Modeling Player Experience in Super Mario Bros

Chris Pedersen, Julian Togelius and Georgios N. Yannakakis

Abstract— **This paper investigates the relationship between level design parameters of platform games, individual playing characteristics and player experience. The investigated design parameters relate to the placement and sizes of gaps in the level and the existence of direction changes; components of player experience include fun, frustration and challenge. A neural network model that maps between level design parameters, playing behavior characteristics and player reported emotions is trained using evolutionary preference learning and data from 480 platform game sessions. Results show that challenge and frustration can be predicted with a high accuracy (77.77% and 88.66% respectively) via a simple single-neuron model whereas model accuracy for fun (69.18%) suggests the use of more complex non-linear approximators for this emotion. The paper concludes with a discussion on how the obtained models can be utilized to automatically generate game levels which will enhance player experience.**

Keywords: Platform games, player satisfaction, content creation, fun, preference learning, entertainment modeling, neuroevolution

I. Introduction

Numerous theories exist regarding what makes computer games fun, as well as which aspects contribute to other types of player experience such as challenge, frustration and immersion [1], [2], [3], [4], [5]. These theories have originated in different research fields and in many cases independently of each other (however, there is substantial agreement on several counts, e.g. regarding the importance of *challenge* and *learnability* for making a game fun). While useful high-level guidance for game design, none of these theories is quantitative — derived models of player experience are not mathematical expressions — and they tend to apply to games in general rather than specific aspects of specific games. This means that if we want to develop algorithms that design or adapt games (or aspects of games) *automatically*, we have to make several auxiliary assumptions in order to achieve the necessary specificity and preciseness of our models.

It seems clear that we need empirical research on particular games to acquire such models. Recently, research in player satisfaction modeling has focused on empirically measuring the effects on player experience of changing various aspects of computer games, such as non-player character (NPC) playing styles in the Pac-Man game [6]. Similarly, efficient quantitative models of player satisfaction have been constructed using in-game data, questionnaires and physiological measurements in augmented-reality games [7].

At the same time, a parallel research direction aims to find methods for automatically generating entertaining game content. Automatic content generation is likely to be of great importance to computer game development in the future; both offline, for making the game development process more efficient (design of content such as environments 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. These efforts see some aspect of a game as variable, define a fitness ("goodness") function based on a theory of player satisfaction, and use a learning or optimization algorithm to change the variable aspect of the game so as to become more "fun" according to some definition. The literature on this is so far scarce, as it is a new research direction. The aspects of games that have been optimized include:

- Environments, such as tracks for racing games [8] and levels for platform games [9].
- Narrative [10]
- Rules for board games [11], [12] and Pac-Man-like games [13].

What the above studies have in common is that the fitness or cost functions used for optimization have been somewhat arbitrary, in that they have been based on intuition in combination with some qualitative theory of player experience. Optimization of game aspects based on empirically derived models