavior have a wide variety both in terms of length and structure, which reflects the diversity of players' playing style and complicate any sequence mining algorithm that can be applied to extract useful information. This diversity is reflected on the normalized compression distance (NCD) measure (Li et al. [2004]) that has been applied to test for structural similarity between the sequences. The results of applying this function on each pair of the action sequences showed a high dissimilarity between the sequences (NCD>0.6 in 71.32% of the cases).

8.6.2.2.3 Fused Sequential Features of both Game Content and Gameplay Data Game content and players' behavior events have been fused together to generate bimodal sequences (M). Events from the two modalities have been extracted with their corresponding time stamp and then logged in temporal order. The generated event contains information about the game content at the specific position in the game where the gameplay event occurred (which is one of the events mentioned in Section 8.6.2.2.1 or none if no content event from the list happens to occur at this specific position) along with the type of the gameplay event.

8.6.2.3 Mining Sequential Features

Sequence mining techniques have been applied to extract useful information from the different types of the sequences generated. Two algorithms for frequent item-set mining have been implemented to find frequent sequence patterns within the dataset of sequences: Apriori and GSP. The Apriori (Agrawal and Srikant [1994]) algorithm has been used to mine single-dimensional sequences that represent game content independently of player behavior, namely, platform structure (P), enemy placement (EP) and enemy and item placement (I). Mining sequences across multiple time series of data — content corresponding to gameplay events (C_g), player behavior (A) and multimodal sequences (M) — have been achieved via the Generalized Sequential Pattern (GSP) algorithm (Srikant and Agrawal [1996]). An explanation of the two algorithms used and a detailed list of sequence mining definitions are presented in Chapter 3.

In the following sections we give a brief description of the way these two algorithms have been used and their experimental setups.

8.6.2.3.1 Apriori Game content for all levels used to collect data from players has been converted into numbers representing different types of content events as described in Section 8.6.2.2.1. Different subsequence lengths and minimum support thresholds values have been explored. Throughout this dissertation, a minimum support threshold of half the size of the full dataset is usually employed unless otherwise mentioned, meaning that each subsequence should occur at least in half of the levels to be considered frequent. Table 8.4 presents the number of frequent subsequences of length 3 that have been found in the 40 levels generated for dataset 2 for the three types of content sequences using a $min_{sup} = 20$. Table 8.5 presents a subset of the frequent subsequences of length three and the number of occurrences of each of them for one example level.

8.6.2.3.2 GSP The GSP algorithm is used for mining sequences that rely on players' behavior (C_g, A_t, A) since it allows more generalized frequent patterns to be found by exploring different max_{gap} , and it is also used of mining multimodal sequences (M) as, by using max_{win} , we can discover simultaneous events from two different modalities (Martinez and Yannakakis [2011]).

8. DATA COLLECTION AND FEATURE EXTRACTION

Table 8.4: The number of frequent subsequences of length three extracted from the levels used in dataset 2 using a minimum support of 20.

Sequence	# frequent subsequences
<i>P</i>	35
<i>EP</i>	7
<i>I</i>	12

Table 8.5: A subset of the frequent subsequences of length three of *D* and the corresponding occurrences of each of them in one example level.

Frequent subsequences	#of occurrence
000, 00 0, 0000, 🏀00, 0000, 0000, 0000, 🏀🏀, 🏀🏀	80,0,1,2,2,2,3,0,0

Different min_{sup} values have also been explored to obtain a reasonable trade-off between considering patterns that are generalized over all players and more specific patterns. For the experiments presented in this dissertation, and because of the wide diversity of player’s behavior patterns, we use a min_{sup} value that enforces sequence patterns to occur in at least around 30% of the samples to be considered frequent.

The parameter max_{win} defines the threshold under which events from two different modalities can be considered as simultaneous events or events from one modality are considered to be happening in a very short interval in space or time. Throughout this dissertation, max_{win} is sec. The value for this parameter has been chosen as a tradeoff between a small window size that does not consider simultaneous events, and a window size that processes clearly asynchronous events from two modalities as events happening in a very small interval. For instance, we wanted a window size that allows the extraction of the pattern of jumping when reaching a gap (🏀, 🏀). These two events should be considered as simultaneous events even though they might not have occurred in the same time-stamp. For the rest of the dissertation we will use parentheses to group simultaneous events. For instance, the two patterns (🏀, 🏀) (🏀) and (🏀, 🏀, 🏀) identify gaps of different width. The first pattern points out to a gap of big width due to the separation

8.6. DATASETS

between the beginning and end of the gap in two items of the pattern. On the other hand, the grouping of the beginning and the end of the gap in one item in the second pattern indicates a gap of small width.

The max_{gap} parameter is used to set up the time gap between two events to be considered as belonging to the same pattern. This parameter has a great impact to the number of frequent patterns that can be extracted. By assigning a large value to this parameter, we allow more generalized patterns to be taken into account and as a consequence, a large number of sequences will be counted as sup_{count} . For example, moving right is the most frequent action the player usually performs, when assigning a large value to max_{gap} the chance of choosing patterns that contain the moving right action as an event increases. The analysis of investigating the frequent patterns containing the moving right action using different max_{gap} values showed that twice as many sequences as the ones supporting the pattern ($\blacktriangleright, \blacktriangleright, \blacktriangleright, \blacktriangleright, \blacktriangleright, \blacktriangleright$) will be found if we increase the max_{gap} parameters from 1 second to 3 seconds.

Another drawback for using large max_{gap} values is that it allows considering less informative patterns. Large max_{gap} values means skipping more specific patterns and focusing on the more generalized ones as can be seen from the previous example. The frequent patterns found consisting of six moving right events contain very little information about player’s behavior since the number of players who press the right button six times in a row is small, and the pattern doesn’t provide any information about the events happening in between.

Thus, the type of patterns we would like to investigate and the frequency of each event play very important roles when setting up the value for this parameter. Table 8.6 presents the number of frequent subsequences of different length found in 1560 multimodal sequences extracted from the data collected in dataset 2 using different max_{gap} settings. Note that the number of extracted frequent patterns increases as the max_{gap} value increases due to the fact that larger max_{gap} values mean more generalized extracted patterns which as a result will be supported by a larger number of sequences.

Furthermore, correctly tuning this parameter has a large impact on the interpretation of the resulted patterns, especially when mining multimodal sequences. For instance, if we use $max_{gap} = 3sec$, the pattern ($\uparrow, \text{👾}, \blacktriangleright$) can be supported

Table 8.6: Number of frequent sequential patterns found of different length for a number of max_{gap} values and a minimum support of 500 from 1560 multimodal sequences of game content and players’ behavior extracted from dataset 2.

Sequence length	max_{gap}		
	1 sec	2 sec	3 sec
1	18	18	18
3	939	1697	2157
4	1982	6768	10577
5	2957	18922	36186
6	462	44609	>103053
7	38	91416	NaN
8	0	NaN	NaN

by any sequence in which the player jumps, moves right and encounters an enemy within a 3 seconds interval (note that within this interval, the player might encounter more than one enemy or a gap between the jumping and moving right events which makes this pattern somehow misleading).

The experiments conducted for tuning the value of this parameter showed that a max_{gap} of 1 sec provides a good tradeoff between the number of patterns extracted and their expressiveness value.

The choice of the length of patterns to be extracted also plays an important role when mining sequential data. Too small length leads to patterns with very little information and high number of occurrences, while very long patterns usually carry information about more specific behaviors that capture the playing style of few players. A compromise has to be made when tuning the value of this parameter for efficient pattern extraction. An experiment has been conducted on the data collected in dataset 2, Table 8.7 presents the different types of sequences and the number of frequent subsequence found for the number of different sequence generation methods presented in Table 8.2. As can be seen from the table, the number of extracted subsequences is quite large for sequences containing information about players’ behavior (A), and the search space for automatic feature selection increases substantially when fusing content and gameplay events for generating multimodal sequences (M); more than 2000 subsequences of length six have been extracted from the players’ behavior and multimodal sequences.

8.6. DATASETS

In order to lower the dimensionality of the feature space and the computational cost of searching for relevant features we chose to use only frequent sequences of length three for the experiments presented in the following chapters.

Table 8.7: Number of frequent sequential patterns found in the sequences extracted from dataset 2 for different sequence length values across different types of sequences (min_{sup} is 500 and max_{gap} is 1 sec). The columns stand for: content corresponding to gameplay events (C_g), gameplay behavior (A), gameplay behavior registered every 0.5 sec ($A_{0.5}$) and every 0.25 sec ($A_{0.25}$) and biltimodal sequences of game content and player behavior (M).

Sequence length	C_g	A	$A_{0.5}$	$A_{0.25}$	M
1	7	8	2	8	18
2	27	64	2	57	205
3	25	310	0	69	939
4	23	939	0	741	1982
5	29	2065	0	1810	2957
6	0	2636	0	2547	2806
7	0	1403	0	1400	1402
8	0	115	0	112	112
9	0	0	0	0	0

Note that the number of frequent subsequences of length three found for A and M is still large (310 and 939, respectively). In an effort to further reduce this space, we perform a patterns-pruning step in which patterns from M that do not contain both content and gameplay events were eliminated. This constraint substantially reduces the number of frequent patterns to 437.

8.6.3 Dataset 3: Behavioral and Visual Cues

The two datasets presented previously rely on subjective data collected from players' self reports. As discussed in Section 8.5 there are a number of limitations embedded in this approach. Therefore, a new experiment has been conducted to design the third dataset. The dataset introduces a large data corpus derived from 58 participants that play Infinite Mario Bros.

An extensive corpus of visual and behavioral data is used for the analysis of the cognitive states and behavior of the player; behavioral and visual cues are



Figure 8.3: Typical instances of players from Denmark (a,b) and Greece (c,d).

fused and used in the later experiments for the prediction of player experience, producing concepts related to the gaming paradigm by relating player states to particular in-game events.

The protocol proposed in Section 8.1 has been used to assess the players' affective states during play. Fifty-eight subjects participated (28 male; player age varied between 22 and 48 years). Participants were seated in front of a computer screen for video recording. Experiments were carried out in Denmark and Greece. Lighting conditions were typical of an office environment, and for capturing players' visual behavior, a High Definition camera was used (Figure 8.3).

8.6. DATASETS

Each participant played from two to five pairs of games on average, resulting to a total of 190 game pairs (more than 6 hours of recordings). The game sessions presented to players have been constructed using the same level generator employed to construct the previous dataset (the advanced parameterized generator with a level width of 100 blocks).

Features extracted from player gameplay behavior and game levels, as well as player visual characteristics that have been utilized as potential indicators of reported affect expressed as pairwise preferences between different game sessions. Our ultimate goal is to be able to construct accurate estimators of player experience derived from game content and player behavior, and by fusing subjective and objective measures of players' affective states, we aim to better model player experience.

Along with direct features of game content and gameplay (the same ones extracted in dataset 2), we examine a series of head movement features as described in the next section.

8.6.3.1 Head Movement Features

A number of head movement features have been investigated to better estimate players' affect while interacting with the game. In particular, we track players' head motion through head horizontal and vertical (yaw and pitch) rotational movements.

8.6.3.1.1 Mean Head Movement Features We considered the first derivative of the norm of the head pose vector (Asteriadis et al. [2009]) and use the average (*Avg*) of its absolute values throughout whole game sessions. A series of further head movement features (Caridakis et al. [2010]) have also been considered in order to elicit emotional information of the player during each game session. More specifically, we considered:

- Overall Activation (*OA*), which is the sum of speeds for each rotational movement, separately.
- Temporal Expressivity (*TE*) parameter.

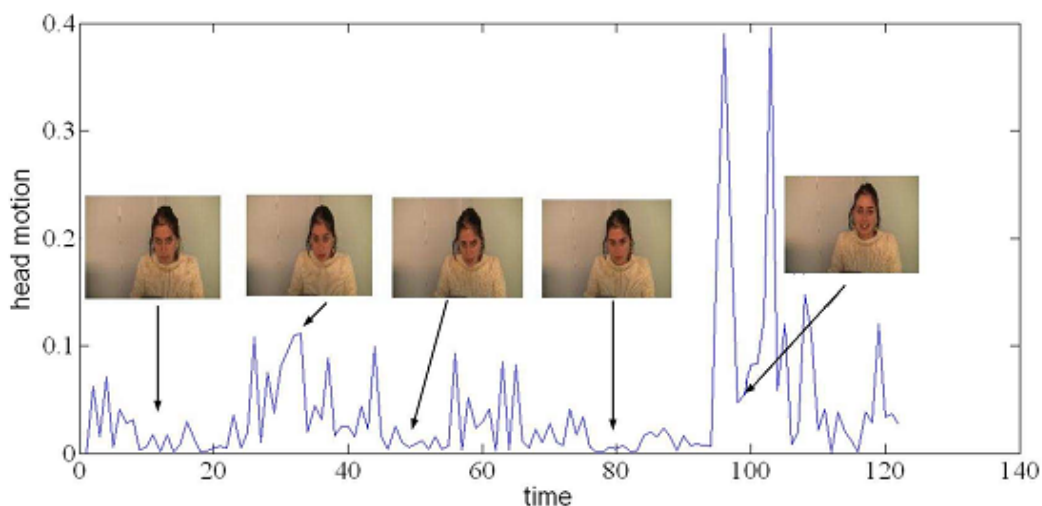


Figure 8.4: Typical Head Expressivity of player reacting to certain game events.

- Spatial Extent (SE) parameter.
- Energy Expressivity parameter (Power) of head movement (PO).
- Fluidity of head movement (FL).

The detailed list of extracted head movement features is presented in Table 8.8. The method proposed by Caridakis et al. [2010] has been used to extract these features. In addition to the above features, the median values of horizontal, $M_{horizontal}$, and vertical, $M_{vertical}$ rotations, as well as medians of head rotation norms M_{norm} are also considered.

8.6.3.1.2 Visual Reaction features As players' expressivity appears to increase during certain events as can be seen from Figure 8.4, we also considered the above features for certain gameplay events as described below:

- When the player loses a life.
- When the player kills an enemy by stomping on it.
- When the player starts or ends a one of the following actions: jump, duck, run, and move left or right.

8.6. DATASETS

Table 8.8: Head Movement Features: Mean head movement features extracted throughout whole sessions and visual reaction features during gameplay events. The gameplay events considered for extracting these features include: losing, stomping, (start/end) jumping, ducking, running left, running right and interacting with items.

Category	Feature	Description
Head Movement Features throughout whole sessions		
Mean Head Movement	Avg	Absolute first order derivative of Head Pose Vector
	OA	Overall Activation
	SE	Spatial Extent
	TE	Temporal Expressivity parameter
	PO	Energy Expressivity parameter
	FL	Fluidity
	$M_{horizontal}$	Median value for horizontal head rotation
	$M_{vertical}$	Median value for vertical head rotation
Head Movement Features during gameplay events		
Visual Reaction Features	Avg_a	Absolute first order derivative of Head Pose Vector when the gameplay event, a occur
	OA_a	Overall Activation when the gameplay event, a occur
	SE_a	Spatial Extent when the gameplay event, a occur
	TE_a	Temporal Expressivity parameter when the gameplay event, a occur
	PO_a	Energy Expressivity parameter when the gameplay event, a occur
	FL_a	Fluidity when the gameplay event, a occur
	M_a	Median value for head rotation norm when the gameplay event, a occur

- When the player interacts with a game item.

These features are calculated for periods of 10 frames before and after the corresponding events. Subsequently, their mean values were compared to the corresponding average values (by calculating fractions) during normal gameplay for each game session separately. A detailed list of the features used can be seen in Table 8.8. Figure 8.5 presents two instances of visual reactions to game events.



Figure 8.5: Visual reactions to game events.

8.7 Summary

Collecting datasets of game content and players' behavior can help us better understand the in-game interaction and provide a solid ground for constructing quantitative measures of player experience that enable a data-driven analysis of the interaction between the player and the game.

Based on the types of data collected, different analysis directions can be followed and variant methods can be performed to squeeze useful information and findings. For the work presented in this dissertation, the attention is given to constructing accurate estimators of player experience and the use of these estimators to personalize gameplay experience. To this end, four datasets have been collected each with different setup, number of participants and variant types of features capturing different aspects of game content and playing behavior. The wealth of the data collected in these datasets enables an in-depth analysis of the interaction between the player and the game by utilizing different types of features that highlight important aspects of the in-game experience across multiple modalities of player input.

Large sets of features have been extracted from each dataset emphasizing the need of applying feature selection methods in an attempt to ease the analysis of the relationship between the content of the game, the players' playing style and

8.7. SUMMARY

the reported experience. Hence, the framework proposed for player experience modeling employs a sequential forward feature selection method to extract subsets of features that have predictive capabilities with respect to reported player affects. Based on the selected feature subsets, models of player experience will be constructed and used for a thorough analysis of the factors that contribute to player experience in platform games.

9

Player Experience Modeling: Experiments

The methodology proposed in the previous chapter (Chapter 7) for modeling player experience has been followed to construct models based on the three datasets presented in Chapter 8. This chapter provides a thorough analysis conducted for testing simple and more complex relationships between the features extracted and the three reported states of player experience. Different experiments have been conducted on the different datasets. We further investigate the generality of the proposed approach and the impact of the types of the features by comparing some of the models constructed in terms of the models' performance and the features selected.

9.1 Correlation Analysis

The correlation analysis provides the simplest mean of investigating the linear relationship between the features and the reported players' preferences. Such an analysis can help us easily interpret the impact of game content and players' behavior on how players reported their experience and what they felt about the game.

In the following sections, we analyze the linear relationship between the different types of features selected in dataset 2 and reported preferences. The reason

for focusing the discussion on this dataset is because it is a generalization of dataset 1 in terms of the content generator used and in terms of the features collected. This dataset provides a rich set of features represented in direct and sequential forms permitting in-depth analysis.

9.1.1 Dataset 2: Advanced Parameterized Generator

A large set of features has been extracted from the game content and gameplay data collected in dataset 2. The data collected was rich enough allowing complete replays and therefore permitting the extraction of sequential features as well as direct features. The full description of the data and the different types of features extracted are presented in Chapter 8.

We performed an analysis for exploring statistically significant correlations ($p - value < 1\%$) between player's expressed preferences and the different types of the direct and sequential features extracted from 780 game pairs. Correlation coefficients are obtained by following the method proposed by Pedersen et al. [2009]. According to this method, correlation coefficients are calculated through:

$$c(z) = \sum_{i=0}^{N_p} \{z_i/N_p\} \quad (9.1)$$

where N_p is the total number of game pairs where players expressed a clear preference ($game_A > game_B$ or $game_B > game_A$) for one of the two games and $z_i = 1$, if the player preferred the game with the larger value of the examined feature and $z_i = -1$, if the player preferred the other game in the game pair i .

The top five significantly correlated features across all experience states investigated are presented in Table 9.1 (refer to Chapter 8 for more information about the feature extraction methods followed and the full list of features extracted). The significance test results showed that nineteen direct features are significantly correlated with engagement with some of them also strongly correlated with frustration and challenge while 21 features are significantly correlated with frustration, and 17 features with challenge.

The features that are strongly correlated with engagement and not with challenge are mostly related to the interaction between the player and boxes (mainly

powerups); These features point to the task of searching for powerups, in which the player has to destroy blocks looking for powerups that as a result changes Mario's mode, as being particularly engaging.

The avatar death feature (signifying that Mario loses a life) is the most significantly correlated with both frustration and challenge indicating a strong relationship between death and these two player experience states.

Regarding changes in platform height patterns, P , only the features presented in Table 9.1 for engagement and frustration are significantly correlated with engagement and frustration while 15 features are strongly correlated with challenge. Seven and 15 out of the features that correlate best with frustration and challenge, respectively, relate to the presence of a gap while engagement is significantly correlated with only four features that indicate a gap. It is interesting to note that despite the relatively small patterns' length (three) almost all features presented for the three emotional states require two or three gameplay actions to be performed.

Ten out of the 12 features from I (item and enemy placement) are significantly correlated with engagement while only three and two features correlate significantly with frustration and challenge, respectively. A first observation is that it is obviously much easier to predict engagement from I than to predict challenge and frustration due to many more features significantly correlated to engagement, and the correlations are stronger. Most features that correlate with engagement point to the placement of items and enemies. This is not the same for frustration which demonstrates less significant effects and the majority of those that do focus on the existence of an enemy; features that correlate with challenge highlight the importance of the relative placement of items and enemies in the challenged perceived.

Large subsets of features of players' actions (A) are significantly correlated with engagement, frustration and challenge (99, 72 and 74, respectively). All features that are highly correlated to frustration are also correlated to challenge. It is worth mentioning that the features that correlate the most with engagement are also significantly correlated with frustration and challenge but at different significance levels. It appears that the number of jumps the player performs plays an important role in predicting engagement as it appears in all top-5 action patterns


Table 9.1: Top five statistically significant correlation coefficients between reported engagement, frustration and challenge and extracted sequential and direct features. The sign before the features indicates positive (+) or negative (-) correlation. The columns stand for: platform structure (P), enemy and item placement (I), gameplay behavior (A), gameplay behavior registered every 0.25 sec ($A_{0.25sec}$), content corresponding to gameplay events (C_g) and multimodal sequences of game content and player behavior (M).

Direct	Sequential					
	P	I	A	$A_{0.25sec}$	C_g	M
	Engagement					
$+t_{comp}$		-	$+(S)(\blacktriangle)(\uparrow)$	$-(\blacktriangle)(\blacktriangle)(S)$	$+(0,0,0)$	$+(\img alt="green square icon"),(\img alt="black triangle icon"),(\img alt="up arrow icon"))$
$+t_{lastLife}$		+	$-(R^{\blacktriangleright})(\blacktriangle)(\blacktriangle)$	$-(\blacktriangle)(\uparrow)(S)$	$+(0,0,0)$	$+(\img alt="green square icon"),(\img alt="black triangle icon"),(\img alt="up arrow icon"))$
$+n_{powerups}$		+000	$-(\uparrow)(S)(S)$	$-(\blacktriangle, \uparrow)(\uparrow)$	$+(0,0,0)$	$+(\img alt="green square icon"),(\img alt="black triangle icon"),(\img alt="up arrow icon"))$
$+n_{state}$		+	$-(R^{\blacktriangleright})(\blacktriangle)(R^{\blacktriangleright})$	$-(\blacktriangle, \uparrow)(S)$	$+(\img alt="green square icon"),(\img alt="black triangle icon"))$	$+(\img alt="green square icon"),(\img alt="black triangle icon"),(\img alt="up arrow icon"))$
$+N_w$		+	$-(\blacktriangle)(\uparrow)(\blacktriangle)$	$+(\blacktriangle, \uparrow, S)$	$+(\img alt="green square icon"),(\img alt="black triangle icon"))$	$+(\img alt="green square icon"),(\img alt="black triangle icon"),(\img alt="up arrow icon"))$
	Frustration					
$+d_{cause}$		+	$+(S)(\blacktriangle)(\uparrow)$	$+(\blacktriangle, \blacktriangle)(S)$	$-(\img alt="green square icon"),(\img alt="black triangle icon"))$	$+(\img alt="black triangle icon"),(\img alt="up arrow icon"),(S))$
$-n_{coin}$	+	+	$-(R^{\blacktriangleright})(\blacktriangle)(\blacktriangle)$	$-(\blacktriangle)(\uparrow)(S)$	$-(\img alt="green square icon"),(\img alt="black triangle icon"),(\img alt="up arrow icon"))$	$+(\img alt="black triangle icon"),(\img alt="up arrow icon"),(S))$
$+d_{num}$	+	+	$-(\uparrow)(S)(S)$	$-(\blacktriangle, S)(\uparrow)$	$-(\img alt="green square icon"),(\img alt="black triangle icon"),(\img alt="up arrow icon"))$	$+(\img alt="black triangle icon"),(\img alt="up arrow icon"),(S))$
$-n_{jump}$	+	+	$-(R^{\blacktriangleright})(\blacktriangle)(R^{\blacktriangleright})$	$+(\blacktriangle)(\blacktriangle)(S)$	$+(\img alt="green square icon"),(\img alt="black triangle icon"))$	$+(\img alt="black triangle icon"),(\img alt="up arrow icon"),(S))$
$-t_{lastLife}$	+	+	$-(\blacktriangle)(\uparrow)(\blacktriangle)$	$+(\blacktriangle)(\uparrow)(S)$	$+(\img alt="green square icon"),(\img alt="black triangle icon"))$	$+(\img alt="black triangle icon"),(\img alt="up arrow icon"),(S))$
	Challenge					
$+d_{num}$		+	$-(S)(\blacktriangle)(\uparrow)$	$-(\blacktriangle)(\blacktriangle)(\uparrow)$	$+(\img alt="green square icon"),(\img alt="black triangle icon"))$	$-(S)(\blacktriangle)(S))$
$+d_{cause}$	+	+	$-(R^{\blacktriangleright})(\blacktriangle)(\blacktriangle)$	$+(S)(\uparrow, S)$	$+(\img alt="green square icon"),(\img alt="black triangle icon"),(\img alt="up arrow icon"))$	$-(0, \blacktriangle, R^{\blacktriangleright})$
$-k_{flower}$		+	$-(\uparrow)(S)(S)$	$-(S, S)(S)$	$+(\img alt="green square icon"),(\img alt="black triangle icon"))$	$-(\blacktriangle)(\img alt="black triangle icon"),(S))$
$-t_{right}$	+	+	$-(R^{\blacktriangleright})(\blacktriangle)(R^{\blacktriangleright})$	$-(\blacktriangle)(\blacktriangle, \blacktriangle)$	$+(\img alt="green square icon"),(\img alt="black triangle icon"))$	$-(\blacktriangle)(\img alt="black triangle icon"),(\img alt="black triangle icon"))$
$-n_{state}$	+	+	$-(\blacktriangle)(\uparrow)(\blacktriangle)$	$-(\blacktriangle, \uparrow)(\uparrow)$	$+(\img alt="green square icon"),(\img alt="black triangle icon"))$	$-(\blacktriangle)(\img alt="black triangle icon"),(\img alt="up arrow icon"))$

combined in most of the cases with moving right and pressing the speed button. This can also explain the significant correlation found between engagement and sequences of I that contains items which mostly require jumping to be collected and enemies which require a jump to be killed or overcome.

While jumping and moving right are the most important actions for predicting engagement, standing still, S (supposedly thinking about how to overcome the next obstacle) is the most frequent action in the subset of features correlated with frustration and challenge

Nine out of the 25 features of C_g are significantly correlated with engagement, while only three features are strongly correlated with challenge and frustration. It is worth noticing that all the features that correlate with challenge contain the same items and differ only in the placement of parentheses (the same applies for frustration). This indicates that the existence (or non-existence) of a sequence of certain content items is more important for the experienced frustration and challenge than its relative placement.

The three correlated C_g features with frustration linked to the existence of gaps and the placement of parentheses within each pattern reflect the width of a gap (a gap beginning and ending within the same item indicates a small gap width since the parentheses enclose events happening within a very short time). The fact that the significant patterns contain  in combination with a gap points out to stairs surrounding the gap or changes in platform height within a very close distance to the gap which add to the difficulty of jumping and as a result, on the reported frustration. Somewhat surprisingly, patterns including the presence of gaps are not correlated with challenge. This suggests a possible nonlinear relationship.

Large subsets of multimodal features are strongly correlated with engagement, frustration and challenge (232, 95 and 119, respectively). While most features correlated with frustration are also strongly correlated with challenge, the most significantly correlated features with engagement, are not strongly correlated with either frustration or challenge. Patterns correlated with engagement draw a picture of most players enjoying running in a platform with changing height that requires jumping. From the patterns correlated with frustration, it seems that frustrated players spend more time standing still, less time running through the

level (this can also be seen in A and $A_{0.25}$ patterns where the standing still and moving right — without the speed button pressed — are the most dominant actions).

Although many features of different forms has been analyzed and interesting findings have been obtained, the correlations calculated and analyzed above provide basic analysis with linear relationships between the extracted features and reported emotions. These relationships are most likely more complex than those that can be captured by linear models. In the following sections we analyze the nonlinear relationships found using players' experience models.

9.2 Nonlinear Relationships

The analysis of the nonlinear relationships between the extracted features and the investigated experience states has been obtained through modeling player experience via neuroevolutionary preference learning. The methodology proposed in Chapter 7 for selecting the relevant features for predicting player experience and for constructing the player experience models is followed. In the following sections we present the models constructed for each dataset. We further analyze the features selected and the models constructed.

9.2.1 Dataset 1: Basic Parameterized Generator

Dataset 1 contains 654 game pairs (1308 game sessions) played by 327 players. In order to construct the player experience models using neuroevolutionary preference learning, the collected data has been preprocessed to remove the pairs with unclear preferences, i.e. those pairs where both games are equally preferred or unpreferred. After this step, 458, 463, and 463 pairs remain for fun, challenge, and frustration, respectively.

A large set of direct features was extracted during gameplay (Table 8.1). Modeling player experience has been done in two steps: First, the relevant subset of features have been extracted using sequential forward feature selection (SFS) and, second, the topology of the model is optimized following the framework presented in Chapter 7.

All features values are normalized to $[0,1]$ using standard max-min normalization. After normalization, all features are used as inputs to sequential forward selection using SLPs (see Chapter 7 for more information about SFS and SLP) and the feature subsets that resulted in the best prediction accuracies have been chosen.

The set of features that yields the highest prediction accuracy differs for each experience state investigated. The selected feature subset for fun consists of four features: time needed to complete the level, t_{comp} , time spent in Super mode, t_{super} , time spent running, t_{run} , and the number of opponents died from unleashed shells, $k_{unleash}$. For frustration, a larger set of seven features were selected, consisting of: time needed to complete the level, t_{comp} , total number of bill blasters in the level, BL , number of power blocks destroyed, $n_{powerups}$, number of times the player shifted mode, n_{mode} , average width of gaps, \bar{G}_w , number of coin blocks destroyed, n_{block} , and number of jumps executed, n_{jump} . Challenge can be predicted using a set of six features: number of collected coins, n_{coin} , time needed to complete the level, t_{comp} , average width of gaps, \bar{G}_w , number of times the player shoots a fire ball, n_{shoot} , total number of bill blasters in the level, BL , and the cause of the last avatar death, d_{cause} .

Since our ultimate aim is to automatically generate game content that is tailored to players' experience in real-time, we need to be able to predict emotions, at least partly, from controllable features. For this purpose, after the subsets of features have been selected, all remaining controllable features that are not already included in the selected feature subsets are forced into the input of MLP models and a model optimization step has been performed to select the MLP topology that gives the highest prediction accuracy. The accuracies obtained are presented in Table 9.2.

Using direct features only, it appears that frustration is the easier to predict since the model constructed to predict reported frustration yields the highest accuracy, although it has the largest subset of features and a two-layers network. Challenge comes next in terms of prediction accuracy (74.66%) and the size of the selected feature set with three features overlapping with the ones selected for predicting reported frustration. The best network topology for predicting reported challenge, however, consists of one layer only with three neurons. Fun

9.2. NONLINEAR RELATIONSHIPS

Table 9.2: The subset of direct features selected for predicting players’ reported experience of fun, frustration and challenge and the corresponding best models’ topologies and performance. SLP_s is the model performance on the selected subset of features and MLP_c is the performance after forcing the controllable features as inputs to the models. The topologies are presented in the form: number of inputs (including the content features)-number of neurons in the first hidden layer-number of neurons in the second hidden layer.

	<i>Fun</i>	<i>Frustration</i>	<i>Challenge</i>
SFS_{slp}	t_{comp}	t_{comp}	n_{coin}
	t_{super}	BL	t_{comp}
	t_{run}	$n_{powerups}$	n_{shoot}
	$k_{unleash}$	n_{mode}	BL
		n_{block}	d_{cause}
		\bar{G}_w	\bar{G}_w
n_{jump}			
$MLP_{topology}$	8-10-0	10-4-2	9-3-0
SLP_s	64%	84.66%	70%
MLP_c	69.66%	89.33%	74.66%

is the hardest to predict from direct features with an average prediction accuracy of 69.66% and the smallest set of selected features (only four features are selected for predicting reported fun). The network selected for predicting reported fun is of moderate size of one hidden layer of ten neurons.

9.2.2 Dataset 2: Advanced Parametrized Generator

A total of 780 game pairs constitute dataset 2. This dataset contains richer information about game content and players’ behavior than those extracted in dataset 1. Along with gameplay direct features used for dataset 1 in the previous experiment, this dataset includes detailed information about level content as well as each action the player performed. Due to its information richness, different types of sequential patterns of content and behavior have been extracted. In order to use sequential features as predictors of players’ experience, sequence

mining techniques have been used to extract frequent sequential patterns from the dataset of all content and gameplay sequences as discussed in Chapter 8. The number of occurrences of each frequent pattern in each game session is then used as input for player experience modeling.

All direct features and the number of occurrences of all frequent sequential patterns extracted are uniformly normalized to $[0,1]$ using standard max-min normalization. After normalization, these values are used as inputs for feature selection and ANN model optimization. The two-phase feature selection method, followed by the model optimization step presented in Chapter 7, have been adopted to construct player experience models. We investigate three types of features as inputs to the experience models; direct features, a number of variations of sequential patterns and we further explore a fusion of direct and sequential features as inputs. The fused features differ from multimodal sequential features in that these features are not fused on the extraction level, such as in the method followed for generating multimodal sequences, but rather on the selection level. The set of features in the fused feature category includes direct features of game content and gameplay, sequential content features corresponding to players' actions, C_g and sequential patterns of players' behavior, A . The selection of these categories is done in order to cover a wide variety of features that capture the variation in level structure and the diversity in players' behavior. Table 9.3 presents the different types of features selected for reported engagement, frustration and challenge.

Note that to design a level generation mechanisms that is driven by the player experience models we construct here, all remaining controllable features that are not selected in the feature selection process should be forced into the input of the MLPs. The MLP performance and topologies of the best MLPs (for both direct, various types of sequential features as well as the fused features) before and after forcing the content features in the inputs are presented in Table 9.4.







Using MLPs with the selected direct features and the remaining controllable features, we were able to predict engagement, frustration and challenge with relatively high accuracy (see Table 9.4). In the following sections, we analyze the results obtained from feature selection and model optimization for the three emotional states under investigation.

9.2.2.1 Engagement

Using different patterns of content and/or gameplay to construct player experience models resulted in models that vary in topology and performance. The best-performing model for predicting engagement has been constructed using selected patterns of players' gameplay taken every 0.25 seconds and achieved a performance that is significantly better than all other models (83.8%) followed by the model constructed on fused sequential and direct features (72.53%) with no significant difference from the model constructed on sequential features extracted from items and enemies placement, I , and models of direct features. It is interesting to note that this model ($A_{0.25}$) outperforms other models after including the direct controllable features in the inputs. Without including the controllable features, the model constructed on fused features outperforms the ones constructed on sequential features with no significant difference from the model constructed on direct features and the ones constructed on patterns extracted from players' gameplay, A .

The subset of direct features for predicting engagement (see Table 9.3) consists of the total time spent playing the game, the time spent doing different activities (running, jumping, in Super mode and in Small mode), the number of coins collected, the number of blocks destroyed (which in part relates to the number of collected coins since player smashes blocks to collect hidden coins and it also relates to the time spent in Super/Small mode), the number of times the jump button is pressed (which relates to the time spent jumping), the cause of death, and the controllable feature that defines the number of goombas and koopas scattered around the level.

Two of the directed features selected appear to be dominant in the selected sequential features of players' actions, $A_{0.25}$, more specially, running right and jumping. The two selected patterns ($\blacktriangleright, \uparrow\blacktriangleright, S$) and (\blacktriangleright)(\uparrow, S) point out to the existence of a content event that causes jumping and standing still behaviors. This can be better explained by looking at the selected content patterns that relate to gameplay events, C_g . By investigating the subset of selected features from these two types together, simple jumping actions, \uparrow , can be explained by changes in platform height and placement of items; moving right followed by jumping and

standing still patterns $((\blacktriangleright, \uparrow, S)$ and $(\blacktriangleright)(\uparrow, S))$ mostly relate to the behavior of overcoming enemies (, , ); the more complex navigation patterns that has been selected, such as $(R^\blacktriangleright, R^\blacktriangleright, R^\blacktriangleright\uparrow)$ that defines the behavior of pressing a combination of buttons at the same time within a very small window time, suggest the existence of a gap that requires speeding up followed by jumping while the moving right and the speed button are still pressed (, , ). Note that the two patterns $((\blacktriangleright, \uparrow, S)$ and $(\blacktriangleright)(\uparrow, S))$ can also be the result of overcoming a gap, which in that case reflect a beginner player playing style. On the contrary the pattern $(R^\blacktriangleright, R^\blacktriangleright, R^\blacktriangleright\uparrow)$ captures a more advanced playing behavior. Since the methodology proposed constructs average models in the sense that the models are trained on a composite of subjective preferences of several subjects, patterns that capture the playing style of beginner and expert players can be selected and presented as inputs to the models.

The subset of fused features includes all direct features along with two other sequential patterns one from content representing a wide gap with a decrease in the platform height and the other from players' action standing for the jumping action followed by a standing which are actions that are mostly performed in response to the existence of a gap or an enemy.

Overall, an engaging Super Mario level, according to this analyses, is the one that provides enough space for running, changes in platform height, items to be collected as well as it contains challenging elements presented in the placement of enemies around collectable items, the existence of gaps and the placement of not easily collectable items.

9.2. NONLINEAR RELATIONSHIPS

Table 9.3: The features selected from the set of direct and sequential features for predicting engagement, frustration and challenge using sequential feature selection with SLP and simple MLP models.



















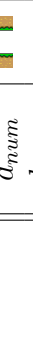


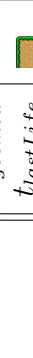



















Direct		Sequential				Fused	
	P	I	A	$A_{0.25sec}$	C_g	M	
Engagement							
SFS_{slp}	      	000       	$(\blacktriangle)(\blacktriangleright)(\blacktriangle)$ $(R^{\blacktriangleright})(R^{\blacktriangleright\uparrow})(R^{\blacktriangleright\uparrow})$ $(\uparrow^{\blacktriangleright})(\uparrow)(S)$ $(R^{\blacktriangleright\uparrow}, \uparrow^{\blacktriangleright})(S)$ $(\blacktriangle)(\uparrow^{\blacktriangleright})(S)$ $(R^{\blacktriangleright})(S)(S)$ $(\uparrow)(\uparrow)(\blacktriangle)$	$(\blacktriangle)(\blacktriangle)(\blacktriangle)$ $(\blacktriangle, \uparrow^{\blacktriangleright}, S)$ $(R^{\blacktriangleright}, R^{\blacktriangleright}, R^{\blacktriangleright\uparrow})$ $(\blacktriangle)(\uparrow, S)$ $(S)(\blacktriangle)(S)$ $(\blacktriangle, R^{\blacktriangleright}, R^{\blacktriangleright\uparrow})$ $(\blacktriangle)(S)(\uparrow)$ $(R^{\blacktriangleright})(R^{\blacktriangleright}, R^{\blacktriangleright\uparrow})$	$(\circ)(\circ, \circ)$ $(\blacktriangle, \circ, \circ)$ $(\blacktriangle, \blacktriangle, \blacktriangle)$ (\circ, \circ, \circ)	$(\blacktriangle)(\circ)(\circ)$ $(R^{\blacktriangleright})(R^{\blacktriangleright\uparrow})(\blacktriangle)$ $(S)(\blacktriangle)(\circ)$ $(\blacktriangle, \blacktriangle, \uparrow^{\blacktriangleright})$ $(R^{\blacktriangleright})(\circ)(R^{\blacktriangleright})$ $(\blacktriangle)(\blacktriangle)(\uparrow^{\blacktriangleright})$ $(\circ)(R^{\blacktriangleright})(R^{\blacktriangleright\uparrow})$ $(\blacktriangle, \blacktriangle)(R^{\blacktriangleright})$	t_{comp} n_{coin} d_{cause} t_{small} E t_{jump} n_{block}
SFS_{mlp}	  		$(\blacktriangle, \uparrow)(\blacktriangle)$		$(\blacktriangle, \blacktriangle, \circ)$		t_{super} t_{run} n_{jump} $(\blacktriangle, \blacktriangle, \blacktriangle)$ $(\blacktriangle, \uparrow^{\blacktriangleright}, S)$
Frustration							
SFS_{slp}	     	000       	$(S)(\blacktriangle)(S)$ $(R^{\blacktriangleright\uparrow})(R^{\blacktriangleright\uparrow})(R^{\blacktriangleright\uparrow})$ $(\uparrow^{\blacktriangleright})(S)(\blacktriangle)$ $(\uparrow^{\blacktriangleright})(\blacktriangle)(\blacktriangle)$ $(\blacktriangle)(\uparrow^{\blacktriangleright})(\blacktriangle)$ $(\blacktriangle)(S)(\blacktriangle)$	$(\blacktriangle)(\blacktriangle)(\blacktriangle)$ $(R^{\blacktriangleright}, R^{\blacktriangleright\uparrow})(R^{\blacktriangleright\uparrow})$ $(S)(\blacktriangle, S)$ $(S)(\blacktriangle)(\blacktriangle)$	$(\circ)(\circ, \blacktriangle)$ (\circ, \circ, \circ) $(\blacktriangle, \blacktriangle, \blacktriangle)$ $(\blacktriangle, \blacktriangle, \blacktriangle)$ $(\blacktriangle, \blacktriangle, \blacktriangle)$ (\circ, \circ, \circ) $(\blacktriangle)(\circ, \circ)$	$(\blacktriangle)(\blacktriangle)(\circ)$ $(\blacktriangle)(\blacktriangle)(S)$ $(\blacktriangle)(S)(\blacktriangle)$ $(\circ, R^{\blacktriangleright})(\blacktriangle)$ $(\blacktriangle)(R^{\blacktriangleright})(R^{\blacktriangleright\uparrow})$ $(\blacktriangle)(\blacktriangle)(\blacktriangle)$ $(R^{\blacktriangleright}, R^{\blacktriangleright\uparrow})(\blacktriangle)$	t_{right} d_{num} d_{cause} k_{goomba} $t_{lastLife}$ \bar{G}_w
SFS_{mlp}			$(\blacktriangle, \uparrow)(\blacktriangle)$	$(\uparrow, S)(\blacktriangle)$		G	




Table 9.3 – Continued

Direct		P	I	A	Sequential $A_{0.25sec}$		C_g	M	Fused
	n_{jump}				$(\blacktriangle)(S)(S)$				n_{jump} $(\circ, \circ, \blacksquare)$ $(\uparrow, \blacktriangle, \blacktriangle)$ $(\uparrow, S, \blacktriangle)$
Challenge									
SFS_{stp}	$t_{lastLife}$ n_{jump} d_{num} n_{coin} t_{right} \bar{G}_w E_p	   	000  	$(\blacktriangle)(\blacktriangle)(S)$ $(\blacktriangle)(R^{\blacktriangle})(R^{\blacktriangle})$ $(\uparrow)(\blacktriangle)(S)$ $(\uparrow)(\uparrow)(S)$ $(\blacktriangle)(S)(S)$	$(\blacktriangle, \blacktriangle)(\blacktriangle)$ $(\blacktriangle, R^{\blacktriangle})(R^{\blacktriangle})$ $(R^{\blacktriangle})(R^{\blacktriangle})(R^{\blacktriangle})$ $(S)(\blacktriangle)(\uparrow)$	$(\circ)(\circ, \circ)$ $(\blacksquare)(\blacksquare)(\blacksquare)$ $(\blacksquare)(\blacksquare)(\circ)$ $(\blacksquare, \circ, \circ)$ $(\blacksquare)(\blacksquare, \blacksquare)$ $(\blacksquare, \blacksquare, \blacksquare)$	$(\blacktriangle)(\circ)(\blacksquare)$ $(\blacksquare)(\blacksquare, \uparrow)$ $(\blacksquare)(\uparrow)(S)$ $(S)(\blacktriangle)(\blacksquare)$ $(\blacksquare, \circ)(R^{\blacktriangle})$ $(\blacksquare, \blacktriangle, \blacksquare)$	$t_{lastLife}$ n_{jump} d_{num} n_{coin} t_{right} \bar{G}	
SFS_{mtp}	t_{left} k_{stomp}		 		$(\blacktriangle)(\blacktriangle)(S)$	$(\blacksquare, \blacksquare)(\circ)$		E_p t_{left} k_{stomp}  $(\blacksquare, \blacksquare, \blacksquare)$ $(\uparrow)(\blacktriangle)(S)$ $(\blacksquare, \blacksquare, \blacksquare)$	

9.2.2.2 Frustration

The best models for predicting frustration were constructed using the subset of direct features and the remaining controllable features and they significantly outperform all other models except for the one constructed on the fused features. These two models outperform all other models also without enforcing the use of controllable features. The models trained on patterns of players' actions achieve the highest performance among other sequential-based models (no significant difference, however).

From the features selected for predicting frustration (see Table 9.3), it appears that for a game to be frustrating, it should contain at least a certain number of gaps with certain width (both positively correlated). The number of kills of goombas and koopas points out to the importance of the number of enemies presented in the game. The selection of the features that relate to avatar death (the number of deaths, the cause of death and the time spent playing in last life) also reveals the importance of gaps and enemies since these two elements constitute major causes of death.

Selected sequential features highlight specific patterns that have an impact on reported frustration. As selected direct features already demonstrated, the existence of enemies and gaps seem to be important for predicting frustration since most of the sequential patterns of P , I and C_g contain these events. The placement of stairs around gaps or the changes in platform height within a close distance to a gap appear to have an influence on how frustrating the game perceived even with moderate to small width gaps (e.g. the pattern ). Another element that factors in the perceived frustration is the placement of several game content events within a small time window as can be seen from the example patterns:  and . It can be observed from the most frequent patterns of players' actions and the correlation between them and reported frustration that the frustrated player rapidly switches between simple actions of moving right (without speeding up), standing still and performing simple jumps.

Although only three sequential patterns have been added to the set of direct features, it appears that fusing sequential and direct features yields an increase in the performance with the models constructed on these features outperforming

Table 9.4: MLP topologies and corresponding performance on direct and sequential features. The performance of MLP models built on the subset of selected features, MLP_s is compared against the models built on selected and forced controllable features, MLP_c . The topologies are presented in the form: number of inputs (including the content features)-number of neurons in the first hidden layer-number of neurons in the second hidden layer. The models with the highest accuracies are presented in bold.

	Direct		Sequential					Fused	
	P	I	A	$A_{0.25sec}$	C_g	M			
Engagement									
$MLP_{topology}$	15-6-8	14-8-4	13-2-6	14-2-2	14-10-0	11-2-2	14-2-2	18-2-0	
MLP_s	73.50%	68.84%	72.19%	73.19%	66.85%	67.16%	63.81%	76.38%	
MLP_c	69.80%	65.80%	71.02%	68.00%	83.80%	68.00%	66.49%	72.53%	
MLP_{avg}	67.18%	64.92%	69.15%	67.07%	82.30%	63.50%	65.16%	69.31%	
Frustration									
$MLP_{topology}$	12-6-0	12-8-2	10-10-10	13-2-0	12-2-0	13-6-0	14-4-0	17-10-4	
MLP_s	83.00%	77.21%	66.66%	68.92%	68.45%	66.29%	72.88%	86.25%	
MLP_c	80.70%	71.93%	69.30%	72.50%	71.37%	68.36%	72.32%	79.84%	
MLP_{avg}	76.50%	69.38%	66.74%	68.25%	69.08%	66.21%	69.15%	77.59%	
Challenge									
$MLP_{topology}$	13-2-2	10-4-0	10-2-8	10-8-6	12-4-4	13-8-6	12-6-2	19-6-2	
MLP_s	79.10%	73.04%	69.22%	63.84%	62.83%	66.50%	68.88%	91.07%	
MLP_c	77.50%	69.60%	69.05%	70.62%	67.62%	71.29%	71.45%	86.28%	
MLP_{avg}	74.03%	68.58%	64.941%	69.19%	62.83%	67.50%	67.50%	83.64%	

all other models.

In summary, the number of gaps and their average width play major roles in perceived frustration. The more gaps and the wider they are, the more frustrating the game is specially when the gaps are combined with changes in platform height. The number of enemies has less direct influence on frustration. It is interesting to note that frustration can be predicted up to a good degree just by the changes in platform height; this can be the result of more player concentration required when height changes rapidly leading to frequent changes of performed actions. It appears that, in general, the placement of a sequence of items after each other within a small distance leads to a more frustrating game as it most likely increases the level of player confusion, cognitive load and level complexity.

9.2.2.3 Challenge

Challenge can be best predicted using a subset of fused features from direct and sequential representation with significantly better performance than all other models. The models constructed on the fused features also outperform the other models when excluding the controllable features. The model constructed from direct features also achieves a high accuracy that is significantly better than all models constructed from sequential features. The best model constructed from sequential features is based on multimodal patterns with a very close performance to the models constructed on patterns from players' actions, A , and the models constructed on patterns of game content, C_g .

The direct features selected for predicting challenge (see Table 9.3) reveal the importance of gaps and enemies since five of them relate to gaps width, placement and killing of enemies and avatar death (all positively correlated). An interesting and somehow expected feature is enemy placement (that describes the way enemies are placed; around gaps, around boxes or at random (see Section 5.3.1.2)), which is negatively correlated with challenge and adds to the difficulty of the game, in particular, when enemies are placed around gaps making it more challenging to jump over and also when placed around blocks making item collection more difficult.

Selected direct features can be better explained when analyzing the selected

sequential patterns. The presence of the standing still action in the same pattern with moving right and/or jumping suggests an existence of a challenging situation in which the player has to pause and spend sometime thinking before taking a simple action (e.g. patterns like $(\uparrow)(\blacktriangleright)(S)$ or $(\blacktriangleright)(S)(\text{👾})$). While challenge is positively correlated with the pattern $(\text{👾})(\text{👾}, \text{👾})$, a negative correlation has been observed between challenge and the pattern $(\text{👾}, \text{👾}, \text{👾})$. This can be explained by the complex situation that arises in the first case and makes jumping over a gap more challenging since the player does not have enough space to speedup before jumping; instead she has to move carefully towards the edge and press a set of combined keys in order to reach the other edge.

One should expect that the models constructed on multimodal data of content and gameplay (M) should achieve the best performance. Surprisingly, the performances obtained from these models are as high or slightly lower than the performance of the best sequential models constructed. A possible explanation is that frequent patterns of length three are rather too small to capture patterns across different data streams and longer pattern lengths should be considered. (We would likely need more data in order to effectively use longer subsequences for analysis). Another related problem is the wide diversity of players' actions when encountering the same in game situation that enlarges the size of the feature space and complicates the mining of the resulted sequences.

On the other hand, very accurate models were constructed when fusing the extracted direct and sequential features together. Using the subset of direct features selected with four sequential features (three of which relate to patterns of content) as inputs to the model yields a significant increase in the performance.

In conclusion, challenge appears to be affected more by the characteristics of particular features rather than the frequency of their appearance in the level: the width of gaps, the placement of stairs around them, the placement of enemies, the frequent changes in platform height, and the placement of items within a small distance to each other contribute to a more challenging game as they imply a higher probably of game failure (Nicollet [2004]).

9.2.3 Dataset 3: Behavioral and Visual Cues

This dataset originally consisted of a total of 380 games (190 pairs). The dataset has been cleaned and interaction instances for which visual data was corrupted were removed. After this step, the dataset consists of 167 pairs of games. In addition, a preprocessing step was applied to remove the game pairs for which players reported unclear preferences (those that were equally preferred or equally unpreferred). After this step 127, 121 and 144 game pairs remain for engagement, frustration and challenge, respectively, and these pairs are used to train models of player experience.

The following sections describe the experiments that have been conducted to construct and compare different models of player experience derived from the features extracted. We construct models based on gameplay and content features only, models from mean head movement features only and models from visual reaction features. We then investigate models constructed from fusing different modalities of player input.

We start by analyzing the features selected and the models' accuracies obtained from each feature set, then we further investigate the differences on significance between the models constructed on the different categories of features.

9.2.3.1 Player Experience Modeling through Gameplay and Content Features

All features presented in Table 8.1 are set as inputs for feature selection and model optimization. The subsets of features selected, the models' accuracies and the best MLP topologies obtained vary across the three emotional states under investigation as can be seen in Table 9.5 and Table 9.6. By constructing models based only on gameplay and content features, we are able to predict the three experience states with average accuracies (across 20 trails) higher than 72% while the best performances obtained exceed 89% for engagement and frustration. The best accuracy obtained for predicting challenge is 80.6% that is significantly lower than the ones obtained for predicting engagement and frustration (significance is set to 1%).

It is worth observing that out of 30 different gameplay and content features,

a maximum of five features only have been considered to be important for predicting each state. However, different feature subsets have been picked for each experience state with only one common feature between engagement and challenge, namely, the time spent jumping t_{jump} . Three out of the six controllable features appear in the subsets of selected features for predicting engagement and challenge, namely, the number of enemies, E , the placement of enemies, E_p and the number of powerups, N_w . Note that frustration can be predicted with the smallest subset of features (only three features have been selected), nevertheless, the prediction accuracy for this emotional state is significantly higher than the ones obtained for predicting engagement and challenge.

Although high accuracies have been obtained for predicting the three emotional states, challenge appears the hardest to model from gameplay features, while frustration is the easiest.

9.2.3.2 Player Experience Modeling through Mean Head Movement Features

In order to map visual behavior to players' reported affect, the mean head movement features presented in Table 8.8 were used as inputs to select the relevant features for predicting players' affect and optimizing the players' experience models. The results presented in Table 9.5 show that the models constructed from the head movement features, extracted throughout whole game sessions yield accuracies that are as good as the ones obtained from gameplay features, or slightly lower.

An analysis on the selected features shows that the median horizontal head rotation ($M_{horizontal}$) is an important feature for all three states, while Overall Activation (OA) and ($M_{vertical}$) are only to be found as predictors of engagement and frustration. Moreover, the energy expressivity parameter (PO) is a common predictor of both engagement and challenge.

Table 9.5: Features selected from the set of gameplay, mean head movement, *MHM* (during whole games) and visual reaction features, *VR* (during certain events) for predicting engagement, frustration and challenge. Content features appear in bold.

One Modality		Bimodality	
Gameplay	MHM	VR	Gameplay & MHM
Engagement			
<i>SFS</i>	t_{jump} k_{stomp} N_w t_{run} E_p	OA_{endRun} FL_{endRun} FL_{stomp} $PO_{startLeft}$ PO_{move} $TE_{endRight}$ $Avg_{endJump}$	t_{jump} k_{stomp} N_w $M_{horizontal}$ t_{comp} n_{box} $M_{vertical}$ FL
	OA Avg $M_{vertical}$ $M_{horizontal}$ SE PO		$TE_{endRight}$ t_{jump} B PO_{stomp} $TE_{endJump}$
Frustration			
<i>SFS</i>	$t_{lastLife}$ n_{box} t_{left}	OA_{lose} Avg_{stomp} SE_{endRun} $PO_{startRun}$ $M_{endJump}$ PO_{item}	$t_{lastLife}$ TE $n_{miscJump}$ n_{box} PO d_{num} OA
	$M_{horizontal}$ OA TE FL $M_{vertical}$		FL_{item} FL_{stomp} t_{small} FL_{lose} n_{box}

Table 9.5 – Continued

One Modality		Bimodality	
Gameplay	MHM	VR	Gameplay & MHM Gameplay & VR
Challenge			
<i>SFS</i>	t_{jump} \mathbf{E} $k_{unleashed}$ d_{num}	$M_{horizontal}$ FL PO	$M_{horizontal}$ t_{jump} t_{run} t_{super} $k_{unleashed}$ t_{small}
		FL_{lose} $PO_{startRun}$ $FL_{startRun}$ $Av_{endLeft}$	FL_{item} $FL_{startRun}$ $M_{startJump}$ FL_{lose} $SE_{endLeft}$ t_{left} $Av_{startJump}$

9.2. NONLINEAR RELATIONSHIPS

The significance test shows that the model constructed for predicting frustration significantly outperforms the two other models for predicting engagement and challenge. Note that this also applies for the models constructed from gameplay features which implies that single input modalities (behavioral or visual) are better for predicting engagement and frustration than for predicting challenge.

9.2.3.3 Player Experience Modeling through Visual Reaction Features

It was our assumption that visual reaction features during certain events (losing, making critical moves, etc.) used as the only input channel for estimating player states would yield more accurate results when compared to mean head movement features (which refer to the overall visual behavior during whole game sessions) or gameplay features. Player experience states seem to be mostly correlated with events occurring at certain instances during the game, rather than whole game durations-related visual features. Visual reaction features are fused on the feature level before feeding the predictive models and feature fusion is expected to boost the model's predictive power.

Accuracy obtained for frustration yields higher values when using visual reaction features: visual behavior during jumping, losing, running and interacting with various items appear to be good predictors of frustration. More specifically, it is typical that the Energy Expressivity parameter during interaction with items (PO_{item}) and starting to run ($PO_{startRun}$), as well as the Overall Activation when losing (OA_{lose}) are related to the notion of frustration. In addition to frustration, very good accuracies have been obtained when using the visual reaction features for predicting challenge with both frustration and challenge significantly outperforming the accuracies obtained for predicting engagement.

9.2.4 Fusing Features for Modeling Player Experience

This subsection presents experiments with bimodal features as inputs to the predictive models. We first fuse the gameplay/content with the mean head movement features and we then examine the impact of the fusion between gameplay/content and the visual reaction features on the prediction accuracy of the models.

Table 9.6: MLP topologies and corresponding best (MLP_{max}) and average (MLP_{avg}) performance on gameplay, mean head movement (MHM) and visual reaction (VR) features. The topologies are presented in the form: number of inputs–number of neurons in the first hidden layer–number of neurons in the second hidden layer. Best performance values obtained (that do not show significant difference with each other) for each player experience state appear in bold.

	One Modality		Bimodality		
	Gameplay/Content	MHM	VR	Gameplay & MHM Gameplay & VR	
Engagement					
$MLP_{topology}$	5-6-0	6-4-0	7-4-6	8-4-6	5-2-0
MLP_{avg}	78.69%	74.23%	78.06%	77.78%	83.97%
MLP_{max}	89.68%	78.57%	86.51%	89.68%	91.27%
Frustration					
$MLP_{topology}$	3-8-2	5-4-0	6-8-10	7-4-4	5-8-0
MLP_{avg}	83.5%	83.04%	86.21%	83.71%	85.92%
MLP_{max}	89.17%	89.17%	92.50%	92.5%	89.17%
Challenge					
$MLP_{topology}$	4-4-2	3-4-8	4-10-8	6-10-10	7-10-10
MLP_{avg}	72.36%	75%	84.13%	77.36%	78.40%
MLP_{max}	80.56%	79.17%	88.88%	85.41%	86.81%

9.2.4.1 Modeling through Gameplay/Content and Mean Head Movement Features

Using head movement features throughout whole game sessions along with gameplay/content features yield accurate results for predicting engagement, frustration and challenge.

Different gameplay and head movement features have been selected for predicting each reported state. Median horizontal and vertical head directionality, together with fluidity in motion, along with gameplay/content features (number of killed enemies by stomping, time spent jumping and completing the whole game and powerups) resulted in a model for predicting engagement with up to 89.68% accuracy. Some of these features (such as the number of powerups, N_w , the time spent jumping, t_{jump} , the median horizontal and vertical head direction, $M_{horizontal}$ and $M_{vertical}$) also appear in the subset of features selected when constructing models from each one of these two modalities on its own. This indicates the importance of the features as predictors of player engagement.

The subset of features selected for predicting frustration includes: Temporal, Energy and Overall Activation expressivity parameters being utilized along with $t_{lastLife}$, $n_{miscJump}$, n_{box} and d_{num} . The Temporal (TA) and Overall Activation (OA) features also appear in the subset of features selected for predicting frustration from only mean head movement features. Unsurprisingly, the time spent playing during the last life ($t_{lastlife}$) and the number of boxes pressed or destroyed (n_{box}) are important predictors of frustration. These gameplay features also appear in the model constructed on gameplay features only.

The features selected for predicting challenge are mainly time-related gameplay features which are fused with the mean head horizontal rotation ($M_{horizontal}$). The gameplay features selected that also appear in the subset of features selected for predicting challenge with only gameplay as input include the time spent jumping (t_{jump}) and the number of opponents that were killed by unleashing a turtle shell ($k_{unleashed}$). The new time-related gameplay features selected (t_{run} , t_{super} and t_{small}) result in an average performance increase of 5% (compared to the average performance of the models built on gameplay features only) indicating the importance of time spent running and time Mario being in large or small mode as

predictors of player challenge.

The t-test shows that the accuracies obtained from the model constructed for predicting frustration are significantly higher than the ones for predicting engagement and challenge (Note that this finding is similar to the ones observed when testing for differences of significance in mean performance values between the models constructed from gameplay features only and from mean head movement features only).

9.2.4.2 Modeling through Gameplay/Content and Visual Reaction Features

Combining gameplay/content features and visual reaction, results in the appearance of features not used when using each one of the two modalities by themselves. This may be attributed to the fact that there are correlations between features used by gameplay/content and visual reaction features alone. As feature selection seeks beyond linear correlated features, new selected feature subsets are expected to be derived for maximizing performance accuracy.

For engagement, a smaller subset of combined features resulted into a higher accuracy than using larger sets of features from each of the two input modalities alone. Most of the features selected do not appear in the subset of features selected for predicting engagement from each of these two modalities at a time. The majority of the features selected are directly or indirectly linked to head movement and gameplay events while jumping; t_{jump} is an indication of the time spent jumping, PO_{stomp} is the head movement energy while stomping on an enemy which is an action that requires jumping, $TE_{endJump}$ is the temporal expressivity parameter when landing, and B refers to the number of boxes which require a jump to interact with. It therefore expectedly appears that the jump event is a contributor for the prediction of engagement in platform games as the average accuracy achieved for engagement (83.97%) via the bimodal fusion of gameplay and visual reaction features is the best obtained across any other feature type as model input.

The selected subset of features for predicting frustration also contains fewer features than the ones selected individually for each modality. It is interesting

9.2. NONLINEAR RELATIONSHIPS

to note that there is no overlap between the features selected from the fused features and the ones selected from the visual reaction features while there is only one common feature (n_{box}) between the selected fused features and the features selected from gameplay.

The feature subset selected for predicting challenge contains a larger number of features when compared to the ones selected from each modality alone. By looking at the features selected for the three modes — the models constructed from gameplay features, the model constructed from visual reaction features, and the model constructed from fusing these two modalities — it appears that there are two overlaps between the visual reaction features selected ($FL_{startRun}$ and FL_{lose}) and there is no gameplay feature in common. The resulting average performance for challenge (78.4%) suggests that the new features selected do not improve the predictive power of the model when compared to the corresponding performance of the visual reaction features. The statistical analysis shows no significant performance difference between the models constructed for predicting engagement and frustration while these two models' performances are significantly higher than the performance of the model constructed for predicting challenge.

9.2.5 Significance Analysis

We perform a statistical analysis to test for significant differences in the accuracies obtained from the models constructed on all different categories of features. Figure 10.1 presents the results obtained from testing for significant performance differences between the models constructed on all categories of features across the three emotional states (significant effect is obtained through $p - value < 1\%$). A significant difference on average performance is illustrated with a solid arrow, while a dash arrow depicts average performance differences of no statistical significance. The p-values obtained from the statistically significant differences are also presented. For example, the performance of the model constructed for predicting engagement from the subset of gameplay features is significantly higher than the one constructed from the selected head movement features; this relationship is shown as a solid arrow from gameplay to head movement in Fig. 10.1.(a). On the other hand, the model constructed on the fused features of gameplay and head

movement for predicting frustration yields a higher performance than the one obtained from the model constructed from gameplay or head movement features on their own (Fig. 10.1.(b)).

As can be seen from Fig. 10.1, mean head movement features do not yield high performances compared to the other features when used on their own; all models constructed from other feature sets yield higher or significantly higher performances than the model constructed based on the mean head movement features for engagement. These features, however, outperform (with no significant difference) the models constructed from gameplay features for predicting frustration and challenge. Fusing the mean head movement features with gameplay features, nevertheless, resulted in better accuracies than the ones obtained when only mean head movement features are used to construct the player experience models for all emotional states. The accuracies obtained are even better than the ones obtained from gameplay features for predicting frustration and challenge.

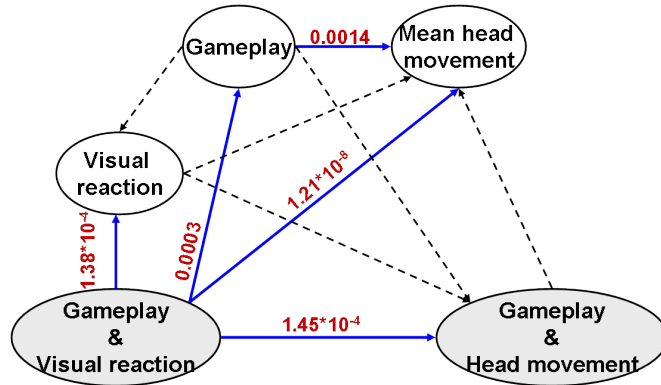
Results obtained from models constructed on visual reaction features, on the other hand, are better than the ones obtained from the models constructed on mean head movement features or on gameplay features for predicting frustration and challenge. These models also improve upon the models constructed on the fused features of gameplay and mean head movement for all emotional states.

By fusing visual reaction features with gameplay features, we were able to construct models with higher performance in predicting engagement than any other models constructed from any other feature sets. This argument also holds for frustration and challenge except for the model constructed from visual reaction features which outperforms the model constructed from fusing these features with gameplay features.

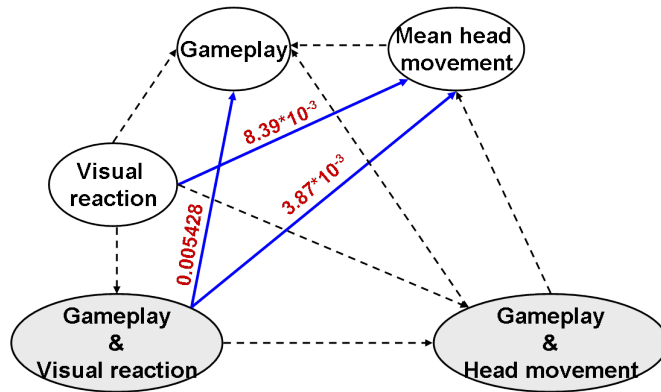
Fusing features from different modalities, in general, appears to result in more accurate models for predicting players' affect than the ones obtained when constructing models from features extracted from one modality. Fusing the features (i.e. visual reaction features) empowers the models with implicit knowledge about more than one channel of information that appears to have a positive impact on the models' performance.

We have anticipated that fusing gameplay and visual reaction features would yield higher accuracies than when using any other feature set. But our assump-

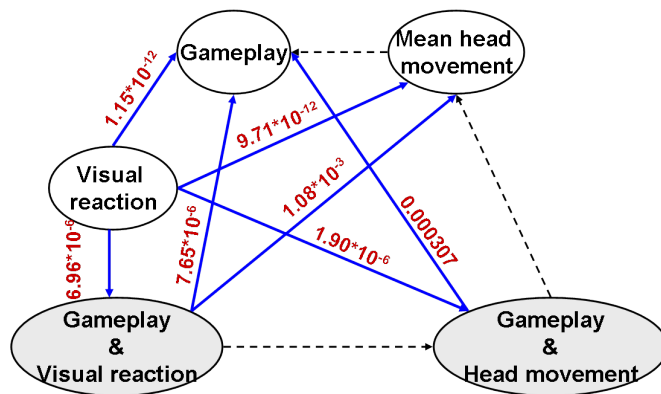
9.2. NONLINEAR RELATIONSHIPS



(a) Engagement



(b) Frustration



(c) Challenge

Figure 9.1: Testing for statistical significance between the obtained performance of the different sets of features examined for modeling player experience. Solid arrows between two feature sets depict a significant difference on the average performance between them. Dash arrows depict average performance differences of no statistical significance. P-values are added next to significant differences.

tion does not hold for the state of challenge. Analyzing the features selected and their correlations with players' preferences would help us shed some light on this effect. However, the models constructed for predicting challenge from visual reaction features and from fusing these features with game play features are multi-layered perceptron of two hidden layers which further implies that the relationship between the features selected and the reported players' preferences is more complex than simple linear correlations.

In order to be able to analyze the non-linear function between an ANN's input and output vectors we investigated the impact of each of the selected features (ANN input) on the prediction accuracy (ANN output) by altering the value of each input exhaustively while keeping the values for all other features (ANN inputs) constant. This process has been repeated for all selected features from visual reaction and for the subset of selected features from gameplay and visual reaction. Results indicate that for the challenge model constructed from visual reaction features, the energy of head movement while starting to run ($PO_{startRun}$) has the largest impact on the prediction accuracy followed by the fluidity of head movement also when starting to run ($FL_{startRun}$). These two features were found to have a strong negative correlation with challenge. On the other hand, $PO_{startRun}$ has not been selected for the model constructed from gameplay and visual reaction features and the strength of the negative correlation for the head movement fluidity feature ($FL_{startRun}$) has been found to be weaker when fusing the two modalities. The strongest correlation observed is with the time spent moving left (t_{left}), which, unsurprisingly, is positively correlated with reported challenge.

We anticipate that the performance decrease obtained when fusing the features is the result of the feature selection approach followed which fails to select the optimal subset of features for prediction when the pool of features to select from become large. For instance a total number of 114 features is reached when fusing gameplay features with visual reaction features.

To further analyze the effect of the interaction between the features on the models' accuracies, we run a two-way ANOVA test. For this test, two factors have been considered: 1) the existence (versus non-existence) of the gameplay features for the prediction of affect, and 2) the existence of visual reaction features (versus

9.2. NONLINEAR RELATIONSHIPS

Table 9.7: P-values obtained from the two-way ANOVA test. The two factors considered are existence (versus non-existence) of the gameplay features and the existence of visual reaction features (versus head movement features). Significant effects appear in bold.

Source (Factors)	Engagement	Frustration	Challenge
(A) Gameplay (vs. no-gameplay)	0.00001	0.78	0.03
(B) Visual reaction (vs. head movement)	$4.21 * 10^{-6}$	0.0004	$3.54 * 10^{-9}$
(A*B) Interaction	0.13	0.512	$1.05 * 10^{-6}$

head movement features). Such an analysis would help us investigate whether the use of visual, or alternatively head movement, features or the fusion of gameplay with visual cues would yield significant changes in the models' performance. The results of a 2×2 ((gameplay and no-gameplay) \times (visual reaction and head movement)) between-groups two-way ANOVA are presented in Table 9.7.

Both independent variables seems to have an impact on engagement prediction with p-values of 0.0001 and $4.21 * 10^{-6}$, respectively. However, no significant effect was identified when analyzing the interaction between the variables ($p - value = 0.13$). As for frustration, the results showed significant difference only for the second factor ($p - value = 0.0004$) while no significant effects were observed for the first factor ($p - value = 0.78$) or for the interaction between the factors ($p - value = 0.512$). Finally, for challenge, significant effects were observed for both factors ($p - value = 0.03$ and $p - value = 3.54 * 10^9$) and for the interaction between the factors ($p - value = 1.05 * 10^{-6}$). These results suggest that the type of the visual cues has a significant impact on the prediction accuracies for the three emotional states, while the inclusion of the gameplay features was found to have a significant effect on predicting engagement and challenge. The interaction between gameplay and visual cues features, on the other hand, was found to have a significant effect only on the prediction of challenge.

9.2.5.1 Adjusting the Models for Control

Forcing all the remaining controllable features that have not been selected in the feature selection step into the inputs of the constructed ANN models is one possible approach to gain control over the generation process. This approach has been followed when adjusting the models constructed in dataset 1 and dataset

Table 9.8: The best (MLP_{max}) and average (MLP_{avg}) performance of the models constructed for predicting engagement, frustration and challenge from visual reaction and gameplay features after forcing the controllable features.

	<i>Engagement</i>	<i>Frustration</i>	<i>Challenge</i>
MLP_{avg}	76.94%	79.33%	75.24%
MLP_{max}	80.95%	85.83%	82.64%

2. However, a performance decrease should be anticipated when adopting such an approach since these features have not been selected in the feature selection step and, hence, they are likely to be irrelevant for predicting the corresponding experience state. For this reason, all remaining non-selected controllable features have been considered by a feature selection procedure implemented to pick the largest possible subset of controllable features that yields the minimum possible decrease in the models' performance. These subsets of selected features have been set as additional inputs of the ANN models constructed from the selected gameplay and visual reaction features.

Different subsets of controllable features have been selected for each player experience state. In particular, the subset that allows maximum content control and minimizes drop in engagement prediction accuracy constitutes of two content features: the enemies placement, E_p , and the number of powerups, N_w . On the same basis, four controllable features have been selected for the model constructed for predicting frustration: the placement of enemies, E_p , the number and the width of gaps, and the total number of boxes. The subset of the controllable features that yields the best compromise between performance and level of control for predicting challenge contains all controllable features except from the number of powerups.

After forcing these features, the average accuracy obtained (presented in Table 9.8) for predicting engagement becomes 76.94% with a maximum performance of 80.95%. For frustration, the average accuracy is 79.33% and the best performance is 85.83% while challenge can be predicted with up to 82.64% accuracy with an average performance of 75.24%.

9.3 Comparison

In this section, we investigate the generalization capability of the methodology proposed by comparing and cross validating the models built on direct features for dataset 1 and dataset 2. We further analyze the impact of the feature types on the modeling accuracies by comparing the different models constructed from the data collected in dataset 2 and dataset 3.

9.3.1 Scalability

Since the first and the third datasets construct models of player experience based on direct features using the same methodology and the main difference is in the size of the dataset and the length of the game sessions, it is worth comparing the models' accuracies and selected features for the three emotional states and investigate how well the methodology proposed scales for a much larger dataset and smaller game sessions in dataset 2 compared to dataset 1. For comparison purposes, we here assume that players' reported fun is consistent to the level of reported engagement, even though this is not entirely accurate. However, we ground this assumption on several studies reported in the literature where fun is used as a synonymous to engagement and both refer to a notion of an entertaining game experience (Read et al. [2002]; Yannakakis et al. [2008]).

The analysis based on comparing the results presented in Table 9.2 and Table 9.3 showed that for engagement, three out of the four features selected in the model built for dataset 1 have also been selected in the dataset 2 model along with seven other features. Despite the expansion in the dataset size and the use of smaller time sessions in dataset 2, the methodology proposed for constructing player experience models of engagement appears to be consistent since the two models are able to predict engagement with a relatively similar accuracy (69.80% and 69.66% for datasets 1 and 2, respectively).

Three out of seven features selected for the frustration model are common for predicting frustration. Comparing the models performance indicates that frustration can be predicted with higher accuracy from the smaller dataset and longer session time (the accuracies obtained are 89.33% for dataset 1 and 80.70% for dataset 2). This can be in part explained by the difficulty in expressing a

clear emotional preference of frustration on different short game variants since data collection in dataset 3 resulted in 169 pairs of unclear preferences compared to 103 and 71 for engagement and challenge, respectively.

Only two features generalize for the two datasets for challenge, namely, the number of collected coins, n_{coin} and the average gaps width, \bar{G}_w . Some of the other features are somehow related, more specifically, the time spent during last life, $t_{lastLife}$ correlates with the time needed to complete the level, t_{comp} and the number of death, d_{num} is a generalization of the number of times the player killed because of a cannon bullet, d_f . Overall, despite the huge increase in the dataset size, challenge is predicted with larger feature subsets and higher accuracy from shorter game sessions (74.66% and 77.50% for datasets 1 and 2, respectively).

To check for the efficiency of the feature selection approach, the impact of the selected subset of features on the prediction accuracy, the influence of the size of the game session and the generality of the proposed methodology, we evaluated the dataset 1 models on the data used to construct the dataset 2 models and vice versa.

Since we are concerned with the generality and not adaptability at this stage, we compare the models after excluding the controllable features to eliminate their effect. Zero is assigned to the features that have been extracted for one dataset while not for the other (like the number of time the player fires in the level, n_{shoot}).

The obtained accuracies for the three player experience states are presented in Table 9.9. As can be seen from the table, the best performance is obtained when the old model evaluates challenge on the new dataset (67.25%) despite that these two models share only two features. Unsurprisingly, that cross-validation performance on challenge is lower than the two corresponding models constructed and evaluated on the same dataset. While reported frustration models are the most accurate for the two datasets —and although four features are found in common between these two models — none of them managed to generalize well when evaluated on the unseen dataset.

The two models for predicting reported engagement achieved similar results when evaluated on the unseen dataset. In summary, it appears that longer game sessions are more relevant for predicting frustration while challenge can be pre-

9.3. COMPARISON

Table 9.9: The performance of the models of dataset 1 on the data collected in dataset 2 ($P_{model1/data2}$) compared to the performance of the dataset 2 models on the dataset 1 ($P_{model2/data1}$).

	Engagement (Fun)	Frustration	Challenge
$P_{model1/data2}$	58.98%	40.68%	67.25%
$P_{model2/data1}$	57.33%	58.18%	45.36%

dicted better from short game sessions.

9.3.2 Modeling Accuracy

Rich information about players behavior has been collected in dataset 2 and dataset 3. Therefore, in this section we compare the best models obtained from these two datasets across the three emotional states. It's worth noting, however, the difference in the size of the datasets. The models constructed in dataset 2 are based on behavioral data collected from 780 game pairs which is a large dataset compared to the 190 game pairs collected in dataset 3.

For the following analysis, we compare the models constructed from the set of selected features only without including the controllable features since we are more interested in investigating the effect of the type of the features on the modeling accuracy than on using the models for control.

For engagement, the models constructed on fusing direct and sequential features give the best prediction accuracy (76.38%) while by fusing gameplay and visual reaction features, we were able to predict engagement with accuracy up to 91.27%. These models, built on visual reaction and gameplay features, significantly outperform any other models built for predicting engagement. It seems that engagement is a notion related to the way a game is played, both in terms of player's actions and her visual information.

Frustration on the other hand can be best predicted from visual reaction features only (92.50%) which is significantly better than all models trained on gameplay or sequential features. Combining gameplay features with visual reaction features or with head movement features, however, yields models of no significant performance difference from the ones built with visual reaction fea-

tures only.

Visual reaction features are also the best predictors of reported challenge (88.88%) with no significant difference from the modes build on a combination of direct and sequential features (86.28%). It, therefore, appears that the player's in-game behavior and her visual reaction in response to game events are equally important for building accurate estimators of perceived challenge.

9.4 Summary

In this chapter we have presented an extensive set of experiments for building models of player experience. Datasets of different sizes and a variety of features have been considered to construct the models. Direct features, sequential patterns of game content and players' actions, head movement while playing the game and visual reactions to game events are all employed to build accurate estimators of player experience. Some of these input modalities are fused together allowing which investigation of bimodal features.

Linear and non-linear relationships between the extracted features and the reported emotional states have been examined. Highly accurate models of player experience have been constructed and used for an in-depth analysis of the factors that contribute to player experience in platform games. The thorough analysis followed shows some generic aspects of level design aesthetics that relate to the three reported emotional states: engagement, frustration and challenge.

The approach presented provides the underlying basis for game adaptation techniques that could be employed to automatically generate game content that optimizes particular aspects of player experience.

10

Game Adaptation

In this chapter we build on the player experience modeling experiments, presented in the previous chapter, and we take the approach one step further. We start by investigating the impact of the size of game session on the accuracy of predicting players' affects since this helps us set the frequency of adaptation. We then aim at closing the affective loop in the game by altering content generation via a feedback signal of the player's affective state as predicted by the constructed models.

We present two approaches for personalizing game content; an exhaustive-based search approach and a more general method by means of evolutionary search.

10.1 Feature Analysis and Adaptation Frequency

When discussing adaptation, it is very important to define the game features that should be manipulated to alter a specific experience. The frequency of adaptation, on the other hand, is another important issue that should be examined. This includes defining the minimum length of time a player needs to play in order for a particular emotion to be elicited. The size should be long enough to invoke emotional manifestations, yet short enough to enable a meaningful adaptation.

In the following sections we describe a set of experiments conducted to better understand the content-affect relationship. For this purpose, the game sessions

have been divided into smaller size sessions and player experience models have been constructed on features extracted from these mini-sessions.

10.1.1 Level Segmentation

The purpose of segmenting the level is to draw a picture of the importance of the features with respect to player experience; different features correlated with player experience for each state could be extracted from each segment of the game pointing out to positions in the games where these features play a role in triggering particular player states. By segmenting the levels we can also identify the size of the level that generates the best prediction accuracy for the three player experience states. That segment size can then potentially be used to set the frequency of a real-time adaptation mechanism.

We start the process by calculating the models' performance over the entire game session. The level is then divided into two equal segments and the values for all content and gameplay features for these two segments are recalculated. To further investigate the effect of the size and choice of the segment that gives the most useful information, we partition the levels into three equal segments and the models are evaluated on individual segments assuming that the expressed whole-game emotional preferences remain constant across those segments. No performance improvement has been obtained by further division of the level, and thus the focus of the remaining of this section is on levels divided for up to three segments.

For the remaining of this dissertation we will use the term *window* to refer to the whole game session, and the term *segment* to refer to parts of a window.

The data collected in dataset 2 is used for the experiments presented in the following sections; six content features have been used to construct the levels in this dataset and gameplay and content data has been represented via direct and sequential feature representations. For the following experiments, we use direct features of gameplay and content, sequential features of content corresponding to gameplay events, C_a , and sequential gameplay features, A , as inputs.

Neuroevolutionary preference learning approach, presented in Chapter 7, was used to build the models. We start the models' constructing procedure by select-

ing the relevant subset of features for predicting each reported player experience, this is done by using SFS to generate the input vector of SLPs. Since the focus of this experiment is on analyzing the impact of the session size on the prediction accuracy rather than on improving the model’s performance, we use small MLP models containing one hidden layer of two neurons as models of player experience. The quality of a feature subset and the performance of each MLP is obtained through the average classification accuracy in three independent runs using 3-fold cross validation across ten runs. Parameter tuning tests were to set up the parameters’ values for neuroevolutionary preference learning that yield the highest accuracy and minimize computational effort. A population of 100 individuals was used, and evolution ran for 20 generations. A probabilistic rank-based selection scheme was used, with higher ranked individuals having higher probability of being chosen as parents. Finally, reproduction was performed via uniform crossover, followed by Gaussian mutation of 1% probability.

10.1.2 MLPs Performance on Partial Information

For each of the half and one-third size segments the statistical and sequential values for the content and gameplay features (as presented in Chapter 8) have been calculated. All feature values are uniformly normalized to the range $[0,1]$ using the standard max-min normalization. Player experience models were constructed based on the different subsets of features selected from direct and sequential features for each segment across the three emotional states. Table 10.1 presents the features selected, the topologies and performance of the best models constructed for each segment. For comparison purposes, the table also presents the data about models constructed on features from the full game session.

10.1.2.1 Analysis

As can be seen from Table 10.1, the networks found vary in the number of selected features and performance. The most accurate model is the one for predicting challenge (91.23%) with a large subset of 13 features selected from the full levels. Engagement comes next with a best model accuracy of 86.43% obtained from features extracted from the first segment out of two followed by frustration which

can be predicted with an accuracy up to 85.88% from a subset of ten features extracted from the full level.

Segmenting the sessions resulted in a performance increase for the models of predicting engagement while a performance decrease has been observed for the experience models of frustration and challenge. The models constructed based on features selected from the first half of the session for predicting engagement significantly (significant effect is determined by $p < 0.01$ over 10 runs) outperform all other models constructed on full and other partial information. A significant performance decrease was found for predicting frustration when constructing the models based on features extracted from segments with one third of the full size. Using features from the full sessions, we were able to predict challenge with accuracy that is significantly higher than all other models constructed on partial information. Figure 10.1 presents the significance relationship between all models constructed for the three player states.

The results suggest that different sizes of game session are needed to elicit different affective/cognitive states. While challenge can be predicted with high accuracy from the full sessions, smaller session size somewhat count-intuitively appears to give better results for predicting engagement. Frustration can be predicted with high accuracy from full and half size sessions.

According to the results obtained, it appears that challenge requires more time than frustration and engagement to be elicited. Challenge is considered a behavioral aspect of gameplay experience that results from the interaction between the player and the game, as opposed to frustration and engagement that capture part of the affective state of the player. The results indicate that those two particular emotional states require less time to be evoked than the challenge behavioral state.

Table 10.1: The features selected from the set of direct and sequential features for predicting engagement, frustration and challenge using sequential feature selection with SLP and simple MLP models and the corresponding best (MLP_{max}) and average (MLP_{avg}) models' performance. The models with the highest accuracies are presented in bold.

	Full level	1 st seg/2	2 nd seg/2	1 st seg/3	2 nd seg/3	3 rd seg/3	
	Engagement						
Selected features	t_{comp} n_{coin} d_{cause} t_{small} t_{jump} E n_{block} t_{super} t_{run} n_{jump} $\uparrow, \downarrow, \leftarrow, \rightarrow$ $\blacktriangle \uparrow S$	n_{state} t_{super} d_{cause} t_{right} E $S \uparrow \uparrow$ $\blacktriangle \blacktriangle \blacktriangle$ $\uparrow \uparrow \uparrow$ $\uparrow \uparrow$	B d_{cause} N_w E n_{box} $\blacktriangle S$ $\blacktriangle R S$ $\blacktriangle \uparrow S$	$n_{miscJump}$ t_{left} n_{box} B \bar{G}_w d_{cause} E_p n_{coin} $\blacktriangle R S$ $S \blacktriangle S$	t_{right} E $n_{miscJump}$ d_{cause} t_{duck} G	B G t_{jump} d_{cause} $k_{unleash}$	
MLP_{max}	76.38%	86.43%	72.03%	72.19%	71.69%	72.86%	
MLP_{avg}	70.95%	80.80%	68.88%	68.74%	68.68%	70.25%	
	Frustration						
Selected features	t_{right} d_{num} d_{cause} k_{goomba} $t_{lastLift}$ \bar{G}_w	$n_{miscJump}$ G n_{coin} E t_{left} $R \blacktriangle \blacktriangle$	G $n_{miscJump}$ \bar{G}_w d_{cause} k_{stomp}	n_{jump} t_{small} t_{left} $S \uparrow S$	G B t_{small}	E_p k_{goomba} B G $n_{miscJump}$ t_{left}	

Table 10.1 – Continued

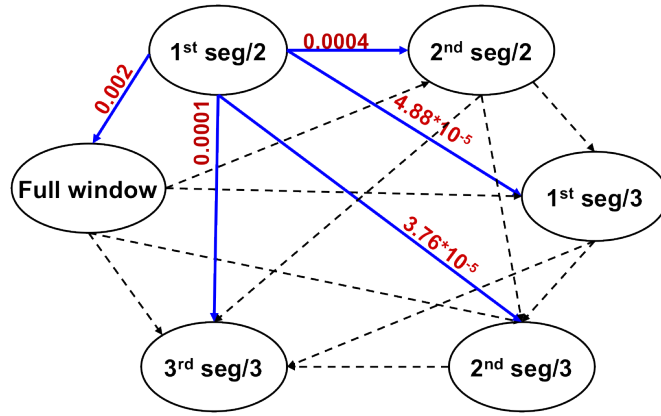
	Full level	1 st seg/2	2 nd seg/2	1 st seg/3	2 nd seg/3	3 rd seg/3
	G n_{jump} \circ, \circ, \square $\uparrow \blacktriangleright \blacktriangleright \blacktriangleright$	$R \blacktriangleright R \uparrow \blacktriangleright$				t_{right} $SS \blacktriangleright$ $R \blacktriangleright R \uparrow S$
$\uparrow \blacktriangleright S \blacktriangleright$						
MLP_{max}	86.25%	81.73%	79.85%	78.72%	78.15%	73.45%
MLP_{avg}	81.28%	77.59%	77.06%	75.63%	72.69%	69.49%
Challenge						
Selected features	$t_{lastLift}$ n_{jump} d_{num} n_{coin} t_{right} G_w E_p t_{left} k_{stomp} \circ, \circ, \circ $\square, \square, \square$ $\uparrow \blacktriangleright S$ $\square, \square, \square$	t_{right} $n_{miscJump}$ n_{state} G t_{small} B $\blacktriangleright \uparrow S$	G B d_{cause} $n_{miscJump}$	$n_{miscJump}$ n_{box} $n_{powerup}$ t_{right} n_{mode} t_{left} t_{super}	G E k_{flower} n_{jump} $k_{unleash}$ $\uparrow S \blacktriangleright$	E_p k_{stomp} B E $k_{unleash}$ $\blacktriangleright SS$
MLP_{max}	91.07%	75.6%	77.19%	72.41%	73.52%	69.38%
MLP_{avg}	87.62%	71.36%	73.68%	65.80%	70.27%	66.00%

The fact that the information gathered from the first half of the session resulted in the best modeling accuracies for engagement and frustration indicates that the features presented at the beginning of the game have the greatest impact of players' experience. However, this might also be the reason of players not being able to complete the levels and thus reporting preferences based only on the part they have experienced. This is also supported by noticing that less features of players' behavior have been selected from the second-third or last third of the levels compared to the ones selected from half-size or one-third segments.

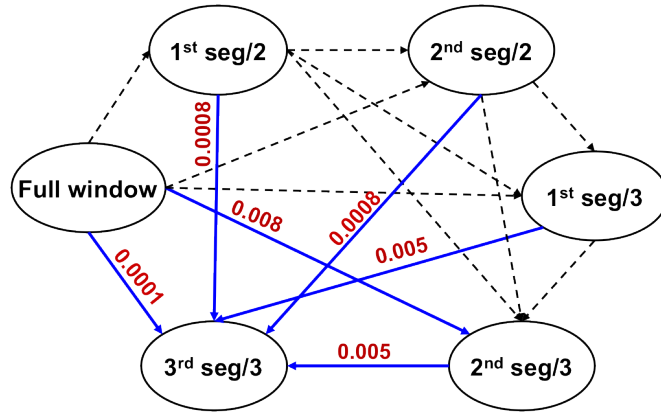
The results indicate that the models performance in general significantly decreases when segmenting the session into more than two segments. This suggests that partitioning the data into more than two segments causes information loss. Another possible explanation is that the session size should be longer than a particular length to elicit a specific player experience state, and it appears that one third of the level size is too small to consider the reported player experience valid while the gameplay experience and the reported affects can still be considered valid for one half of the session size for engagement and frustration.

The results show that the ANN preference models built on data derived from the game as a whole gives the best performance for predicting frustration and challenge over all other models that have been constructed. Thus, the results suggest that the minimum acceptable size of the segment for which the model is able to predict player's reported preferences with acceptable accuracy is the one that has been chosen in the first place when designing the experiment.

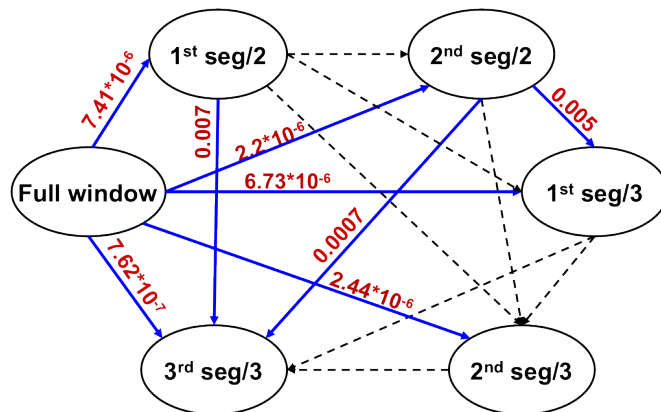
The different subsets of features selected from each segment draw a picture of the importance of the positioning of the features within the game and the different impacts this has on the different emotional/cognitive states under investigation. Some content features have been selected from the full sessions and also appear in the subset of features selected from the parts, such as the number of enemies (E) and the number of gaps (G) for predicting engagement and frustration, respectively. This suggests the importance of these features for eliciting a particular emotional state regardless of their specific positioning within the game. Other features such as the number of powerups (N_w) appears to have an impact on engagement when presented in the second half of the game. This can be explained by the fact that powerups are more important to the players towards



(a) Engagement



(b) Frustration



(c) Challenge

Figure 10.1: Testing for statistical significance between the obtained performance of the different segments examined for modeling player experience. Solid arrows between two feature sets depict a significant difference on the average performance between them. Dash arrows depict average performance differences of no statistical significance. P-values are added next to significant differences.

the end of the game since this increases their chance of winning, the selection of *cause of death* feature in all segments also supports this assumption. It is worth noting that only one controllable feature has been selected for the best model for predicting engagement and the rest of the features relate to the particular playing style for each player. Most aspects of level design appear to have a large impact on challenge since five content features (direct and sequential) have been selected for the best model of predicting challenge.

The different subsets of features selected for predicting player experience states suggest differing relative importance of design elements for different aspects of player experience. This has the potential to partly decouple dissimilar aspects of player adaptation.

10.2 Adapting Game Content

In the following sections we present our attempts to close the affective loop by applying two approaches to personalize the gameplay experience. The player experience models previously constructed are utilized as measures of content quality and their outputs are used as a feedback signal to alter content generation and enable personalized content. The next chapter presents the testing and evaluation framework designed to validate these methods.

10.2.1 Exhaustive Search

In the experiments presented in this section we describe the method followed for tailoring content generation driven by the player experience models. We focus on the models built on dataset 1 (Section 9.2.1) and dataset 3 (Section 9.2.3) as four or six controllable features were used when constructing these models enabling efficient full scan of the content feature space.

Our approach to online game adaptation is based on performing exhaustive search in the space of controllable features to find the combination that, taken together with observed gameplay features, maximizes the MLP output value.

The player experience models adjusted for control are utilized to tailor the content of the game to individual players.

The search space in these two datasets consists of a maximum of five features, being the number of controllable features enforced into the input of the models. For dataset 1, four features were used; number of gaps, average width of gaps, gap placement and number of direction switches with value ranges of $[4,10]$, $[10,30]$, $[0,1]$ and $[0,1]$, respectively. The search space is explored by starting from the minimal possible values and at each step the values are increased by 1, 1, 0.1, and 0.1 respectively, resulting to a total number of 14700 configurations. Up to five content features were forced into the inputs of the MLPs when adjusting the models for control in dataset 3. These features are: number of gaps, average width of gaps, number of enemies, enemies placement and number of boxes with value ranges of $[2,6]$, $[5,15]$, $[3,7]$, $[0,2]$, and $[0,15]$, respectively. The same procedure for exploring the space of content followed for dataset 1 is employed and an increase value of 1 is set for all features resulting to 13200 different configurations.

Each configuration of controllable features is fed (together with the recently observed gameplay features) into the MLP; the combination that maximizes the network output is chosen to generate the next level. With such a small search space (14700 and 13200 configurations for the first and third dataset, respectively) we can find the optimal configuration almost instantly, allowing real time level generation.

10.2.2 Evolving Personalized Content

The GE-based level generator is used to evolve player-adapted content by employing an adaptation mechanism as a fitness function to optimize the player experience of the three emotional states under investigation.

A set of experiments has been conducted to test this approach using the player experience models constructed based on the data collected in dataset 2. Player experience models constructed with direct features and six forced content features as inputs have been utilized as fitness functions. The fitness value assigned for each individual (level) in the evolutionary process is the output of the MLP model which is the predicted value of engagement, frustration or challenge. The MLPs output is calculated by computing the values of the models' inputs; this includes the values of the content features which are directly calculated for each level

design generated by GE and the values of the gameplay features estimated from the player playing style while playing a *previous* level.

The search for the best content features that optimize a particular state is guided by the model's prediction of the player experience states, with a higher fitness given to the individuals that are predicted to be more engaging, frustrating or challenging for a particular player.

10.3 Summary

This chapter presented methodologies for analyzing the frequency of game adaptation and illustrated two methods for quantitatively personalizing game content. The levels used for collecting the datasets and constructing the experience models have been segmented into smaller chunks to investigate the best size of a game session that yields the highest prediction accuracy. This size defines the frequency of which game content should be adapted.

Two methods for online content personalization have been presented. In the first approach, an exhaustive search method is utilized to search relatively small content spaces for the best set of content features that matches the player playing style. For larger search spaces, another stochastic global optimization technique based on evolutionary algorithms is proposed and implemented.

In the next chapter, these two methods for closing the affective loop in games are evaluated using AI agents and human players. A thorough analysis of the adaptation methods' behavior and efficiency is presented.

11

Evaluation

In this chapter, we test the two adaptation approaches presented in the previous chapter. We generate player-adapted content by employing adaptation mechanisms as fitness functions to evaluate the content generated and optimize the player experience of three emotional states: engagement, frustration and challenge. The fitness functions used are models of player experience constructed in our previous experiments from crowd-sourced gameplay data collected from hundreds of players.

11.1 AI Agents

Two different AI controllers were used to test the adaptation mechanism. Both controllers were submitted to the 2009 edition of the Gameplay track of the Mario AI Competition ¹; a competition about designing controllers that play Infinite Mario Bros as well as possible, in the sense of completing as many levels as possible.

The two agents are the following:

1. The agent that won the competition, submitted by Robin Baumgarten (Togelius et al. [2010a]). This agent is based on an A* search algorithm in state space and simulates the future trajectory of both itself and enemy NPCs for each considered action. It performs very well on the type of levels generated

¹www.marioai.org

11.2. DATASET 1: BASIC PARAMETERIZED GENERATOR

by the level generator, as evidenced by it managing to finish all levels in the competition.

2. The competition entry of Sergio Lopez. This agent is based on a relatively simple heuristic function that decides when to jump and how high, and otherwise walks left. While performing less well than Robin’s agent, it could still complete most of the levels in the competition.

These two agents have been chosen because of their good performance as evidenced by them managing to win a large number of levels and because they exhibit a varying playing style. Figure 11.1 presents several statistical gameplay features extracted from 100 levels played by each agent. The figure illustrates the differences in playing styles between the two agents; while the A* agent tends to spend most of the time running and jumping, Sergio’s agent appears to be moving slower and performing jumps only when necessary which lead to more time spent to finish the levels compared to the time needed by the A* agent. These behavioral differences are directly reflected in metrics used by the player experience model, allowing us to test the adaptation mechanism with automated play-through of significantly different styles.

Using such controllers with varying playing styles, we are able to test the adaptation mechanism ability to recognize different playing characteristics and evolve levels accordingly.

In the following sections we present the different experiments conducted to test the ability of the adaptation mechanism to generate personalized content and to recognize differences in playing styles.

11.2 Dataset 1: Basic Parameterized Generator

The first set of experiments is conducted based on the player experience models constructed for dataset 1. As discussed in the previous chapter (Section 10.2.1), an exhaustive search method is used to select the combination of values for the content features that maximize the model’s output for a specific state given a particular set of gameplay features. In the following experiments, we focus on

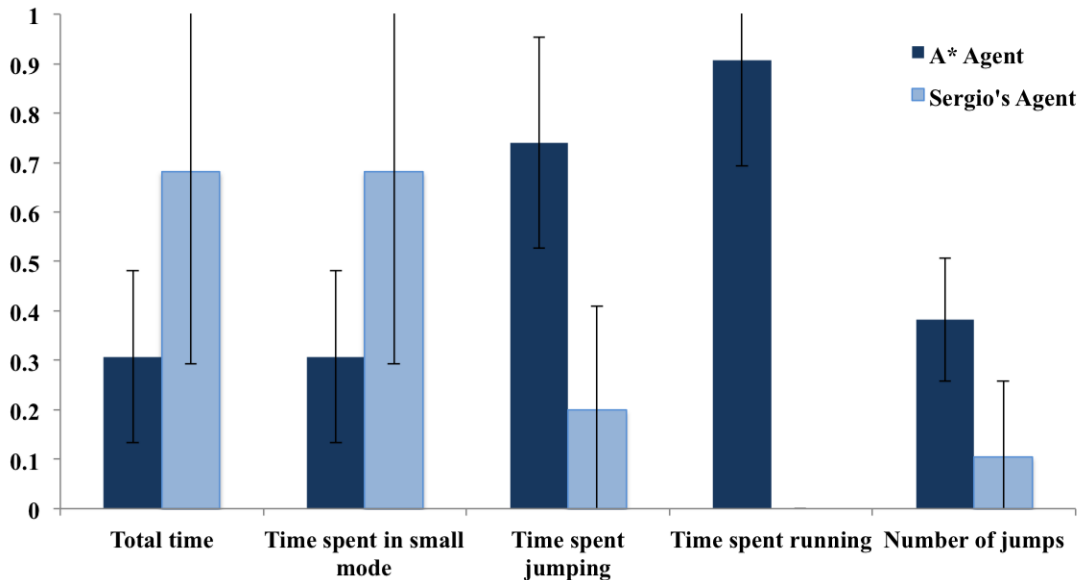


Figure 11.1: Average and standard deviation values of several gameplay statistical features that have been extracted from 100 different sessions played by the two agents.

generating adapted levels for maximizing reported fun. Further experiments on optimizing reported frustration and challenge are conducted on datasets 2.

11.2.1 Experiment 1: Optimizing Player Experience for a Fixed Playing Style

The experiment presented in this section is focused on generating optimized levels for a particular player. The experiment is done in the following steps:

1. An initial level is generated with random parameters.
2. An AI agent plays the level while gameplay features are recorded.
3. Using the set of recorded gameplay features, the values for the controllable features that optimizes the player experience are chosen.
4. A new adapted level is generated based on the optimized controllable features.

11.2. DATASET 1: BASIC PARAMETERIZED GENERATOR

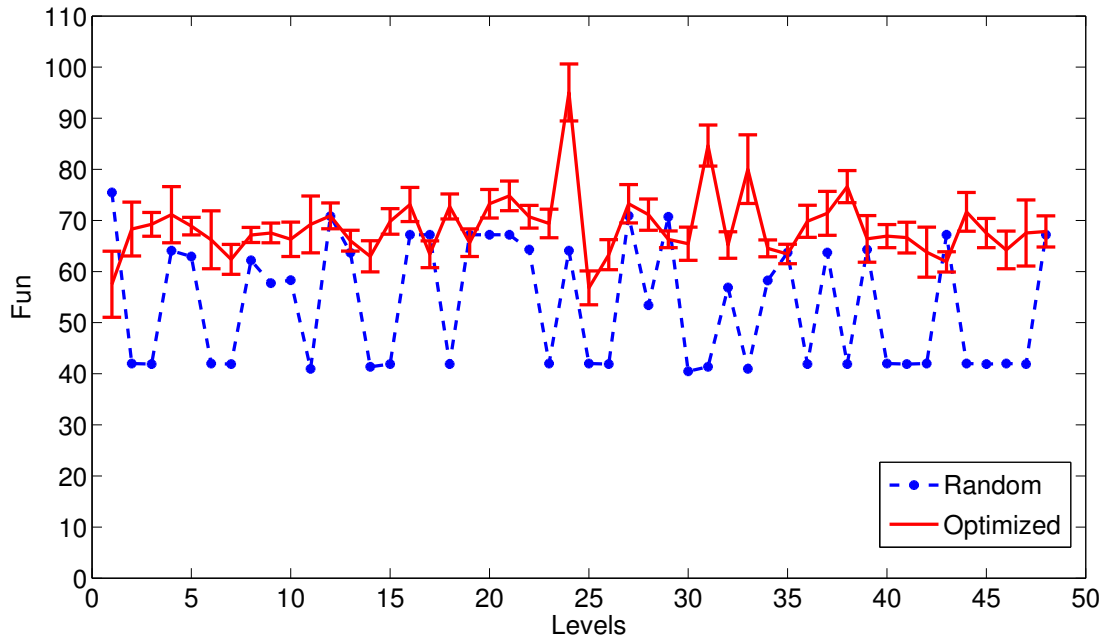
We assume that the playing style is maintained during consecutive game sessions and thus a player’s characteristics in a previous level provide a reliable estimator of his gameplay behavior in the next level. The adaptation mechanism is capable of generating a new level that optimizes some aspect of predicted player experience almost instantaneously given the playing style of the previous level. To compensate for the effect of learning while playing a series of levels, the adaptation is applied considering the recent playing style, which is the player performance in a previous level. Thus, in order to effectively study the behavior of the adaptation mechanism, it is important to monitor this behavior over time.

The two AI agents presented in Section 11.1 have each played a set of 50 levels with the first level generated randomly, followed by a set of adapted levels that aim at maximizing player experience based on gameplay during the previous level. The experiment is repeated 10 times for each agent starting from a different random level each time.

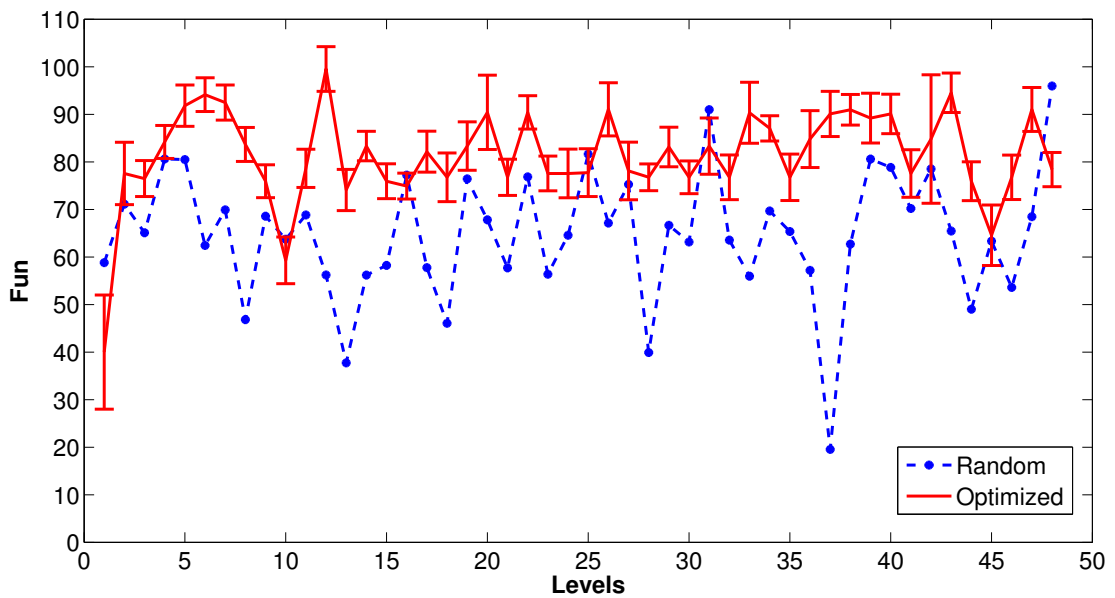
For comparison purposes, 50 random levels were generated by assigning random values for the controllable features instead of searching for the optimal set of values. The results depicted in Figure 11.2 show the performance of the proposed approach against the randomly generated levels. As can be seen from the figure, the adaptation mechanism is able to generate levels with higher predicted fun level than the baseline random levels. (In the discussion of this section, “fun” always refers to *predicted* fun levels. We do not claim that the algorithms had fun.)

The results indicate that the algorithm was able to generate levels for Sergio’s agent that are 81.31% fun, on average, while the optimized fun levels for the A* agent have a slightly lower fun value with an average of 68.71%. This difference is mostly related to the agent’s playing style. The observation of each agent playing characteristics showed that Sergio’s agent plays in a more human-like style; it moves at an average speed so it is able to complete the levels approximately at the same time as a human player would, and it jumps only when necessary. Robin’s agent, on the other hand, is able to clear the levels fast, it keeps jumping and it fires even when unnecessary.

In general, the method is able to construct levels with higher fun values for the agent who plays in a human like manner. Since the model has been trained



(a) A* agent



(b) Sergio's agent

Figure 11.2: Optimized fun levels versus random levels for the two agents.

on data collected from human players, we suspect that this is due to the nature of the data our fun model was built on. Sergio’s agent playing style may be a better match of patterns of behavior existent in human players.

11.2.1.1 Statistical Analysis

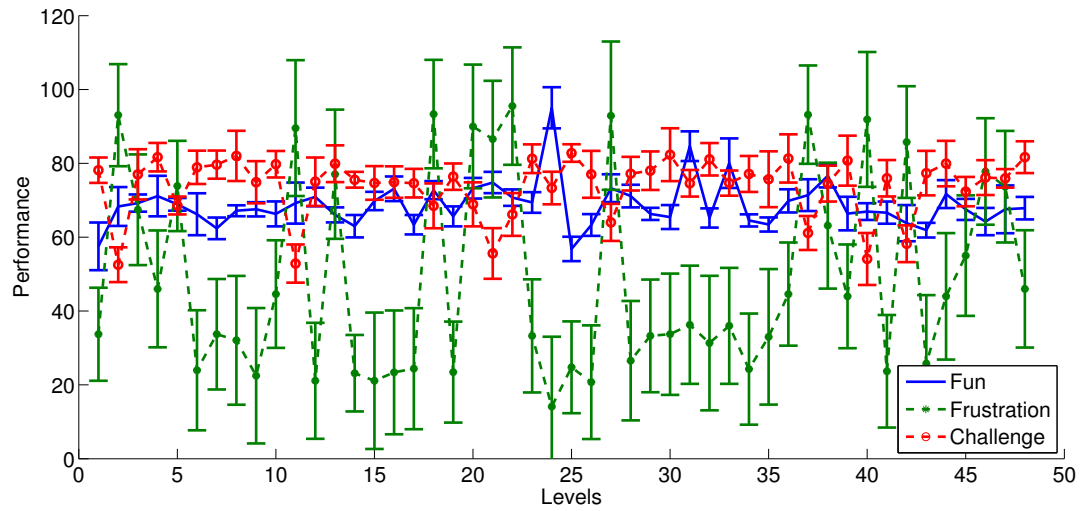
To check whether optimizing for one reported state affects the others, we generated levels to optimize the fun value while monitoring how the values change for frustration and challenge. We performed an analysis for exploring statistically significant correlations between the models’ output for these three emotions. The results of the statistical analysis are presented in Table 11.1. A statistically significant effect ($p < 0.01$) is observed between challenge and frustration ($p = 3.57 * 10^{-20}$) for the A* agent. The negative correlation (-0.66) obtained between these two emotions shows that for players like the A* agent challenging levels are predicted to be less frustrating.

The statistical analysis for Sergio’s agent showed no significant correlation between any of the emotions investigated. This shows the sensitivity of the model to the type of player and his playing style. Another interesting observation is that the levels optimized for fun induced high levels of challenge for both agents while low level of frustration is observed.

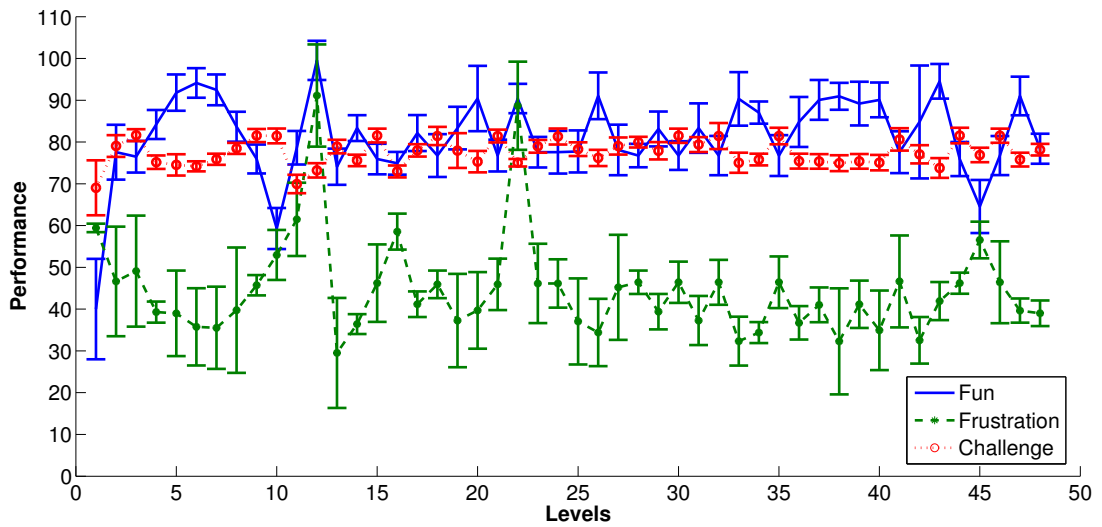
It’s also worth noting the fluctuation frustration curve while optimizing reported fun. Figure 11.3(a) shows many changes in the values obtained for predicted frustration with most of the levels having very low values (the average value is 48.92%) and very few of them having relatively high values.

These observations have not been obtained for Sergio’s agent. Fun and challenge were found to be negatively correlated though the effect was not significant. An interesting observation for this agent, however, is that although fun and frustration were found to be negatively correlated, one can notice the few levels where values of high level of frustration and high level of fun were obtained.

Investigating game content and player behavior at such points is of no doubt useful since this will helps us better understand the interaction between the player and the game and the impact of the design choice on player experience. The main focus of this experiment, however, was on monitoring the method performance



(a) A* agent



(b) Sergio's agent

Figure 11.3: Optimized fun levels while monitoring the changes in predicting frustration and challenge for the two agents.

11.2. DATASET 1: BASIC PARAMETERIZED GENERATOR

Table 11.1: Correlation coefficient values obtained between the predictions of fun, frustration (F) and challenge (C) when optimizing fun while monitoring the prediction of the other emotional states. Strong correlations ($p - value < 0.01$) are presented in bold.

	<i>Frustration</i> <i>Challenge</i>	
	A* agent	
<i>Fun</i>	0.11	-0.17
<i>Frustration</i>		-0.66
	Sergio's agent	
<i>Fun</i>	-0.13	-0.19
<i>Frustration</i>		-0.23

and hence we did not register game content or player behavior. Therefore, for the experiments conducted later, information about the game and in-game interaction were saved enabling a more in-depth analysis.

11.2.2 Experiment 2: Dynamic Adaptation to Changing Playing Styles

In the following experiments we test the model's ability to generalize over different types of players. This is done by evaluating the method with AI agents and human players.

11.2.2.1 AI Agents

The two AI agents (Section 11.1) were set to play in turns while monitoring how the fun value evolves. The experiment starts from a randomly generated level. The agents play a set of 100 levels and every 20 levels the playing agent is switched. The result in Figure 11.4 shows the changes in the prediction of fun over 100 levels. The figure illustrates that the fun value ranges around 70% in the first 20 levels where Robin's agent is playing, and when we switch the agent to Sergio's agent in the following 20 levels, the average value increases to 80% approximately, and it drops again to 70% when Robin's agent is set back to play. These results provide evidence for the model's ability to adapt to the differences in players' behavior.

It is interesting to observe the method behavior at the specific points where the agent is switched: the points corresponding to levels 21, 41, 61 and 81 in Figure 11.4. These levels are generated to maximize fun for one agent while played by the other agent; i.e. the values of the content features have been chosen to maximize the model’s output based on the gameplay data for *agent A*, but they are played and evaluated based on the gameplay data for *agent B*. Therefore, a performance drop has been obtained for all of these levels.

11.2.2.2 Human Players

For further investigation, we did the same experiment on human players playing a smaller set of 12 levels. The outcome of this experiment is illustrated in Figure 11.5, which shows the evolution of fun over 48 levels played consecutively by four different human players. The results obtained showed that levels with high fun values have been generated for all human players. The average values obtained are 80.65%, 93.88%, 82.11%, and 85.49% for first, second, third and fourth player respectively. As seen from Figure 11.5, the adaptation mechanism is robust enough to adapt to a particular player and to generalize over different types of players. Ten participants were asked to report their preferences between the randomly generated level and the first adapted level following the 4-AFC protocol. The results showed that six out of ten of the participants enjoyed an adapted game more than a randomly generated one.

11.3 Dataset 3: Behavioral and Visual Cues

The richness of the data collected in this dataset makes it interesting to test the behavior of the adaptation mechanism. Furthermore, we are not aware of a previous data-driven approach for procedurally generating content driven by computational models of fused modalities of player input.

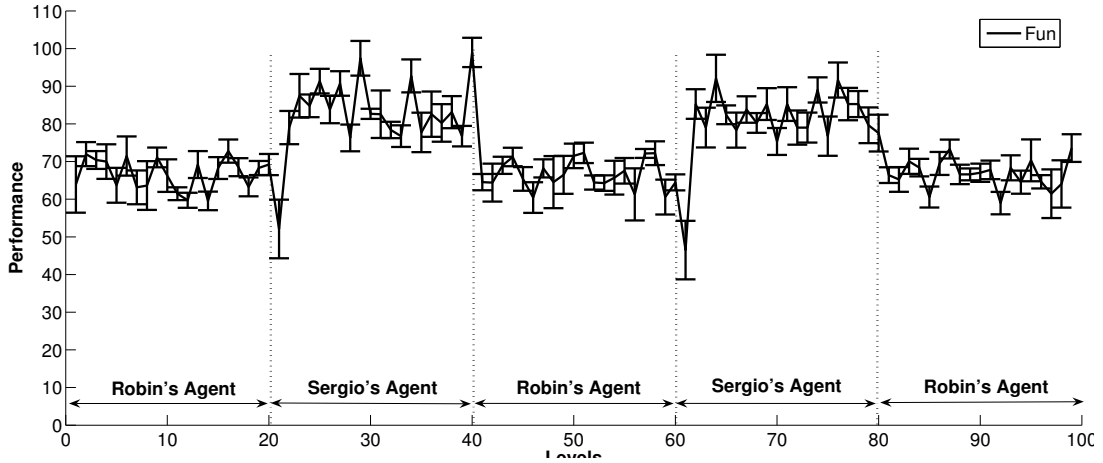


Figure 11.4: Optimized fun levels for the two AI agents

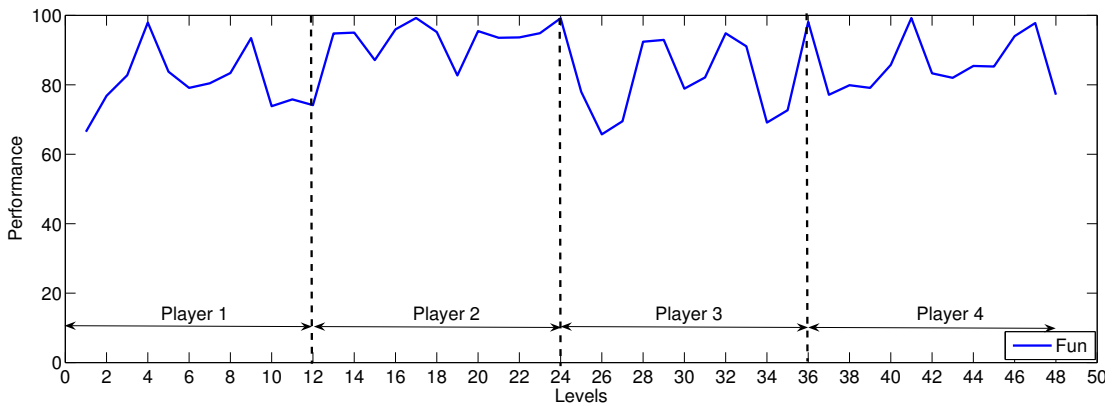


Figure 11.5: Optimized fun levels for four human players

11.3.1 Optimizing Player Experience for a Fixed Playing Style

To generate levels that are tailored to an individual player, we implemented the exhaustive search method presented in Section 10.2.1. We use the player experience models constructed on fused features of gameplay and visual reaction.

As a proof-of-concept experiment, we generate levels that maximize predicted engagement, frustration and challenge for two human players having different visual reaction features. These two samples have been selected from the recorded corpus and have not been used for constructing the experience models. The con-

tent space is explored exhaustively. As presented in Section 10.2.1, the search space consists of 13200 total configurations corresponding to the different variations of content features. This guarantees an efficient exploration of the search space and an online adaptation of game content.

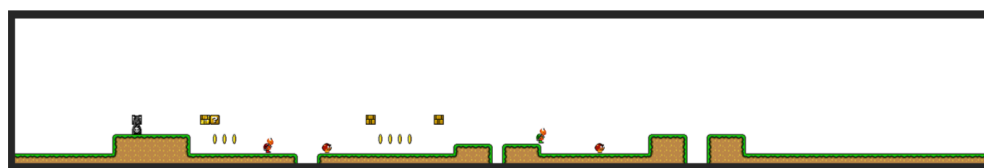
The combination of content features that maximizes the ANN output — given the player behavior and visual reaction — is chosen to generate the game level. Using this adaptation mechanism, we were able to generate a new level for each player that optimizes some aspects of predicted player experience. Figure 11.6 presents the different levels adapted for the two human players chosen along the three experience states.

The efficiency of the adaptation mechanism can be investigated by comparing the generated levels along the dimensions of the content features optimized to generate them for each state. For example, the number of enemies and the number of powerups are the two content features used to tailor level generation to maximize predicted engagement, hence, when comparing the two levels generated for the two human players, one should identify aspects of game content along these two dimensions (refer to Section 9.2.5.1 for the content features selected for each player experience state).

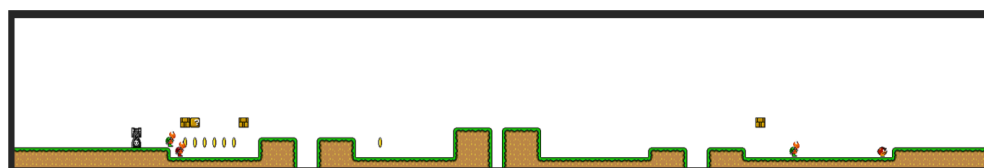
By investigating the generated levels following this approach, it appears that different values have been assigned to the content features when optimizing predicted engagement; while the two players appear to enjoy levels with powerups, the level appears to be more engaging for the first player when enemies are placed around gaps while the second player enjoys a level with enemies scattered randomly around the level.

On the other hand, levels generated to maximize predicted frustration can be compared along four dimensions (the placement of enemies, the number and the width of gaps, and the total number of boxes). A level can be more frustrating for the first player when it contains more gaps with small width, a large number of boxes, and enemies scattered randomly around. A level with fewer gaps having small width and enemies around them is found to be more frustrating for the second player.

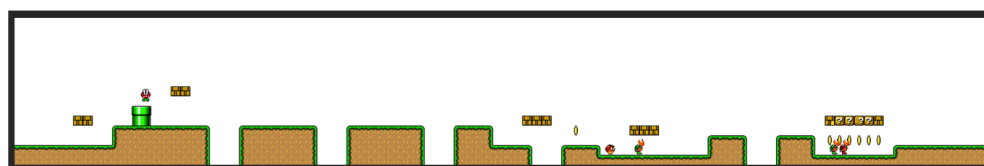
A challenging level for the first player is the one containing small width gaps, a small number of enemies scattered randomly around the level, and no boxes.



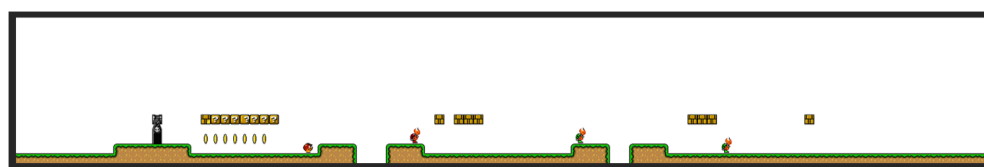
(a) Generated level for maximum engagement (Subject no. 1)



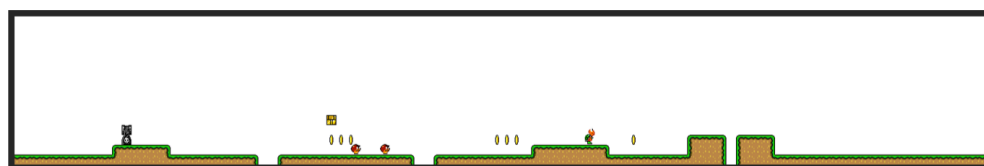
(b) Generated level for maximum engagement (Subject no. 2)



(c) Generated level for maximum frustration (Subject no. 1)



(d) Generated level for maximum frustration (Subject no. 2)



(e) Generated level for maximum challenge (Subject no. 1)



(f) Generated level for maximum challenge (Subject no. 2)

Figure 11.6: Example levels generated to maximize predicted engagement, frustration and challenge for two human players with different visual reaction features. Sub-figures (a), (c) and (e) are levels generated to maximize engagement, frustration and challenge, respectively for the first player. Sub-figures (b), (d) and (f) are example levels generated to, respectively, maximize engagement, frustration and challenge for the second player.

A level with slightly more challenging aspects has been generated for the second player where a smaller number of gaps has been chosen but with wider width, and enemies placed around collectable items.

Note that neither player behavioral data nor self-reported experience is available for the generated levels and, hence, there is no guarantee that the adaptation mechanism generates higher levels of engagement, challenge and frustration. However, the highly accurate ANN models built (above 80% accuracy) — that drive the generation of levels — suggest that higher values are most likely achieved for all emotional states. Moreover, the earlier experiments on dataset 1 (Section 11.2.1) — where the same exhaustive search approach was followed to generate personalized levels based on simpler player models — demonstrated that personalized levels are preferred by six out of ten human players.

11.4 Dataset 2: Advanced Parameterized Generator

The experiments conducted based on the data collected for dataset 1 showed promising results in terms of predicting the emotional states with a relatively high accuracy and adapting to player’s playing style and characteristics. However, only four controllable features were used in that dataset, three of which are related to gaps in the level. Moreover, no information about the content generated or player behavior has been recorded in those experiments which limited our ability of fully exploit the results obtained. Therefore, we run further experiments based on the richer models constructed on dataset 3. We use these models to evaluate the content generated by the GE-generator (Section 5.4) since the content space explored by this generator has not been constrained along very few dimensions. And we provide a thorough analysis based on fully recorded game sessions.

The methodology proposed in Section 10.2.2 has been employed to generate content that optimizes specific aspects of predicted player experience for a particular player given the player playing style in a test level.

The player is set to play *a level* and his playing style is recorded and used by GE to evaluate each individual design generated. Each individual is given fitness

according to the recorded player behavior and the values of its content features. The best individual found by GE is then visualized and set for the player to play.

11.4.1 AI Agents: Optimizing Player Experience for a Fixed Playing Style

AI agents have been employed to test the online adaptation mechanism in a similar manner to the one used in the experiments on dataset 1 (Section 11.2). AI agents have been employed because we wanted to test the efficiency of adaptation over time that requires the player to play-test a large number of levels.

A similar methodology to the one proposed in Section 11.2.1 to test the adaptation mechanism is followed; each AI agent is set to play a test level while its behavior is recorded. The evolution process is then started and GE is initialized with a random population of level designs, each individual is ranked according to the predicted player experience it provides (as predicted by the player experience models) given the player behavior in the test level and the content features extracted from the level design. The levels are then evolved and the best individual found by GE is visualized and the AI agent is set to play it. The same procedure is repeated for 100 rounds taking into account the player behavior in the previous round to evaluate the level designs in the current round.

Figures 11.7 and 11.8 presents the fitness values obtained for the best individual in each round when optimizing the level design to maximize engagement, frustration and challenge for the A* and Sergio's agent, respectively, while monitoring the models' prediction of the other emotional states than the one optimized.

The results presented in Table 11.2 show that, using the adaptation mechanism, we were able to construct levels for the two agents with high predictions across the three emotional states. The adaptation mechanism appears to maintain the same level of predicted engagement for the two agents compared to the values obtained for frustration and challenge as can be observed from the standard deviations values presented in Table 11.2.

By examining the gameplay data recorded while optimizing the levels, we observed that most of the low prediction values obtained when predicting challenge are the cause of outliers in the time spent playing (very short or very long game-

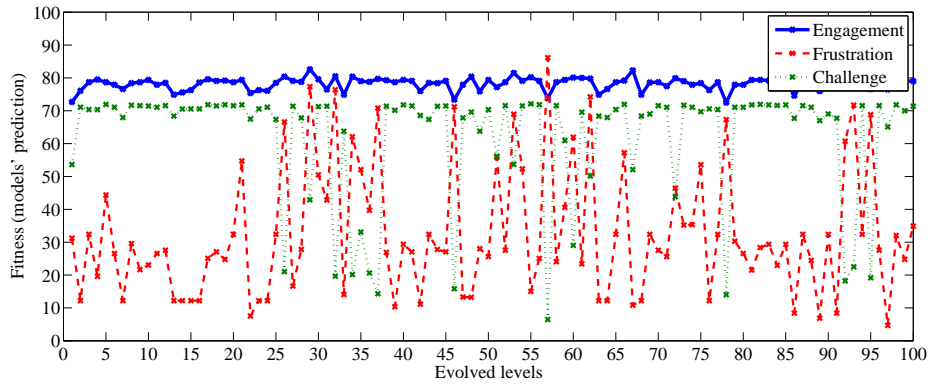
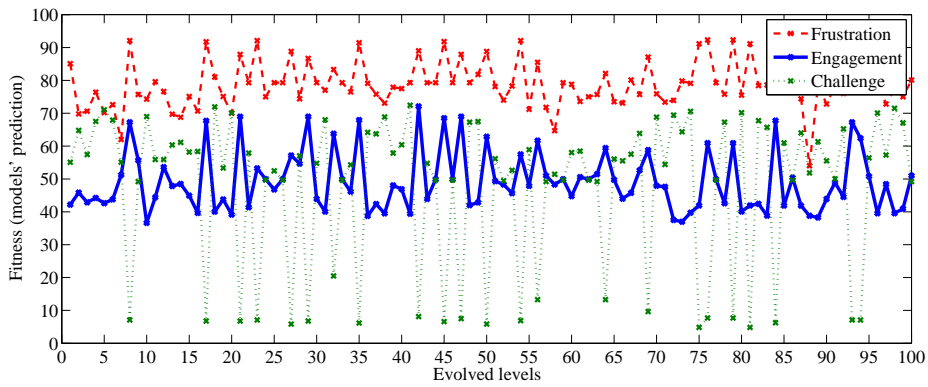
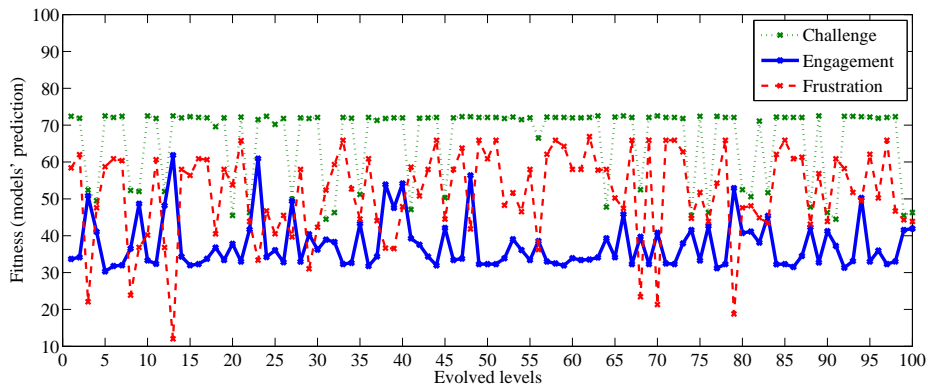
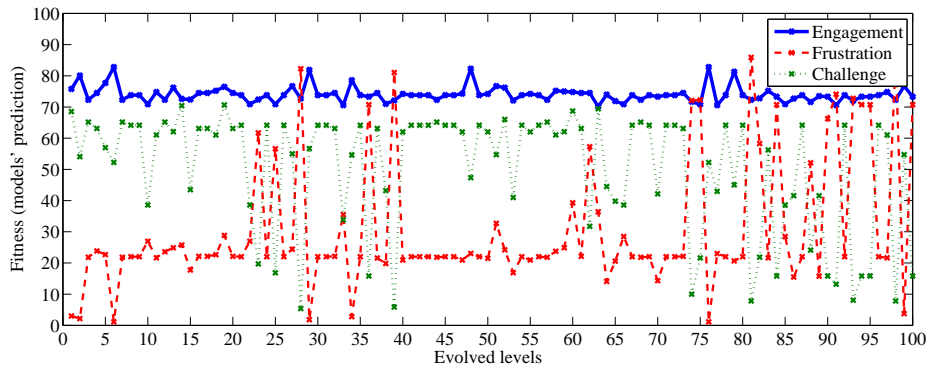
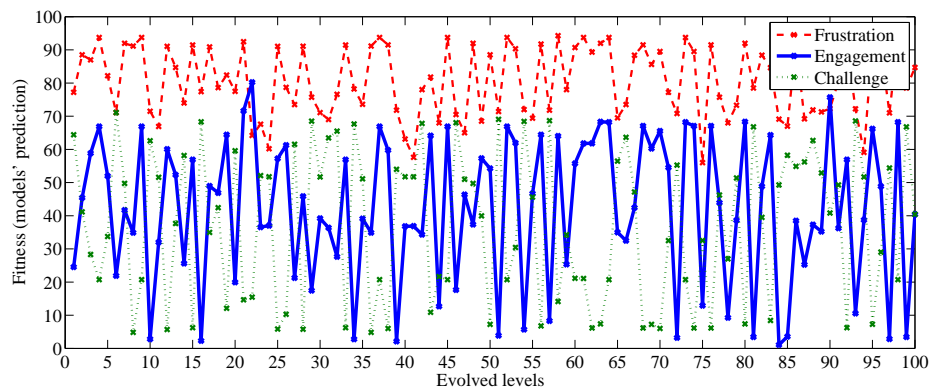
(a) Optimized predicted *engagement* for the A* agent.(b) Optimized predicted *frustration* for the A* agent.(c) Optimized predicted *challenge* for the A* agent.

Figure 11.7: The fitness function for the optimized levels evolved for the A* agent for each player experience state while monitoring the models' prediction of the other states.

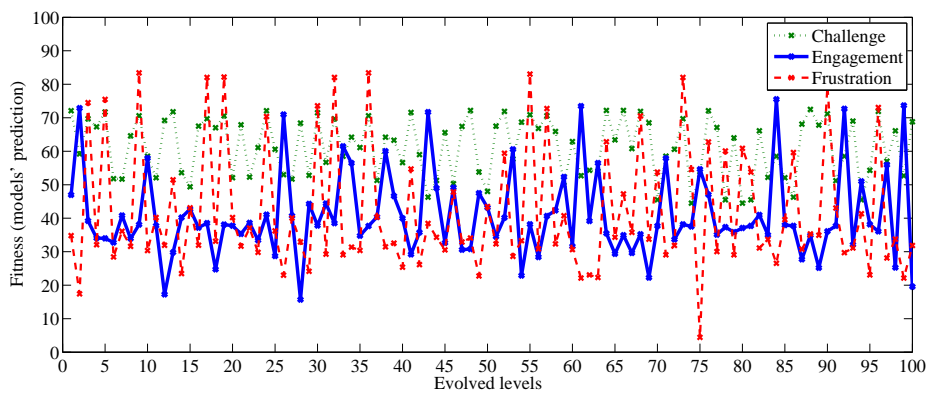
11.4. DATASET 2: ADVANCED PARAMETERIZED GENERATOR



(a) Optimized predicted *engagement* for Sergio agent.



(b) Optimized predicted *frustration* for Sergio agent.



(c) Optimized predicted *challenge* for Sergio agent.

Figure 11.8: The fitness function for the optimized levels evolved for Sergio's for each player experience state while monitoring the models' prediction of the other states.

Table 11.2: Average fitness values obtained for the evolved levels across the three emotional states for each agent along with the standard deviation values.

	<i>Engagement</i>	<i>Frustration</i>	<i>Challenge</i>
A* agent			
<i>Avg</i>	78.23% \pm 1.88	79.06% \pm 7.36	66.2% \pm 10.23
Sergio's agent			
<i>Avg</i>	74.01% \pm 2.56	80.59% \pm 10.58	61.25% \pm 8.75

play session). These outliers are the result of dying very early in the level or being stuck in a dead end as a result of the agents incapacity to backtrack which will eventually lead to losing the game by reaching the maximum session time allowed.

The same argument holds for predicted frustration but with different effects; while short gameplay sessions resulted in a high level of frustration, being stuck in a dead end until the session time expires resulted in low prediction values of frustration. However, the length of the gameplay session appears to have less influence of the prediction of engagement.

Levels with varying content parameters have been evolved for each agent across the three emotional states as can be seen from Figure 11.9 which presents the best levels generated to optimize players' experiences. The levels can be analyzed according to the six content features optimized to maximize specific experience. Levels with different structure have been observed when optimizing engagement for the two agents; while a flat level with some coins have been evolved for Sergio's agent, the most engaging level for the A* agent contains gaps and enemies with no coins. The best levels evolved for optimizing frustration, on the other hand, exhibit more similar structure with both of them having the same number of gaps while differing in the number and placement of enemies; a small number of enemies scattered around gaps have been generated for the A* agent, while more enemies randomly placed around the level have been evolved for the most frustrating level for Sergio's agent. A slightly more challenging level with more gaps has been evolved for Sergio's agent than the one generated for the A* agent.

Figure 11.10 presents statistics for the six content features used to evaluate

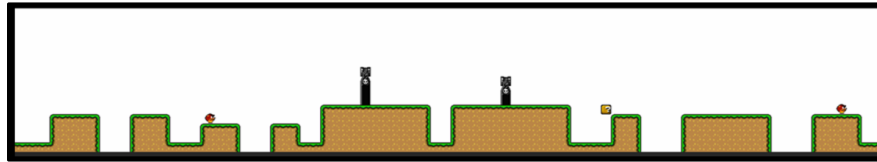
game content extracted from the 100 levels generated to optimize the prediction of the three experience states for the two agents. All values are normalized to the range $[0,1]$ using the standard max-min normalization. As can be seen from the figure, the most engaging levels evolved for optimizing engagement for the two agents vary in the number and the width of gaps and the placement of enemies; the levels evolved for the A* agent contain more and wider gaps with enemies mostly placed around gaps and blocks compared to less and narrower gaps with enemies scattered randomly around the levels observed for the levels optimized for Sergio’s agent. The opposite observations can be derived from the levels optimized for challenge. It hence appears that the approach was able to recognize the differences in the playing styles and consequently evolve levels with different characteristics to optimize particular player experience states. On the other hand, similar average values for the content features have been observed from the maximum frustrating levels for the two agents.

11.4.2 Statistical Analysis

We performed statistical tests to further investigate the results obtained, the method’s ability to recognize and adapt to a particular playing style and the relationship between the three player experience states.

The first test has been conducted to test the ability of the adaptation approach to recognize a particular playing style and adapt accordingly by testing for significant differences in the prediction accuracies obtained from the best levels evolved for each agent. The statistical test showed that the adaptation mechanism evolved significantly more engaging and challenging levels for the A* agent compared to the ones generated for Sergio’s agent while no significant difference has been observed for the levels evolved to maximize frustration for the two agents (significance is determined by $p < 0.01$). It, hence, appears that it is easier to generate engaging and challenging levels for a player with an expert playing behavior than for a player with an average playing performance.

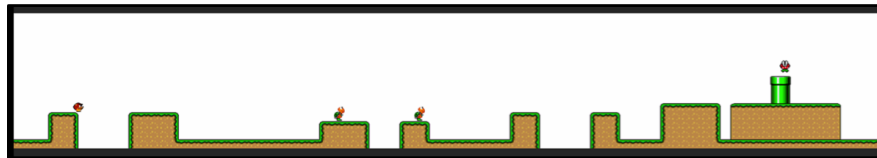
To analyze the dependences between the three reported states, the effects of optimizing one state on the prediction of the other states and the relationship between them, we performed a test to check for correlations between the observed



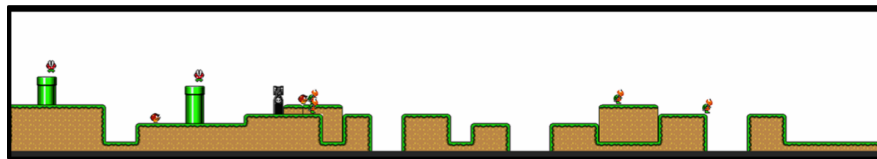
(a) Evolved level for maximum engagement for the A* agent



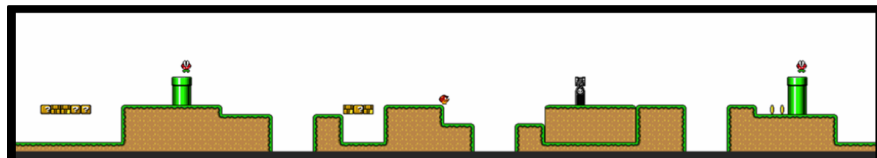
(b) Evolved level for maximum engagement for Sergio's agent



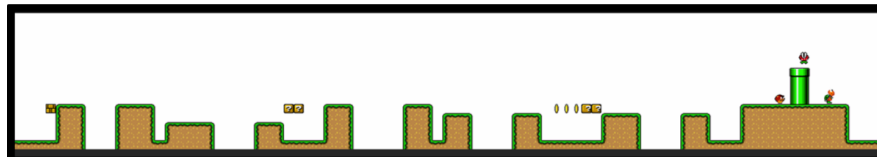
(c) Evolved level for maximum frustration for the A* agent



(d) Evolved level for maximum frustration for Sergio's agent



(e) Evolved level for maximum challenge for A* agent



(f) Evolved level for maximum challenge for Sergio's agent

Figure 11.9: The best levels evolved to maximize predicted engagement, frustration and challenge for the two agents. Sub-figures (a), (c) and (e) are levels evolved to maximize engagement, frustration and challenge, respectively, for the A* agent. Sub-figures (b), (d) and (f) are best levels generated to, respectively, maximize engagement, frustration and challenge for the Sergio's agent.

11.4. DATASET 2: ADVANCED PARAMETERIZED GENERATOR

Table 11.3: Correlation coefficient values obtained between the predictions of engagement (E), frustration (F) and challenge (C) when optimizing each of these player experience states (columns) while monitoring the prediction of the other emotional states (rows). Significant values ($p - value < 0.01$) appear in bold.

	Optimized player states					
	A* agent			Sergio's agent		
	<i>E</i>	<i>F</i>	<i>C</i>	<i>E</i>	<i>F</i>	<i>C</i>
<i>E</i>		0.62	-0.34		0.613	-0.4
<i>F</i>	0.23		0.31	-0.45		0.36
<i>C</i>	0.04	-0.78		0.12	-0.67	

predictions of the non-optimized experience states in each experiment. For example, in the experiment conducted to optimize engagement, the correlation between the models' prediction of frustration and challenge has been calculated. The correlation effects obtained showed that optimizing one state resulted in negatively correlated predictions of the other two experience states for the two agents.

To further investigate the intra-correlations between the three player states, we calculated the correlations between the predictions of an optimized state and the predictions of the other non-optimized experience states. The results presented in Table 11.3 show that for the two agents, evolving engaging levels resulted in levels that are also challenging while generating challenging levels were found to generate levels that are less engaging for the two agents.

On the other hand, different effects have been observed between the predictions of engagement and frustration: while evolving engaging levels were also found to generate frustrating levels—as evidenced by the predictions of these two emotional states to be positively correlated for the A* agent—, these emotional states were found to be highly and negatively correlated for Sergio's agent. Evolving frustrating levels, on the other hand, resulted in levels that are also engaging for the two agents.

As for the relationship between predicted frustration and challenge, the correlation analysis showed that when optimizing predicted frustration, less challenging levels are generated for the two agents. However, predicted challenge and frustration were found to be highly and positively correlated when evolving challenging levels for the two agents.

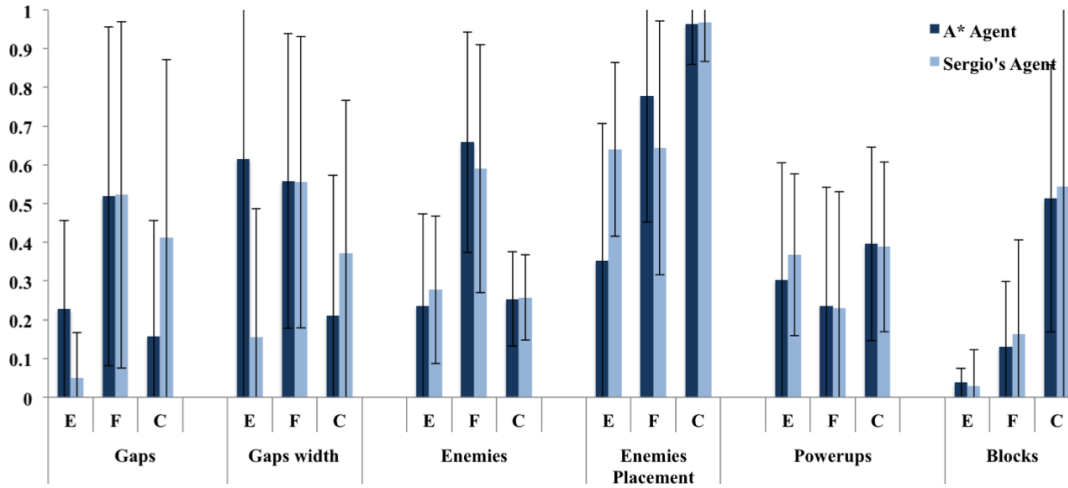


Figure 11.10: Average and standard deviation values of six statistical content features extracted from 100 levels evolved to optimize predicted engagement (E), frustration (F) and challenge (C) for the two agents. Enemies placement $E_p = 0$ when $P_g = 80\%$, $E_p = 0.5$ when $P_x = 80\%$ and $E_p = 1$ when $P_r = 80\%$.

This variety of relationships observed between the predictions of the three player experience states across the agents reflects the method’s ability to recognize different playing styles and evolve levels with different characteristics that adapt to a specific style.

11.4.3 Discussion

The analysis conducted to test for the adaptation method’s validity and the dependencies among the predictions of the experience states showed that it is easier to optimize engagement and challenge for an expert player than for an average performing player.

The results obtained revealed interesting relationships among the predictions of the emotional states. Engaging levels were found to be challenging, in general, while challenging levels are not necessarily engaging. The same observation has been obtained between predicted frustration and challenge; the features that play a role in generating challenging levels appear to also have a positive effect on the perceived frustration, while the influence of the features positively affecting perceived frustration has been observed to be negative on predicted challenge.

The right amount of frustration should be present in a level for it to be engaging, this also depends on the playing style; while engaging levels are also found to be frustrating for some players, other players might enjoy less frustrating levels.

Note that these observations are valid in this specific setup where AI agents are employed, and further analysis with human players is required if we are to draw more general conclusions.

11.5 Summary

The chapter introduced an approach for evaluating the adaptation mechanisms presented in the previous chapters. To this end, previously constructed player experience models that map game content to reported player experience (engagement, frustration and challenge) were utilized as fitness functions. Two AI agents (and in some cases human players) with different playing styles were used to test the adaptation approach. A set of experiments have been conducted on the three datasets collected and a thorough analysis of the results is presented. The results obtained illustrate the methods' ability to recognize differences in playing styles and generate content accordingly. The analysis performed revealed interesting relationships between the three emotional states and showed promising potential for the approach. However, most of the analysis is based on the behavior of two AI agents and we believe that further investigations with more human players would allow us to draw a clearer picture for the suitability of experience-driven PCG in platform games.

12

Conclusions

In this dissertation we have presented a data-driven framework for closing the affective loop in games based on three datasets comprised different number of players and variety of features of game content and player behavior. By implementing this framework, we aimed at addressing the two main research objectives of this dissertation, more specifically: constructing accurate estimators of player experience while interacting with a game and building a data-driven approach to effectively close the affective loop in games. Infinite Mario Bros, a clone of the well-known game Super Mario Bros, is employed as a testbed game for our experiments.

Given our objectives, we devised three content generators for the game varying along different dimensions of content coverage. In two of the generators, an integer vector representation of the content space is used to control content creation and generate variety of content. Grammatical formulation of the possible content is utilized in the third generator and grammatical evolution is employed to explore the content space. An expressivity analysis framework is proposed to analyze the content space covered by each generator along six expressive dimensions defined to highlight the dissimilarity among these generators, a histogram-based visualization method is proposed to assess the expressive range of each generator.

These three generators are used to crowd-source three datasets from hundreds of players. The focus in each dataset is given to a set of game content and player behavior data allowing thorough analysis of different aspects that contribute to player experience in games.

According to our assumption, there is an unknown function between game content and player experience that can be approximated using machine learning techniques. Based on this assumption, we construct models of player experience based on different feature types: game content features extracted from the levels played, gameplay features capturing the characteristics of playing style, visual cues extracted from video recordings of gameplay sessions and players' experience states as reported via player answers to questionnaires.

Once the data has been collected, we introduced the player experience modeling framework. The framework consists of three main steps: feature extraction for identifying the pool of features that represents the in-game interaction, feature selection for reducing the size of the feature space and consequently increasing the modeling accuracy and ease the analysis, and model optimization for adjusting the model topology for the best approximation of the constructed function. Neuroevolutionary preference learning is used as a modeling mechanism while sequential forward selection is employed to select the subsets of relevant features for predicting reported affects.

In order to allow a rich representation of the in-game interaction, along with the variety of types of the features collected, we introduced different methods for feature extraction and representation. Direct features, calculated as frequencies of items and events, as well as sequential patterns of content and behavior, extracted using data mining methods, have been considered. Given the wealth of the data collected, we also presented unimodal and bimodal features that allow capturing and analyzing the cause-effect relationships between the different modalities considered.

Accurate models of player experience have been constructed based on the features extracted from each dataset. In each set of modeling experiments, the focus is given to analyzing the relationship between players' reported affective states and the specific features collected for each specific dataset. The results have shown that player experience can be predicted with high accuracy from information about the interaction with the game. Linear and non-linear relationships have been analyzed revealing interesting correlations and accommodating for a better understanding of the gameplay experience.

Based on the models constructed, an online adaptation approach is presented.

An experiment is conducted to investigate the best frequency that should be used to adapt game content. This is done by segmenting the game sessions into smaller parts while monitoring the changes in the models' prediction accuracies. The chosen size is one that is short enough to allow acceptable adaptation frequency, yet long enough to allow reliable gameplay experience measurement and to elicit player emotions. Online generation of game content is achieved by utilizing the constructed player experience models as a measure of content quality. The content space, as represented by a vector of up to six content features, is explored using an exhaustive search approach that chooses the combination of feature values that optimized particular player experience by maximizing the models' output.

The exhaustive search adaptation method is appropriate due to the relatively small size of the search space explored by two of the content generators. As the search space becomes larger in our third generator, exhaustive search becomes infeasible. Therefore, more sophisticated and effective online adaptation procedure is designed to deal with larger search spaces. To this end, an evolutionary method is constructed and player experience models are integrated in the evolutionary process as fitness functions. This way, generating personalized content becomes part of the content evolution process by ranking each design configuration generated according to its appeal to a particular player.

The two adaptation approaches are tested using two AI agents of different gameplay characteristics. This permits evaluating the methods ability to recognize differences in playing style and generate levels accordingly. Moreover, it allows analyzing the methods' performance over time. The results demonstrated the method's effectiveness in capturing the dissimilarity between the two agents and in generating personalized content. The adaptation framework is further tested with human players where the observations showed that 60% of the players preferred personalized levels over randomly generated levels when the exhaustive search method is used for adaptation.

12.1 Contributions

This section summarized this dissertation’s main contributions to advance the state-of-the-art of game AI and affective computing. We claim the results in this dissertation to be useful, and to some extent applicable, to the fields of affect recognition, human-computer interaction, game design and procedural content generation. More specifically, this dissertation has contributed the following:

- We introduced, through the use of grammatical evolution, a new method for creating game content for a 2D platform game. The method is able to generate infinite variation of playable content which showcases an interesting and useful use of grammatical evolution in games.
- We introduced a framework for analyzing and comparing the expressive ranges of different content generators. The expressivity measures presented can be used to analyze other content generators than the ones presented in this dissertation, and we believe the expressivity analysis framework is relevant for evaluating generators of other 2D games. The framework can also be a great assistive tool for game designers to visualize the implications of their design choices and to analyze the strength and weakness of their generators.
- We crowd-sourced three datasets from hundreds of players each with a different set of features. We plan to publish these datasets so that they will be available for other researchers to conduct different experiment and show different perspectives of which these datasets can be used in research. We believe these datasets to be useful for testing different machine learning, AI and data mining techniques as well as being valuable for researchers in the field of affect recognition and human-computer interaction.
- We identified different feature sets for capturing a variety of aspects of player interaction with a 2D platform game. More specifically, we collected features about game content, players’ actions and visual reactions to gameplay events as well as players’ reported preferences of different affective states. Some of these features are game specific, such as the content features, while

others can be generalized to other games from the same genre such as the player behavior features. Yet the third feature category, the visual cues feature, appear to be linked to generic game events such as killing an enemy, winning or losing the game, and, therefore, they appear to be applicable to most computer games.

- We introduced two feature representations for accommodating for the spatial and temporal order of events or actions, and we implemented data mining methods for extracting content and/or behavioral frequent patterns. The methods followed for feature representation and extraction can be applied to any other similar problem where detailed data is available.
- We presented the successful use of features from one modality and bimodal features for capturing player experience. Bimodal features are extracted by fusing actions and content events on the sequences level and by relating visual reactions to game events. Both of these types of features showed promising results in terms of modeling player interaction with the game. The experiments presented can be used as initial success points for a more thorough analysis of these features as well as an inspiration for the use of dissimilar types of features from other modalities.
- Based on datasets collected, we established accurate quantitative measures of player experience. The framework proposed for model construction in terms of feature extraction, selection and model optimization could be easily scaled to other games from the same genre or other genres of games.
- The thorough experiments conducted revealed interesting observations about the interaction between the player and the game and the relationship between game content, player behavior and reported affects. The framework proposed can be viewed as a mean for quantifying aesthetics in 2D platform games and in that sense, the results can be used by game designers to better understand the possible experience their design would trigger when played. The models constructed can ultimately be useful for designers to assess the design process as they can inform the designer about the set of game level

features (such as the number of enemies and gaps) that can maximize (or indeed minimize) the modeled player state for a particular player.

- The player experience models constructed were utilized for measuring content quality, and hence, enabled the creation of personalized content. Online content creation is achieved through employing the models in the content creation loop establishing an efficient adaptation mechanism. Searching the content space for optimal content is realized by implementing two search methods: exhaustive search and an evolutionary algorithm. Combining player experience modeling with parameter optimization for online creation of personalized content proved to be an efficient method for fulfilling the requirements of quantitatively closing the affective loop in games.
- The adaptation results demonstrated the methods' success in recognizing different playing styles and in generating content that maximizes aspects of player experience. We consider the adaptation framework to be of interest for industrial game development since it provides a complete and promising framework for sensing player affects, modeling player experience and adapting game content, which is a direction that is receiving increasing attention in game industry recently (Yannakakis [2012]).

12.2 Limitations and Opportunities

The limitations of the proposed methodology, as appeared throughout the experiments and analysis of this dissertation, are summarized in this section. They are categorized into limitations concerning the tools used and limitations considering the methodology proposed. Suggestions and ideas for overcoming such drawbacks are discussed and provide the ground for future investigations.

12.2.1 Tools

There are a number of limitations inherent in the player experience modeling approach followed. The feature selection method provides an efficient mechanism

for selecting relevant features when the size of the search space is rather small. This method, however, results in a suboptimal subset of features when searching a large space. Moreover, when searching for the relevant subset of features, we are mostly interested in the minimal independent feature set and this is not guaranteed by SFS. Automatic feature selection is an essential step when constructing the experience models since selecting the correct subset of features may have a great impact on the prediction accuracy obtained. Improving the global search abilities of the feature selection process is one way to improve the prediction accuracy and the interpretation of the models. Algorithms relying on meta-heuristic search such as genetic-based feature selection ([Martínez and Yannakakis \[2010\]](#)), or Monte Carlo Tree Search techniques such as Future UCT Selection ([Gaudel et al. \[2010\]](#)), can allow the detection of more appropriate feature subsets. Other techniques such as deep learning can be investigated to extract new types of features ([Bengio \[2009\]](#)).

The other limitation concerns the use of neuroevolutionary preference learning as a modeling approach which suffers from the limited transparency power of the experience models. By using neuroevolutionary preference learning, we gain the advantage of universal approximation capacity for constructing accurate non-linear models, but we lose the ability of easily analyzing the cause-effect relationships between the features selected and the models' prediction of each player state. Thus, exploiting the use of more expressive model representations such as decision trees, Bayesian networks or fuzzy neural networks for modeling player experience constitutes a future direction. Moreover, ANNs have shown a great potential and accurate modeling results in similar studies ([Martinez et al. \[2009\]](#); [Pedersen et al. \[2009\]](#); [Yannakakis \[2009b,c\]](#)). However, better results might be obtained by other machine learning techniques such as support vector machines.

Our adaptation mechanism was limited to content parameters adjustment via the use of search methods. Although interesting and promising results were obtained in terms of generating personalized content, we believe there are a number of improvements that can be applied to increase the adaptation efficiency. Altering the structure of the models is one promising direction. For instance, NeuroEvolution of Augmenting Topologies (NEAT) ([Stanley and Miikkulainen \[2002\]](#)) can be used to evolve personalized network topologies which can be up-

dated as players progress through the game. Another interesting approach is to include game mechanics in the adaptation loop and consequently evolve personalized or player-centered mechanics building on the experience models presented in this thesis and the work done by Cook [2012]; Cook and Colton [2011] on inventing mechanics.

12.2.2 Methodology

The experiment conducted on utilizing grammatical evolution for creating content for the game showed that wide variety of infinite content can be generated. The expressivity analysis, however, showed that the GE-generator is unable to generate levels with high density due to the height constraint defined in the grammar forcing the generated chunks to be placed within a predefined height limit to ensure playability. One possible solution is to define a constraint-free grammar and play-test the generated levels to check for the playability. This can be done automatically by exploiting the use of AI agents that pass through the levels and check for possible path from the start to the end, and/or check whether all chunks generated are reachable. Another solution is to adopt context-sensitive grammar such as attribute grammars to control the parameter values of the solutions as they are being generated during the mapping process (O'Neill et al. [2004]). Moreover, in this dissertation, we focused on parameterized and grammar-based representation of game content. Exploring other forms, such as direct representation, constitutes another interesting direction.

In the expressivity analysis framework, we introduced a number of dissimilar measures of content variability. None of these measures, however, rank the content from the player's perspective or based on the experience it provides. A future direction includes defining more in-depth expressivity measures along which content quality can be analyzed and compared. Designers' knowledge or the player experience models constructed in this dissertation can be utilized as content quality measures to rank the content generated according to the elicited gameplay experience.

In the experiments presented in this thesis, we focused on specific set of content and behavioral features. Other parameters such as personality traits

(Spronck et al. [2012]; van Lankveld et al. [2011]), experience with games, gender, ethnicity, smaller facial gestures (e.g. small lip movements and other personalized features), as well as other people’s presence (and its affect on gameplay and/or visual reactions) might also be important predictors of player experience. Future research could take into account a series of such additional factors in order to maximize control over player states, increase context influence and encourage personalization in game playing.

The models constructed in this dissertation are purely data-driven. It would be interesting to validate the methodology proposed and its findings with designers’ knowledge and expertise of what makes a level engaging, frustrating or challenging, and validate the extent to which the observed findings add to our knowledge about the creative process of game design.

For the feature extraction experiments presented in this dissertation, we used only sequences of length three. Patterns of this length could be rather too small to draw general conclusions. Frequent sequences of longer length have been investigated; although these sequences are more expressive, a performance drop has been observed. Longer sequences tend to capture more specific patterns across multiple modalities of player input in which we expect larger data variation due to variant playing styles. A solution might be to cluster the resulting sequences and construct models for each cluster, or consider sequences of different length as inputs to ANN models. A step towards clustering players’ behavior based on sequence patterns of actions has already been taken and preliminary results indicate the promise of the approach. One could also investigate the use of other sequence prediction techniques such as Hidden Markov Model to classify the resulted sequences and to extract sequential patterns.

12.2.3 Adaptation Framework

The personalized Super Mario Bros levels generated show that the experience-driven procedural content generation framework (Yannakakis and Togelius [2011]) can be realized and the affective loop can be closed in games. The adaptation framework followed provides a nouvelle approach for control and adaptation in computer games. The adaptation methodology proposed, however, needs to be

12.2. LIMITATIONS AND OPPORTUNITIES

further validated with more human players in actual gameplay sessions. The results presented based on studies on a small group of human players showcase that the adaptation framework is effective in generating levels which are preferred by the majority of players but further investigation is required for more accurate and generalizable results.

The adaptation framework and the analysis presented focus on optimizing specific predicted player experience states for a particular playing style. Future directions include testing the method's ability to optimize more than one aspect of player experience. Multi-objective optimization techniques can be used to optimize, for example, engagement and challenge while minimizing frustration. This will help us explore new dimensions of player experience and ultimately generate content that maximizes some dimensions while minimizing others to optimize player experience.

AI agents have been employed to test the adaptation methodology. Although the two agents have been chosen so that they exhibit varying playing characteristics, the main drawback of this approach lies in their lack of yielding human-like behavior. The obvious better alternative is to use AI agents that are trained to imitate human playing style (J. Togelius and Shake [2011]; Ortega et al. [2012]). This allows more accurate examination of the adaptation performance. AI agents that adapt their behavior and learn over time can ultimately be used to simulate the human learning process.

Promising results have been obtained for the experiments conducted utilizing the GE-generator for creating content. The approach showed an ability to capture specific playing style and generate adapted content accordingly. However, the analysis showed that there is an important limitation imposed in the approach followed that concerns the use of player experience models constructed based on levels generated by a content generator with a different expressivity range (see Chapter 6). While the basic parameterized generator produces content by varying the values of six content features with all other dimensions being fixed, the GE-based generator explores the content space without such limitations. It is, therefore, important to conduct further experiments to investigate the effect of the content features on players' judgment since we anticipate that new factors of level design will play a role in human experience.

While the experiments conducted focused on defining the frequency of adaptation and how to adapt game content, there is still one important aspect of content adaptation that has not been explored, more specifically, whether or not we should apply adaptation. While personalizing game content can be appreciated by some players, there are others who dislike adaptation and prefer static content which they can replay and master. The adaptation mechanism should be able to recognizing player's preferences and if adaptation should be applied. The framework presented in this dissertation provides a partial solution to this problem by continuously sensing player behavior and adjust the content accordingly. However, we will need to empirically evaluate whether the approach is indeed capable of capturing such preferences.

The adaptation framework is implemented for research purposes but playable demonstrations have been published online where players can play the game and choose a particular experience to optimize (Shaker [2011]). A new personalized level is then generated based on her playing style. Constructing a complete commercial standard demonstrator of the framework constitutes an important future direction sine this will highlight and emphasize the potential of the methodology.

12.3 Extensibility

Accurate estimators of player experience and a robust adaptation framework were designed, implemented and analyzed in this dissertation. In this section, we discuss the potential of the methodology in other games from the platform genre or other genres of games.

12.3.1 Player Experience Modeling

As mentioned earlier in Chapter 4, Super Mario Bros *defines*, more or less, the platform game genre. We argue that the approach presented has a great potential to be applied successfully to other 2D platform games since most of the gameplay features defined can be easily generalized to capture playing styles in a variety of other games. Moreover, the generality of the selected game content patterns

allows the use of them to design and analyze other levels with similar characteristics, yet different graphical representations. Also, the applicability of the visual reaction features—which proven to be efficient predictors of player’s affect— appears to be a trivial process since the extraction of these features depends on key performance events of the context (such as indicators of losing and winning).

We also believe that our approach can generalize well to other game genres, at least first- or third-person action games, since the features we define have straightforward analogies in those games. Not only do many such games have platform elements, but also in a typical FPS like *Halo* (343 Industries [2012]) concepts such as average movement speed, speed of progress, number of shots fired, entropy of object placement, and number and concentration of enemies/items/obstacles are clearly defined and can be assumed to have impact on player experience.

The proposed approach and the analysis presented can also be used as an assistive tool in a mixed-initiative level design process (Smith et al. [2010]). A level can be crafted by a human designer and models constructed from game content and reported player experience can be used to encourage the designer to include or modify features or patterns based on the experience the designer wishes to provide.

12.3.2 Adaptation Methodology

The adaptation methodology presented can be generalized to any other game once player experience has been empirically measured. Exhaustive search approach can be applied within small dimensional search spaces whereas a similar meta-heuristic global search method can be implemented to deal with larger spaces.

12.4 Summary

This dissertation has presented a number of methods for automatically generating personalized game content for a 2D platform game. Moreover, it introduced an efficient method for quantitatively modeling player experience through the use of various feature types crowd-sourced from different players of such games. These

models are further used to assess content quality in an online adaptation framework that is designed so as to close the affective loop in games. The methodology proposed showed promising results and demonstrated a potential for extensibility to other games.

Bibliography

- 343 Industries, 2012. Halo, Microsoft Studios. [229](#)
- Activision, 1982. Pitfall!, Activision. [72](#)
- T. Adams, 2006. Dwarf Fortress, Bay 12 Games. [34](#), [39](#)
- R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th international conference on Very Large Data Bases, VLDB*, volume 1215, pages 487–499, 1994. [65](#), [66](#), [68](#), [139](#)
- R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14. IEEE, 1995. [65](#)
- S. Almeida, A. Veloso, L. Roque, and O. Mealha. The eyes and games: A survey of visual attention and eye tracking input in video games. In *SBGames: X Brazilian Symposium on Computer Games and Digital Entertainment - Arts & Design Track*, , Salvador, Brazil, 2011. [24](#)
- G. Andrade, G. Ramalho, H. Santana, and V. Corruble. Automatic computer game balancing: a reinforcement learning approach. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1111–1112. ACM, 2005a. [43](#)
- Gustavo Andrade, Geber Ramalho, Hugo Santana, and Vincent Corruble. Challenge-sensitive action selection: an application to game balancing. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 194–200, Washington, DC, USA, 2005b. [43](#)

- D. Ashlock, C. Lee, and C. McGuinness. Search-based procedural generation of maze-like levels. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):260–273, 2011. [40](#)
- D.A. Ashlock, T.W. Manikas, and K. Ashenayi. Evolving a diverse collection of robot path planning problems. In *IEEE Congress on Evolutionary Computation. CEC*, pages 1837–1844. IEEE, 2006. [40](#)
- S. Asteriadis, P. Tzouveli, K. Karpouzis, and S. Kollias. Estimation of behavioral user state based on eye gaze and head pose - application in an e-learning environment. *Multimedia Tools and Applications, Springer*, 41(3):469 – 493, 2009. [22](#), [145](#)
- P. Avery, J. Togelius, E. Alistar, and R.P. van Leeuwen. Computational intelligence and tower defence games. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1084–1091. IEEE, 2011. [48](#)
- A. Bahamonde, G.F. Bayón, J. Díez, J.R. Quevedo, O. Luaces, J.J. Del Coz, J. Alonso, and F. Goyache. Feature subset selection for learning preferences: a case study. In *Proceedings of the twenty-first international conference on Machine learning*, page 7. ACM, 2004. [62](#)
- Ryan S. J. d. Baker, Sidney K. D’Mello, Ma.Mercedes T. Rodrigo, and Arthur C. Graesser. Better to be frustrated than bored: The incidence, persistence, and impact of learners’ cognitive-affective states during interactions with three different computer-based learning environments. *Intonation Journal on Human-Computer Studies*, 68(4):223–241, 2010. [126](#)
- C. Bartneck. Integrating the occ model of emotions in embodied characters. In *Workshop on Virtual Conversational Characters*. Citeseer, 2002. [19](#)
- Y. Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009. [224](#)
- P. Bentley. *Evolutionary design by computers*, volume 1. Morgan Kaufmann, 1999. [56](#)

BIBLIOGRAPHY

- P.J. Bentley. Exploring component-based representations-the secret of creativity by evolution. In *Fourth International Conference on Adaptive Computing in Design and Manufacture (ACDM 2000)*, page 161, 2000. 56
- C.M. Bishop. Neural networks for pattern recognition. 1995. 60
- S. Björk and J. Holopainen. *Patterns in game design*. Cengage Learning, 2005. 30
- Blizzard Entertainment and Mass Media, 1998. StarCraft, Blizzard Entertainment and Nintendo. 42
- Blizzard North, 1997. Diablo, Blizzard Entertainment, Ubisoft and Electronic Arts. 34, 39
- S. Bojarski and C. Congdon. Realm: A rule-based evolutionary computation agent that learns to play mario. In *IEEE Symposium Computational Intelligence and Games (CIG)*, pages 83–90, 2010. 78
- M. Booth. The ai systems of left 4 dead. In *Keynote, Fifth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE09)*, 2009. 36, 38
- D. Braben and I. Bell. 33
- M.M. Bradley and P.J. Lang. Measuring emotion: the self-assessment manikin and the semantic differential. *Journal of behavior therapy and experimental psychiatry*, 25(1):49–59, 1994. xiii, 26
- M.E. Bratman. *Intention, plans, and practical reason*. Cambridge University Press, 1999. 20
- Brøderbund, Red Orb, Ubisoft, Pipeworks, and Gameloft, 1989. Prince of Persia, Brøderbund, TLC, Mattel, Ubisoft and SCEJ. 72
- C. Browne. Yavalath, 2007. URL <http://www.cameronius.com/games/yavalath/>. 39
- C. Browne. Elegance in game design. *IEEE Transactions on Computational Intelligence and AI in Games*, 2012. 32

- C. Browne and F. Maire. Feature Analysis for Modeling Game Content Quality. In *IEEE Conference on Computational Intelligence and AI in Games (CIG)*, pages 1–16, 2010a. [41](#)
- C. Browne and F. Maire. Evolutionary game design. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):1–16, 2010b. [39](#)
- C. Browne, GN Yannakakis, and S. Colton. Guest editorial: Special issue on computational aesthetics in games. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(3):149–151, 2012. [29](#), [31](#), [32](#)
- Fabio Buttussi, Luca Chittaro, Roberto Ranon, and Alessandro Verona. Adaptation of graphics and gameplay in fitness games by exploiting motion and physiological sensors. In *Smart Graphics, 7th International Symposium, SG 2007, Kyoto, Japan, June 25-27, Proceedings*, Lecture Notes in Computer Science, pages 85–96. Springer, 2007. [24](#)
- E. Byrne. *Game level design*. Delmar Thomson Learning, 2005. [126](#)
- J. Byrne, M. Fenton, E. Hemberg, J. McDermott, M. O’Neill, E. Shotton, and C. Nally. Combining structural analysis and multi-objective criteria for evolutionary architectural design. *Applications of Evolutionary Computation*, pages 204–213, 2011. [57](#)
- G. Calleja. *In-Game: From immersion to incorporation*. MIT Press, 2011. [2](#), [130](#)
- W.B. Cannon. The james-lange theory of emotions: A critical examination and an alternative theory. *The American Journal of Psychology*, 39(1/4):106–124, 1927. [18](#)
- W.B. Cannon. Again the james-lange and the thalamic theories of emotion. *Psychological Review*, 38(4):281, 1931. [18](#)
- L. Cardamone, G. Yannakakis, J. Togelius, and P. Lanzi. Evolving interesting maps for a first person shooter. *Applications of Evolutionary Computation*, pages 63–72, 2011a. [40](#), [42](#)

BIBLIOGRAPHY

- Luigi Cardamone, Daniele Loiacono, and Pier Luca Lanzi. Interactive evolution for the procedural generation of tracks in a high-end racing game. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 395–402. ACM, 2011b. [40](#), [42](#)
- G. Caridakis, S. Asteriadis, and K. Karpouzis. User modeling via gesture and head pose expressivity features. pages 19–24. 5th International Workshop on Semantic Media Adaptation and Personalization, (SMAP 2010), Limassol, Cyprus, 2010. [145](#), [146](#)
- G. Castellano, A. Pereira, I. Leite, A. Paiva, and P. W. McOwan. Detecting user engagement with a robot companion using task and social interaction-based features. In *Proceedings of the 2009 international conference on Multimodal interfaces*, pages 119–126, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-772-1. doi: <http://doi.acm.org/10.1145/1647314.1647336>. [22](#)
- Guillaume Chanel, Cyril Rebetez, Mireille Bétrancourt, and Thierry Pun. Boredom, engagement and anxiety as indicators for adaptation to difficulty in games. In *Proceedings of the 12th international conference on Entertainment and media in the ubiquitous era*, pages 13–17, New York, NY, USA, 2008. ACM. [126](#)
- D. Charles and M. Black. Dynamic player modeling: A framework for player-centered digital games. In *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, pages 29–35, 2004. [xiii](#), [47](#)
- J. Chen. Flow in games (and everything else). *Communications of the ACM*, 50(4):31–34, 2007. [21](#)
- W. Chu and Z. Ghahramani. Preference learning with gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 137–144. ACM, 2005. [62](#)
- Clover Studio, 2003. Viewtiful Joe, Capcom. [73](#)

- C. Conati. Probabilistic assessment of user's emotions in educational games. *Applied Artificial Intelligence*, 16(7-8):555–575, 2002. 19, 25
- M. Cook. Introducing mechanic miner, 2012. URL <http://www.gamesbyangelina.org/?p=227>. 225
- M. Cook and S. Colton. Multi-faceted evolution of simple arcade games. In *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*, pages 289–296. IEEE, 2011. 225
- M. Cook, S. Colton, and J. Gow. Initial results from co-operative co-evolution for automated platformer design. *Applications of Evolutionary Computation*, pages 194–203, 2012. 41
- R. Cowie, E. Douglas-Cowie, N. Tsapatsoulis, G. Votsis, S. Kollias, W. Fellenz, and J.G. Taylor. Emotion recognition in human-computer interaction. *Signal Processing Magazine, IEEE*, 18(1):32–80, 2001. 22
- Crystal Dynamics, Nixxes Software, Buzz Monkey Software, Santa Cruz Games, and EA Mobile, 2008. Tomb Raider: Underworld, Eidos Interactive, Feral Interactive. 46
- Mihaly Csikszentmihalyi. *Flow: The Psychology of Optimal Experience*. Harper Perennial, March 1991. xiii, 2, 20, 21
- G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989. 59
- R.J. Davidson, K.R. Scherer, and H.H. Goldsmith. *Handbook of affective sciences*. Oxford University Press, USA, 2002. 18
- O. Delalleau, E. Contal, E. Thibodeau-Laufer, R. Chandias, Y. Bengio, and F. Zhang. Beyond skill rating: Advanced matchmaking in ghost recon online. *IEEE Transactions on Computational Intelligence and AI in Games*, 2012. 31
- Delphine Software Internationals, 1995. Fade to Black, Electronic Arts. 73

BIBLIOGRAPHY

- DFC. Worldwide Market Forecasts for the Video Game and Interactive Entertainment Industry, 2012. URL <http://www.dfcint.com/>. 2
- J. Dias and A. Paiva. Feeling and reasoning: A computational model for emotional characters. *Progress in artificial intelligence*, pages 127–140, 2005. 19
- A. Drachen, A. Canossa, and G.N. Yannakakis. Player modeling using self-organization in tomb raider: underworld. In *IEEE Symposium on Computational Intelligence and Games*, pages 1–8. IEEE, 2009. 46
- A. Drachen, L.E. Nacke, G. Yannakakis, and A.L. Pedersen. Correlation between heart rate, electrodermal activity and player experience in first-person shooter games. In *Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games*, pages 49–54. ACM, 2010. 27
- Z. Duric, W.D. Gray, R. Heishman, F. Li, A. Rosenfeld, M.J. Schoelles, C. Schunn, and H. Wechsler. Integrating perceptual and cognitive modeling for adaptive and intelligent human-computer interaction. *Proceedings of the IEEE*, 90(7):1272–1289, 2002. 22
- EA Digital Illusions CE, 2008. *Mirror’s Edge*, Electronic Arts. 74
- A.E. Eiben and J.E. Smith. *Introduction to evolutionary computing*. springer, 2008. xiv, 51, 52, 53, 55, 56
- Clark Davidson Elliott. *The affective reasoner: a process model of emotions in a multi-agent system*. PhD thesis, Evanston, IL, USA, 1992. 19
- P.C. Ellsworth. William james and emotion: is a century of fame worth a century of misunderstanding? *Psychological Review*, 101(2):222, 1994. 18
- Epyx, 1984. *Impossible Mission*, Epyx. 72
- Exact Co. Ltd., 1995. *Jumping Flash!*, Sony Computer Entertainment. 73
- P. Fagerberg, A. Ståhl, and K. Höök. Designing Gestures for Affective Input: An Analysis of Shape, Effort and Valance. In *Proceedings of the 2nd International Conference on Mobile Ubiquitous and Multimedia (MUM 2003)*, 2003. 22

- B. Fasel and J. Luetttin. Automatic facial expression analysis: a survey. *Pattern Recognition*, 36(1):259–275, 2003. [22](#)
- C.N. Fiechter and S. Rogers. Learning subjective functions with large margins. In *Stanford University*, pages 287–294. Morgan Kaufmann Publishers, 2000. [62](#)
- Firaxis Games, 2005. Civilization IV, 2K Games & Aspyr. [34](#)
- L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley, New York, USA, 1966. [52](#)
- J. Fürnkranz and E. Hüllermeier. *Preference learning*. Springer-Verlag New York Inc, 2010. [61](#), [62](#)
- K.Z. Gajos, D.S. Weld, and J.O. Wobbrock. Automatically generating personalized user interfaces with supple. *Artificial intelligence*, 174(12):910–950, 2010. [1](#)
- M. Garofalakis, R. Rastogi, and K. Shim. Spirit: Sequential pattern mining with regular expression constraints. In *Proceedings of the international conference on very large data bases*, pages 223–234, 1999. [67](#)
- R. Gaudel, M. Sebag, et al. Feature selection as a one-player game. In *International Conference on Machine Learning*, pages 359–366, 2010. [224](#)
- M.T. Gervasio, M.D. Moffitt, M.E. Pollack, J.M. Taylor, and T.E. Uribe. Active preference learning for personalized calendar scheduling assistance. In *Proceedings of the 10th international conference on Intelligent user interfaces*, volume 5, pages 90–97. Citeseer, 2005. [62](#)
- S. Giannatos, M.J. Nelson, Y.G. Cheong, and G.N. Yannakakis. Suggesting new plot elements for an interactive story. In *Workshops at the Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2011. [40](#)
- S. Giannatos, Y.G. Cheong, M.J. Nelson, and G.N. Yannakakis. Generating narrative action schemas for suspense. In *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2012. [41](#)

BIBLIOGRAPHY

- K.M. Gilleade and A. Dix. Using frustration in the design of adaptive videogames. In *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 228–232. ACM, 2004. [23](#), [47](#), [126](#)
- Global Movie Production. Global Movie Production & Distribution: Market Research Report, 2012. URL <http://www.ibisworld.com/>. [2](#)
- J. Gow, R. Baumgarten, P. Cairns, S. Colton, and P. Miller. Unsupervised modelling of player style with lda. *IEEE Transactions on Computational Intelligence and AI in Games*, 2012. [31](#)
- M.K. Greenwald, E.W. Cook, and P.J. Lang. Affective judgment and psychophysiological response: Dimensional covariation in the evaluation of pictorial stimuli. *Journal of psychophysiology*, 1989. [19](#)
- J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.C. Hsu. Freespan: frequent pattern-projected sequential pattern mining. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 355–359. ACM, 2000. [67](#)
- Erin J. Hastings, Ratan K. Guha, and Kenneth O. Stanley. Evolving content in the galactic arms race video game. In *Proceedings of the 5th international conference on Computational Intelligence and Games, CIG'09*, pages 241–248, Piscataway, NJ, USA, 2009. IEEE Press. [32](#), [38](#), [40](#), [45](#), [48](#), [49](#)
- S. Haykin. *Neural Network Theory: a Comprehensive Foundation*. Prentice-Hall, 1999. [58](#)
- M. Hemberg and U.M. O'Reilly. Extending grammatical evolution to evolve digital surfaces with genr8. In *Proceedings of the 7th European Conference on Genetic Programming , EuroGP*, pages 299–308. Springer-Verlag, 2004. [57](#)
- M. Hendrikx, S. MEIJER, J. VAN DER VELDEN, and A. IOSUP. Procedural content generation for games: a survey. *ACM Transactions on Multimedia Computing, Communications and Applications*, 2011. [34](#)

- R. Herbrich, T. Graepel, P. Bollmann-Sdorra, and K. Obermayer. Learning preference relations for information retrieval. In *ICML-98 Workshop: text categorization and machine learning*, pages 80–84, 1998. [62](#)
- J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the theory of neural computation*, volume 1. Addison-Wesley, Reading, MA, 1991. [58](#)
- Hoei Corporation, 1981. Jump Bug, Rock-Ola. [72](#)
- John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI,, 1975. [52](#), [55](#)
- V. Hom and J. Marks. Automatic design of balanced board games. In *Proceedings of the 3rd Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 25–30, 2007. [41](#)
- Kristina Höök. Affective loop experiences - what are they? In *Lecture Notes in Computer Science*, volume 5033, pages 1–12. Springer, 2008. [2](#)
- M. Hoque, D. McDuff, and R. Picard. Exploring temporal patterns in classifying frustrated and delighted smiles. 2012. [25](#)
- G.S. Hornby and J.B. Pollack. The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2001 Congress on Evolutionary Computation*, volume 1, pages 600–607. IEEE, 2001. [57](#)
- K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989. [59](#)
- Ryan Houlette. Player Modeling for Adaptive Games. In Steve Rabin, editor, *AI Game Programming Wisdom 2*. Charles River Media, 2003. [44](#)
- Eva Hudlicka. Affective computing for game design. In *GAMEON-NA '08: Proceedings of the 4th Intl. North American Conference on Intelligent Games and Simulation*, pages 5–12, Montreal, Canada, 2008. [22](#), [25](#), [44](#)
- Kenneth Hullett and Jim Whitehead. Design patterns in fps levels. In *FDG '10: Proceedings of the Fifth International Conference on the Foundations of Digital Games*, pages 78–85. ACM, 2010. [30](#)

BIBLIOGRAPHY

- id Software and David A. Palmer Productions, 1990. Commander Keen , Apogee Software and Softdisk and Activision. [73](#)
- id Software, Logicware, Raster Productions, Hammerhead, and Hyperion Entertainment, 1997. Quake II, Activision. [46](#)
- W. A. Ijsselsteijn, Y. A. W. de Kort, and K. Poels. The Game Experience Questionnaire: Development of a self-report measure to assess the psychological impact of digital games. *FUGA technical report, Deliverable 3.3, Technical University Eindhoven.*, 2008. [27](#)
- A. Illiger, 2011. Tiny Wings, Andreas Illiger. [35](#)
- Infogrames, 1990. Alpha Waves, Infogrames and Data East. [73](#)
- Insomniac Games, 1998. Spyro the Dragon, Sony Computer Entertainment. [74](#)
- Interactive Data, 2011. SpeedTree. [35](#)
- S. Ioannou, G. Caridakis, K. Karpouzis, and S. Kollias. Robust Feature Detection for Facial Expression Recognition. *EURASIP Journal on Image and Video Processing*, (2), 2007. [22](#)
- K. Isbister, U. Schwekendiek, and J. Frye. Wriggle: An exploration of emotional and social effects of movement. In *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems*, pages 1885–1890. ACM, 2011. [24](#)
- Poika Isokoski, Markus Joos, Oleg Spakov, and Benoît Martin. Gaze controlled games. *Universal Access in the Information Society*, 8(4):323–337, 2009. [24](#)
- Howell O. Istance, Aulikki Hyrskykari, Stephen Vickers, and Thiago Chaves. For your eyes only: Controlling 3d online games by eye-gaze. In *Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part I*, pages 314–327, 2009. [24](#)
- S. Karakovskiy J. Togelius, G. N. Yannakakis and N. Shake. Believability in computer games. Springer-Verlag. to appear, 2011. [77](#), [78](#), [227](#)

- Charlene Jennett, Anna L. Cox, Paul Cairns, Samira Dhoparee, Andrew Epps, Tim Tijs, and Alison Walton. Measuring and defining the experience of immersion in games. *International Journal of Human-Computer Studies*, 66:641–661, 2008. ISSN 1071-5819. [22](#)
- M. Jennings-Teats, G. Smith, and N. Wardrip-Fruin. Polymorph: dynamic difficulty adjustment through level generation. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, page 11. ACM, 2010. [30](#), [40](#)
- L. Johnson, G.N. Yannakakis, and J. Togelius. Cellular automata for real-time generation of infinite cave levels. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, page 10. ACM, 2010. [40](#)
- S. Kaiser, T. Wehrle, and S. Schmidt. Emotional episodes, facial expressions, and reported feelings in human-computer interactions. In *Proceedings of the Xth Conference of the International Society for Research on Emotions*, pages 82–86. Würzburg: ISRE Publications, 1998. [23](#), [24](#)
- A. Kapoor, W. Burleson, and R.W. Picard. Automatic prediction of frustration. *International Journal of Human-Computer Studies*, 65(8):724–736, 2007. [22](#), [24](#)
- S. Karakovskiy and J. Togelius. The mario ai benchmark and competitions. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):55–67, 2012. [78](#)
- C. D. Katsis, N. Katertsidis, George Ganiatsas, and Dimitrios I. Fotiadis. Toward emotion recognition in car-racing drivers: A biosignal processing approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 38(3):502–512, 2008. [23](#)
- S. Kazmi and IJ Palmer. Action recognition for support of adaptive gameplay: A case study of a first person shooter. *International Journal of Computer Games Technology*, page 1, 2010. [48](#)

BIBLIOGRAPHY

- G. Kelly and H. McCabe. Citygen: An interactive system for procedural city generation. In *Fifth International Conference on Game Design and Technology*, pages 8–16, 2007. [34](#)
- M. Kerssemakers, J. Tuxen, J. Togelius, and G.N. Yannakakis. A procedural procedural level generator generator. *IEEE Conference on Computational Intelligence and AI in Games*, pages 335–341, 2012. [41](#)
- J. Kim. Bimodal emotion recognition using speech and physiological changes. *Robust Speech Recognition and Understanding*, pages 265–280, 2007. [23](#)
- J. Kim and E. André. Emotion recognition based on physiological changes in music listening. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(12):2067–2083, 2008. [23](#)
- K.H. Kim, SW Bang, and SR Kim. Emotion recognition system using short-term monitoring of physiological signals. *Medical and biological engineering and computing*, 42(3):419–427, 2004. [23](#)
- A.N. Kolmogorov. *On the Representation of Continuous Functions of Several Variables in the Form of Super Positions of Continuous Functions of One Variable and Additive Functions*. SLA Translations Center, 1963. [59](#)
- Konami, 2007. Pro Evolution Soccer 2008, Konami. [48](#)
- R. Koster. *A theory of fun for game design*. Paraglyph press, 2004. [3](#), [21](#)
- J.R. Koza. *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*. Stanford University, Department of Computer Science, 1990. [52](#)
- J.R. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4(2):87–112, 1994. [52](#)
- J.R. Koza, M.A. Keane, and M.J. Streeter. Evolving inventions. *Scientific American*, 288(2):40–7, 2003. [56](#)

- J.R. Koza, M.A. Keane, M.J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic programming IV: Routine human-competitive machine intelligence*. Springer-Verlag New York Inc, 2005. [52](#), [56](#)
- R. Lazarus. *Emotion and Adaptation*. Oxford University Press, 1991. [19](#)
- Nicole Lazzaro. Why We Play Games: Four Keys to More Emotion Without Story. In *Game Developers Conference*, March 2004. [21](#)
- Sangkyung Lee and Keechul Jung. Dynamic Game Level Design Using Gaussian Mixture Model. In *PRICAI 2006: Trends in Artificial Intelligence*, volume 4099 of *Lecture Notes in Computer Science*, pages 955–959. Springer Berlin / Heidelberg, 2006. [43](#)
- Iolanda Leite, André Pereira, Samuel Mascarenhas, Ginevra Castellano, Carlos Martinho, Rui Prada, and Ana Paiva. Closing the loop: from affect recognition to empathic interaction. In *Proceedings of the 3rd international workshop on Affective interaction in natural environments*, AFFINE '10, pages 43–48, New York, NY, USA, 2010. ACM. [22](#)
- M. Li, X. Chen, X. Li, B. Ma, and P.M.B. Vitányi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004. [104](#), [138](#)
- A. Liapis, G.N. Yannakakis, and J. Togelius. Optimizing visual properties of game content through neuroevolution. In *Artificial Intelligence for Interactive Digital Entertainment Conference*, 2011. [40](#)
- A. Liapis, G. Yannakakis, and J. Togelius. Adapting models of visual aesthetics for personalized content creation. *IEEE Transactions on Computational Intelligence and AI in Games*, 2012. [32](#), [40](#), [48](#)
- M.Y. Lin and S.Y. Lee. Fast discovery of sequential patterns by memory indexing. *Data Warehousing and Knowledge Discovery*, pages 227–237, 2002. [67](#)
- C.L. Lisetti and F. Nasoz. Maui: a multimodal affective user interface. In *Proceedings of the tenth ACM international conference on Multimedia*, pages 161–170. ACM, 2002. [22](#)

BIBLIOGRAPHY

- D. Loiacono, L. Cardamone, and P.L. Lanzi. Automatic track generation for high-end racing games using evolutionary computation. *IEEE Conference on Computational Intelligence and AI in Games*, 3(3):245–259, 2011. [40](#), [42](#)
- L. Maat and M. Pantic. Gaze-x: Adaptive, affective, multimodal interface for single-user office scenarios. *Artificial Intelligence for Human Computing*, pages 251–271, 2007. [22](#)
- N.R. Mabroukeh and C.I. Ezeife. A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys (CSUR)*, 43(1):3, 2010. [68](#)
- B. Magerko, C. Heeter, J. Fitzgerald, and B. Medler. Intelligent adaptation of digital game-based learning. In *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, pages 200–203. ACM, 2008. [29](#)
- T. Mahlmann, J. Togelius, and G. Yannakakis. Towards procedural strategy game generation: Evolving complementary unit types. *Applications of Evolutionary Computation*, pages 93–102, 2011. [40](#), [44](#)
- T. Mahlmann, J. Togelius, and G.N. Yannakakis. Evolving card sets towards balancing dominion. In *2012 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2012. [40](#), [44](#)
- Thomas Malone. *What makes computer games fun?* New York, NY, USA, 1981. ACM. [3](#), [20](#), [29](#)
- Regan L. Mandryk and M. Stella Atkins. A fuzzy physiological approach for continuously modeling emotion during interaction with play technologies. *International Journal of Human-Computer Studies*, 65(4):329–347, April 2007. ISSN 1071-5819. [27](#)
- R.L. Mandryk, K.M. Inkpen, and T.W. Calvert. Using psychophysiological techniques to measure user experience with entertainment technologies. *Behaviour & Information Technology*, 25(2):141–158, 2006. [27](#)
- H. Mannila and H. Toivonen. Discovering generalized episodes using minimal occurrences. In *Proceedings of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining*, pages 146–151, 1996. [65](#)

- A. Martin, A. Lim, S. Colton, and C. Browne. Evolving 3d buildings for the prototype video game subversion. *Applications of Evolutionary Computation*, pages 111–120, 2010. [41](#)
- H.P. Martínez and G.N. Yannakakis. Genetic search feature selection for affective modeling: a case study on reported preferences. In *Proceedings of the 3rd international workshop on Affective interaction in natural environments*, pages 15–20. ACM, 2010. [27](#), [224](#)
- H.P. Martinez and G.N. Yannakakis. Mining multimodal sequential patterns: A case study on affect detection. In *Proceedings of the 13th International Conference in Multimodal Interaction, ICMI 2011, Alicante*. ACM Press, November 2011. [27](#), [69](#), [139](#)
- H.P. Martinez, A. Jhala, and G.N. Yannakakis. Analyzing the impact of camera viewpoint on player psychophysiology. In *International Conference on Affective Computing and Intelligent Interaction and Workshops*, pages 1–6. IEEE, 2009. [9](#), [10](#), [11](#), [28](#), [62](#), [224](#)
- H.P. Martínez, K. Hullett, and G.N. Yannakakis. Extending neuro-evolutionary preference learning through player modeling. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 313–320. IEEE, 2010. [46](#)
- Peter A. Mawhorter and Michael Mateas. Procedural level generation using occupancy-regulated extension. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, pages 351–358, 2010. [78](#)
- Maxis, 2008. Spore, Electronic Arts. [34](#)
- D. McDuff, R. el Kaliouby, and R. Picard. Crowdsourced data collection of facial responses. In *Proceedings of the 13th international conference on multimodal interfaces*, pages 11–18. ACM, 2011. [25](#)
- D. McDuff, R. el Kaliouby, and R. Picard. Crowdsourcing facial responses to online videos. In *IEEE Transactions on Affective Computing*, 2012. [25](#)

BIBLIOGRAPHY

- S. McGlinchey. Learning of ai players from game observation data. In *Proceedings of the 4th International Conference on Intelligent Games and Simulation (GAME-ON 2003)*, pages 106–110, 2003. [46](#)
- S.W. Mcquiggan, B.W. Mott, and J.C. Lester. Modeling self-efficacy in intelligent tutoring systems: An inductive approach. *User Modeling and User-Adapted Interaction*, 18(1):81–123, 2008. [27](#)
- A. Melzer, I. Derks, J. Heydekorn, and G. Steffgen. Click or strike: realistic versus standard game controls in violent video games and their effects on aggression. *Entertainment Computing-ICEC 2010*, pages 171–182, 2010. [24](#)
- Microsoft Game Studios, 2005. Forza Motorsport, Microsoft. [36](#)
- D. Milam and M. Seif El-Nasr. Analysis of level design 'push & pull' within 21 games. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, pages 139–146. ACM, 2010. [30](#)
- Minecraft Wiki. Minecraft wiki. URL <http://www.minecraftwiki.net/>. [37](#)
- Mojang, 2011. Minecraft, Mojang and Microsoft Studios. [35](#), [37](#)
- D.J. Montana and L. Davis. Training feedforward neural networks using genetic algorithms. In *Proceedings of the eleventh international joint conference on artificial Intelligence*, volume 1, pages 762–767. San Mateo, CA, 1989. [64](#)
- Philippe Morel, Hatem Hamda, and Marc Schoenauer. Computational chair design using genetic algorithms. *Concept*, 71(3):95–99, 2005. [87](#)
- D. Moura, M. Seif El-Nasr, and C.D. Shaw. Visualizing and understanding players' behavior in video games: discovering patterns and supporting aggregation and comparison. In *Proceedings of the 2011 ACM SIGGRAPH Symposium on Video Games*, page 2. ACM, 2011. [30](#)
- P. Mueller, S. Haegler, A. Ulmer, S. Schubiger, M. Specht, S. Müller, and B. Weber, 2011. CityEngine, Esri R&D Center Zurich. [35](#)

- L.E. Nacke and C.A. Lindley. Affective ludology, flow and immersion in a first-person shooter: Measurement of player experience. *arXiv preprint arXiv:1004.0248*, 2010. 27
- Lennart Nacke, Sophie Stellmach, Dennis Sasse, Jörg Niesenhaus, and Raimund Dachsel. Laif: A logging and interaction framework for gaze-based interfaces in virtual entertainment environments. In *Electronic Proceedings of the Interactive Cultures Conference 2010*, pages 19–28. Oldenburg Publishing, 9 2010. 24
- Namco, 1980. Pac-Man, Namco and Midway. 72
- Namco, 2001. Klonoa 2: Lunatea’s Veil , Namco and SCEE. 73
- F. Nasoz, C.L. Lisetti, K. Alvarez, and N. Finkelstein. Emotion recognition from physiological signals for user modeling of affect. In *Proceedings of the 3rd Workshop on Affective and Attitude User Modelling (Pittsburgh, PA, USA, 2003)*. xiii, 28
- NaturalMotion, 2007. Euphoria, NaturalMotion. 35
- Naughty Dog, 2001. Jak and Daxter: The Precursor Legacy, Sony Computer Entertainment. 74
- V. Nicollet. *Difficulty in dexterity-based platform games*. [Online]. Available: <http://www.gamedev.net/reference/design/features/platformdiff>, March 2004. 166
- Nintendo Creative Department, 1985. Super Mario Bros, Nintendo. 37, 46, 72, 74
- Nintendo EAD, 1995. Super Mario World 2: Yoshi’s Island , Nintendo. 73
- Nintendo EAD, 1996. Super Mario 64, Nintendo. 73
- Nintendo EAD, 2007. Super Mario Galaxy, Nintendo. 74
- Nintendo EAD, Rare, Namco, Paon, and Retro Studios. Donkey kong, 1981. Nintendo. 71

BIBLIOGRAPHY

- Tim Oates, Matthew D. Schmill, David Jensen, and Paul R. Cohen. A family of algorithms for finding temporal structure in data. In *In 6th Intl. Workshop on AI and Statistics*, pages 371–378, 1997. [65](#)
- J.K. Olesen, G.N. Yannakakis, and J. Hallam. Real-time challenge balance in an rts game using rtneat. In *Computational Intelligence and Games, CIG'08.*, pages 87–94. IEEE, 2008. [45](#)
- M. O'Neill and A. Brabazon. Recent patents on genetic programming. *Economics*, 32(1):251–166, 2001. [56](#)
- M. O'Neill and A. Brabazon. Evolving a logo design using lindenmayer systems, postscript & grammatical evolution. In *IEEE Congress on Evolutionary Computation*, pages 3788–3794. IEEE, 2008. [57](#)
- M. O'Neill and C. Ryan. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358, 2001. [41](#), [56](#)
- M. O'Neill, R. Cleary, and N. Nikolov. Solving knapsack problems with attribute grammars. In *Proceedings of the Third Grammatical Evolution Workshop (GEWS04)*. Citeseer, 2004. [113](#), [225](#)
- M. O'Neill, E. Hemberg, C. Gilligan, E. Bartley, J. McDermott, and A. Brabazon. Geva: grammatical evolution in java. *ACM SIGEVOlution*, 3(2):17–22, 2008. [96](#)
- M. O'Neill, J.M. Swafford, J. McDermott, J. Byrne, A. Brabazon, E. Shotton, C. McNally, and M. Hemberg. Shape grammars and grammatical evolution for evolutionary design. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1035–1042. ACM, 2009. [57](#)
- M. O'Neill, J. McDermott, J.M. Swafford, J. Byrne, E. Hemberg, A. Brabazon, E. Shotton, C. McNally, and M. Hemberg. Evolutionary design using grammatical evolution and shape grammars: Designing a shelter. *International Journal of Design Engineering*, 3(1):4–24, 2010. [56](#), [86](#)
- J. Ortega, N. Shaker, J. Togelius, and G.N. Yannakakis. Imitating human playing styles in super mario bros. *Entertainment Computing*, 2012. [46](#), [78](#), [227](#)

- A. Ortony, G.L. Clore, and A. Collins. *The cognitive structure of emotions*. Cambridge university press, 1990. [19](#)
- M. Pantic and L.J.M. Rothkrantz. Toward an affect-sensitive multimodal human-computer interaction. *Proceedings of the IEEE*, 91(9):1370–1390, 2003. [22](#)
- PCG Wiki. Procedural content generation wiki. URL <http://pcg.wikidot.com/>. [34](#)
- Chris Pedersen, Julian Togelius, and Georgios N. Yannakakis. Modeling player experience in super mario bros. In *CIG'09: Proceedings of the 5th international conference on Computational Intelligence and Games*, pages 132–139, 2009. [29](#), [62](#), [63](#), [151](#), [224](#)
- Chris Pedersen, Julian Togelius, and Georgios N. Yannakakis. Modeling player experience for content creation. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):54–67, 2010. [9](#), [10](#), [11](#), [12](#), [29](#), [32](#), [40](#), [45](#), [48](#), [62](#), [78](#), [125](#), [126](#)
- Diego Perez, Miguel Nicolau, Michael O'Neill, and Anthony Brabazon. Evolving behaviour trees for the mario ai competition using grammatical evolution. In *Applications of Evolutionary Computation*, volume 6624 of *Lecture Notes in Computer Science*, pages 123–132. Springer Berlin / Heidelberg, 2011. [78](#)
- M. Persson. Infinite mario bros. URL <http://www.mojang.com/notch/mario/>. [37](#), [41](#)
- R.W. Picard. *Affective computing*. Perceptual Computing Section, Media Laboratory, Massachusetts Institute of Technology, 1995. [21](#), [22](#)
- R.W. Picard, E. Vyzas, and J. Healey. Toward machine emotional intelligence: Analysis of affective physiological state. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1175–1191, 2001. [23](#)
- D. Plans and D. Morelli. Experience-driven procedural music generation for games. *IEEE Transactions on Computational Intelligence and AI in Games*, 2012. [31](#)

BIBLIOGRAPHY

- K.H. Pribram and F.T. Melges. Psychophysiological basis of emotion. *Handbook of clinical neurology*, 3:316–341, 1969. [18](#)
- P. Rani, N. Sarkar, and C. Liu. Maintaining optimal challenge in computer games through real-time physiological feedback. In *Proceedings of the 1st International Conference on Augmented Cognition, Las Vegas, NV*, 2005. [27](#)
- A.S. Rao, M.P. Georgeff, et al. Bdi agents: From theory to practice. In *Proceedings of the first international conference on multi-agent systems (ICMAS-95)*, pages 312–319. San Francisco, 1995. [20](#)
- JC Read and SJ MacFarlane. Measuring fun. *Computers and Fun*, 3, 2000. [21](#)
- JC Read, SJ MacFarlane, and C. Casey. Endurability, engagement and expectations: Measuring children’s fun. In *Interaction Design and Children*, volume 2, pages 1–23. Shaker Publishing Eindhoven, 2002. [130](#), [181](#)
- Realtime Associates, 1995. Bug!, Sega. [73](#)
- I. Rechenberg. *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution*. Frommann-Holzboog, 1973. [52](#)
- W.S. Reilly. Believable social and emotional agents. Technical report, DTIC Document, 1996. [19](#)
- S. Risi, J. Lehman, D.B. D’Ambrosio, R. Hall, and K.O. Stanley. Combining search-based procedural content generation and social gaming in the petalz video game. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*. AAAI Press, 2012. [32](#)
- S.A. Roberts and S.M. Lucas. Evolving spaceship designs for optimal control and the emergence of interesting behaviour. [40](#)
- Rockstar London and Rockstar Leeds and Rockstar Toronto, 2007. Manhunt, Rockstar Games. [24](#)
- F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. [59](#)

- D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1*. MIT Press, Cambridge, MA, USA, 1986. [60](#)
- D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. Technical report, 1985. [58](#)
- J.A. Russell. A circumplex model of affect. *Journal of personality and social psychology*, 39(6):1161, 1980. [19](#)
- J.A. Russell and A. Mehrabian. Evidence for a three-factor theory of emotions. *Journal of research in Personality*, 11(3):273–294, 1977. [19](#)
- T. Saari, M. Turpeinen, K. Kuikkaniemi, I. Kosunen, and N. Ravaja. Emotionally adapted games—an example of a first person shooter. *Human-Computer Interaction. Interacting in Various Application Domains*, pages 406–415, 2009. [48](#)
- Jyotirmay Sanghvi, Ginevra Castellano, Iolanda Leite, André Pereira, Peter W. McOwan, and Ana Paiva. Automatic analysis of affective postures and body motion to detect engagement with a game companion. In *Proceedings of the 6th international conference on Human-robot interaction, HRI '11*, pages 305–312, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0561-7. [22](#)
- N. Savva, A. Scarinzi, and N. Berthouze. Continuous recognition of player’s affective body expression as dynamic quality of aesthetic experience. *IEEE Transactions on Computational Intelligence and AI in Games*, 2012. [31](#)
- S. Schachter. The interaction of cognitive and physiological determinants of emotional state. *Advances in experimental social psychology*, 1:49–80, 1964. [23](#)
- Jocelyn Scheirer, Raul Fernandez, Jonathan Klein, and Rosalind W. Picard. Frustrating the user on purpose: a step toward building an affective computer. *Interacting with Computers*, 14(2):93–118, 2001. [23](#)
- K. R. Scherer. On the nature and function of emotion: A component process approach. In K. R. Scherer and P. Ekman, editors, *Approaches to emotion*, pages 293–317, Hillsdale, 1984. NJ: Erlbaum, NJ: Erlbaum. [18](#), [23](#)

BIBLIOGRAPHY

- K. R. Scherer. Emotion, the psychological structure of emotions. In N. J. Smelser and P. B. Baltes, editors, *International encyclopedia of the social & behavioral sciences*, pages 4472–4477, Oxford, 2002. Harvard Libraries, Harvard Libraries. 18, 23
- K.R. Scherer. What are emotions? and how can they be measured? *Social science information*, 44(4):695–729, 2005. 18, 23, 25, 26
- K.R. Scherer and H. Ellgring. Are facial expressions of emotion produced by categorical affect programs or dynamically driven by appraisal? *Emotion*, 7(1):113, 2007. 19
- N. Sebe, I. Cohen, and T.S. Huang. Multimodal emotion recognition. *Handbook of Pattern Recognition and Computer Vision*, 4:387–419, 2005. 22
- Magy Seif El-Nasr and Su Yan. Visual attention in 3d video games. In *Advances in Computer Entertainment Technology*, page 22, 2006. 24
- N. Shaker. Infinite mario bros demos, 2011. URL <http://noorshaker.com/Demos.html>. 228
- N. Shaker, G.N. Yannakakis, J. Togelius, M. Nicolau, and M. O'Neill. Evolving personalized content for super mario bros using grammatical evolution. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*. AAAI Press. 41
- N. Shaker, M. Nicolau, G. Yannakakis, J. Togelius, and M. O'Neill. Evolving levels for super mario bros using grammatical evolution. *IEEE Conference on Computational Intelligence and Games (CIG)*, pages 304–311, 2012. 41, 57
- Noor Shaker, Julian Togelius, and Georgios N. Yannakakis. Towards automatic personalized content generation for platform games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*. AAAI Press, 2010. 37, 40, 41, 78
- Noor Shaker, Julian Togelius, Georgios N. Yannakakis, Ben Weber, Tomoyuki Shimizu, Tomonori Hashiyama, Nathan Sorenson, Philippe Pasquier, Peter

- Mawhorter, Glen Takahashi, Gillian Smith, and Robin Baumgarten. The 2010 Mario AI championship: Level generation track. *IEEE Transactions on Computational Intelligence and Games*, 3:332–347, 2011. [78](#)
- Shiny Entertainment and Playmates Interactive Entertainment, 1994. Earthworm Jim , Sega of America and Virgin Interactive and Takara. [73](#)
- R. Smelik, T. Tutenel, K.J. de Kraker, and R. Bidarra. Integrating procedural generation and manual editing of virtual worlds. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, page 2. ACM, 2010. [39](#)
- R.M. Smelik, K.J. De Kraker, T. Tutenel, R. Bidarra, and S.A. Groenewegen. A survey of procedural methods for terrain modelling. In *Proceedings of the CASA Workshop on 3D Advanced Media In Gaming And Simulation (3AMIGAS)*. Citeseer, 2009. [34](#)
- A. M. Smith and Michael Mateas. Variations Forever: Flexibly Generating Rule-sets from a Sculptable Design Space of Mini-Games. *IEEE Transactions on Computational Intelligence and AI in Games*, 2010. [40](#)
- A.M. Smith, C. Lewis, K. Hullett, G. Smith, and A. Sullivan. An inclusive taxonomy of player modeling. *University of California, Santa Cruz, Tech. Rep. UCSC-SOE-11-13*, 2011. [44](#)
- C. Smith and H. Scott. A componential approach to the meaning of facial expressions. *The psychology of facial expression*, 229, 1997. [19](#)
- G. Smith and J. Whitehead. Analyzing the expressive range of a level generator. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, page 4. ACM, 2010. [95](#), [98](#), [99](#), [102](#)
- G. Smith, J. Whitehead, and M. Mateas. Tanagra: A mixed-initiative level design tool. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, pages 209–216. ACM, 2010. [11](#), [30](#), [39](#), [229](#)

BIBLIOGRAPHY

- Gillian Smith, Mee Cha, and Jim Whitehead. A framework for analysis of 2d platformer levels. In *Sandbox '08: Proceedings of the 2008 ACM SIGGRAPH symposium on Video games*, pages 75–80, New York, NY, USA, 2008. ACM. [30](#)
- G.M. Smith. *Expressive Design Tools: Procedural Content Generation for Game Designers*. PhD thesis, University of California, 2012. [95](#)
- J. David Smith and T. C. Nicholas Graham. Use of eye movements for video game control. In *Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology, ACE '06*, New York, NY, USA, 2006. ACM. [24](#)
- R.L. Solomon. The opponent-process theory of acquired motivation: the costs of pleasure and the benefits of pain. *American Psychologist*, 35(8):691, 1980. [18](#)
- Sonic Team and Dimps, 2011. Sonic Generations, Sega. [74](#)
- Sonic Team and Sega Technical Institute, 1994. Sonic 3 & Knuckles , Sega. [73](#)
- N. Sorenson, P. Pasquier, and S. DiPaola. A generic approach to challenge modeling for the procedural creation of video game levels. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):229–244, 2011. [40](#), [41](#), [78](#)
- Nathan Sorenson and Philippe Pasquier. Towards a generic framework for automated video game level creation. In *Proceedings of the European Conference on Applications of Evolutionary Computation (EvoApplications)*, volume 6024, pages 130–139. Springer LNCS, 2010. [44](#), [78](#)
- P. Spronck, Sprinkhuizen I. Kuyper, and E. Postma. Difficulty scaling of game AI. In *Proceedings of the 5th International Conference on Intelligent Games and Simulation (GAME-ON 2004)*, pages 33–37, 2004. [43](#)
- P. Spronck, I. Balemans, and G. van Lankveld. Player profiling with fallout 3. In *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2012. [226](#)

- Pieter Spronck, Marc Ponsen, and Eric Postma. Adaptive game ai with dynamic scripting. In *Machine Learning*, pages 217–248. Kluwer, 2006. [43](#)
- R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. *Advances in Database Technology*, pages 1–17, 1996. [65](#), [67](#), [68](#), [105](#), [139](#)
- K.O. Stanley and R. Miikkulainen. Efficient evolution of neural network topologies. In *Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02.*, volume 2, pages 1757–1762. IEEE, 2002. [61](#), [224](#)
- K.O. Stanley, B.D. Bryant, and R. Miikkulainen. Real-time neuroevolution in the nero video game. *IEEE Transactions on Evolutionary Computation*, 9(6): 653–668, 2005. [36](#)
- P. Sundström. *Exploring the affective loop*. PhD thesis, Stockholm University, 2005. [2](#), [18](#), [22](#)
- M. Suwa, N. Sugie, and K. Fujimora. A preliminary note on pattern recognition of human emotional expression. *Proceedings of the International Joint Conference on Pattern Recognition*, pages 408–410, 1978. [22](#)
- P. Sweetser and P. Wyeth. Gameflow: a model for evaluating player enjoyment in games. *Computers in Entertainment (CIE)*, 3(3):3–3, 2005. [20](#)
- Jonathan Sykes. Affective gaming: measuring emotion through the gamepad. In *CHI 2003: New Horizons*, pages 732–733. ACM Press, 2003. [25](#)
- H. Takagi. Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, 2001. [56](#)
- Y. Takatsuki. Cost headache for game developers, 2007. news.bbc.co.uk/2/hi/business/7151961.stm. [33](#)
- J. Tao and T. Tan. Affective computing: A review. *Affective Computing and Intelligent Interaction*, pages 981–995, 2005. [22](#), [27](#)

BIBLIOGRAPHY

- The Entertainment Software Association, 2012. URL <http://www.theesa.com/>.
1, 2
- C. Thureau, C. Bauckhage, and G. Sagerer. Learning human-like movement behavior for computer games. In *Proceedings of International Conference on the Simulation of Adaptive Behavior*, pages 315–323, 2004. 46
- C. Thureau, T. Paczian, and C. Bauckhage. Is bayesian imitation learning the route to believable gamebots. *Proc. GAME-ON North America*, pages 3–9, 2005. 46
- Tim J. W. Tijs, Dirk Brokken, and Wijnand IJsselsteijn. Creating an emotionally adaptive game. In *Entertainment Computing*, pages 122–133, 2008. 27, 48
- J. Togelius and J. Schmidhuber. An experiment in automatic game design. In *IEEE Symposium On Computational Intelligence and Games. CIG'08*, pages 111–118. IEEE, 2008. 41, 44
- J. Togelius, R. D. Nardi, and S. M. Lucas. Making racing fun through player modeling and track evolution. In *Proceedings of the SAB'06 Workshop on Adaptive Approaches for Optimizing Player Satisfaction in Computer and Physical Games*, 2006. 30, 40, 44, 45
- J. Togelius, R. De Nardi, and S.M. Lucas. Towards automatic personalised content creation for racing games. In *IEEE Symposium on Computational Intelligence and Games, 2007. CIG 2007*, pages 252–259. IEEE, 2007. 42, 48, 49
- J. Togelius, S. Karakovskiy, and R. Baumgarten. The 2009 mario ai competition. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2010a. 196
- J. Togelius, M. Preuss, N. Beume, S. Wessing, J. Hagelbäck, and G.N. Yannakakis. Multiobjective exploration of the starcraft map space. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*, pages 265–272. Citeseer, 2010b. 40, 42

- J. Togelius, M. Preuss, and G.N. Yannakakis. Towards multiobjective procedural map generation. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, page 3. ACM, 2010c. [44](#)
- Julian Togelius, Sergey Karakovskiy, Jan Koutník, and Jürgen Schmidhuber. Super mario evolution. In *Proceedings of the 5th international conference on Computational Intelligence and Games, CIG'09*, pages 156–161, Piscataway, NJ, USA, 2009. IEEE Press. [77](#)
- Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. Search-based procedural content generation. In *Proceedings of EvoApplications*, volume 6024. Springer LNCS, 2010d. [12](#), [40](#)
- S. Tognetti, M. Garbarino, A. Bonarini, and M. Matteucci. Modeling enjoyment preference from physiological responses in a car racing game. In *IEEE Symposium on Computational Intelligence and Games (CIG)*, pages 321–328. IEEE, 2010. [28](#), [29](#)
- Traveller’s Tales and Robosoft Technologies , 2006. *Lego Star Wars II: The Original Trilogy*, LucasArts, TT Games, Feral Interactive. [48](#)
- IG Tsoulos and IE Lagaris. Solving differential equations with genetic programming. *Genetic Programming and Evolvable Machines*, 7(1):33–54, 2006. [57](#)
- Ubisoft, Ubisoft Montpellier, and Digital Eclipse, 1995. *Rayman* , Ubisoft. [73](#)
- Ubisoft Montpellier, Ubisoft, DC Studios, and Gameloft, 1999. *Rayman 2: The Great Escape*, Ubisoft and Gameloft. [74](#)
- Universal, 1980. *Space Panic*, Universal. [71](#)
- Valve Corporation, 2008. *Left 4 Dead*, Valve Corporation. [36](#), [38](#)
- Albert J. N. van Breemen, Xue Yan, and Bernt Meerbeek. icat: an animated user-interface robot with personality. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 143–144, 2005. [22](#)

BIBLIOGRAPHY

- WM Van den Hoogen, WA IJsselsteijn, and YAW de Kort. Exploring behavioral expressions of player experience in digital games. In *Proceedings of the Workshop on Facial and Bodily Expression for Control and Adaptation of Games ECAG*, pages 11–19, 2008. [24](#)
- G. van Lankveld, P. Spronck, and M. Rauterberg. Difficulty scaling through incongruity. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference, Stanford, California, USA*, 2008. [45](#)
- G. van Lankveld, P. Spronck, J. van den Herik, and A. Arntz. Games as personality profiling tools. In *IEEE Conference on Computational Intelligence and Games (CIG)*, pages 197–202. IEEE, 2011. [226](#)
- Ning Wang and Stacy Marsella. Introducing evg: An emotion evoking game. In *Intelligent Virtual Agents*, volume 4133 of *Lecture Notes in Computer Science*, pages 282–291, 2006. [23](#)
- D. Watson, L.A. Clark, and A. Tellegen. Development and validation of brief measures of positive and negative affect: the panas scales. *Journal of personality and social psychology*, 54(6):1063, 1988. [19](#)
- B.G. Weber and M. Mateas. A data mining approach to strategy prediction. In *IEEE Symposium on Computational Intelligence and Games, 2009.*, pages 140–147. IEEE, 2009. [46](#)
- A.W. Whitney. A direct method of nonparametric measurement selection. *IEEE Transactions on Computers*, 100(9):1100–1103, 1971. [62](#)
- C.E. Williams and K.N. Stevens. Emotions and speech: Some acoustical correlates. *The Journal of the Acoustical Society of America*, 52(4B):1238–1250, 1972. [22](#)
- C. Wong, J. Kim, E. Han, and K. Jung. Human-centered modeling for style-based adaptive games. *Journal of Zhejiang University-Science A*, 10(4):530–534, 2009. [46](#)
- M. Wooldridge. *Reasoning about rational agents*. MIT press, 2000. [20](#)

- G. Yannakakis and J. Hallam. Entertainment modeling in physical play through physiology beyond heart-rate. *Affective Computing and Intelligent Interaction*, pages 254–265, 2007. [9](#), [10](#), [11](#), [125](#)
- G. Yannakakis and J. Hallam. Ranking vs. preference: a comparative study of self-reporting. *Affective Computing and Intelligent Interaction*, pages 437–446, 2011a. [126](#)
- G. N. Yannakakis and J. Hallam. Real-time Game Adaptation for Optimizing Player Satisfaction. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(2):121–133, June 2009. [29](#)
- G. N. Yannakakis and J. Togelius. Experience-Driven Procedural Content Generation. *IEEE Transactions on Affective Computing*, 2011. [xiii](#), [2](#), [6](#), [7](#), [12](#), [19](#), [26](#), [38](#), [42](#), [43](#), [45](#), [47](#), [48](#), [123](#), [126](#), [226](#)
- Georgios Yannakakis and John Hallam. A generic approach for generating interesting interactive pac-man opponents. In *In Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 94–101, 2005. [30](#), [43](#)
- Georgios Yannakakis and John Hallam. Towards Capturing and Enhancing Entertainment in Computer Games. In *Advances in Artificial Intelligence*, volume 3955 of *Lecture Notes in Computer Science*, pages 432–442. Springer Berlin / Heidelberg, 2006. [28](#), [29](#), [45](#), [50](#)
- Georgios N. Yannakakis and John Hallam. Interactive opponents generate interesting games. In *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, pages 240–247, 2004a. [44](#)
- Georgios N. Yannakakis and John Hallam. Evolving opponents for interesting interactive computer games. pages 499–508. Proceedings of the 8th International Conference on Simulation of Adaptive Behavior (SAB-04). The MIT Press, 2004b. [43](#)

BIBLIOGRAPHY

- Georgios N. Yannakakis, Manolis Maragoudakis, and John Hallam. Preference learning for cognitive modeling: a case study on entertainment preferences. *IEEE Transactions on Systems, Man, and Cybernetics. Part A*, 39:1165–1175, November 2009. ISSN 1083-4427. [62](#), [63](#), [117](#), [119](#), [122](#)
- G.N. Yannakakis. Game adaptivity impact on affective physical interaction. In *3rd International Conference on Affective Computing and Intelligent Interaction and Workshops*, pages 1–6. IEEE, 2009a. [12](#)
- G.N. Yannakakis. Learning from preferences and selected multimodal features of players. In *Proceedings of the 2009 international conference on Multimodal interfaces*, pages 115–118. ACM, 2009b. [224](#)
- G.N. Yannakakis. Preference learning for affective modeling. In *International Conference on Affective Computing and Intelligent Interaction and Workshops*, pages 1–6. IEEE, 2009c. [224](#)
- G.N. Yannakakis. Game ai revisited. In *Proceedings of the 9th conference on Computing Frontiers*, pages 285–292. ACM, 2012. [2](#), [223](#)
- G.N. Yannakakis and J. Hallam. Erratum: Ranking vs. preference: a comparative study of self-reporting. In *Proceedings of the 4th international conference on Affective computing and intelligent interaction-Volume Part I*, pages 619–619. Springer-Verlag, 2011b. [26](#)
- G.N. Yannakakis, H.H. Lund, and J. Hallam. Modeling children’s entertainment in the playware playground. In *IEEE Symposium on Computational Intelligence and Games*, pages 134–141. IEEE, 2006. [28](#), [64](#)
- G.N. Yannakakis, J. Hallam, and H.H. Lund. Entertainment capture through heart rate activity in physical interactive playgrounds. *User Modeling and User-Adapted Interaction*, 18(1):207–243, 2008. [63](#), [117](#), [130](#), [181](#)
- G.N. Yannakakis, H.P. Martínez, and A. Jhala. Towards affective camera control in games. *User Modeling and User-Adapted Interaction*, 20(4):313–340, 2010. [27](#), [48](#)

- X. Yao. Evolutionary artificial neural networks. *International Journal of Neural Systems*, 4(03):203–222, 1993. [61](#)
- D. Yu and A. Hull, 2009. Spelunky, Independent. [35](#), [39](#)
- H. Yu and T. Trawick. Personalized procedural content generation to minimize frustration and boredom based on ranking algorithm. In *Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2011. [12](#)
- Mohammed J.. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1):31–60, 2001. [65](#), [67](#)
- Z. Zeng, M. Pantic, G.I. Roisman, and T.S. Huang. A survey of affect recognition methods: Audio, visual, and spontaneous expressions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(1):39–58, 2009. [19](#), [22](#)
- Q. Zhao and S.S. Bhowmick. Sequential pattern mining: A survey. *ITechnical Report CAIS Nanyang Technological University Singapore*, pages 1–26, 2003. [67](#)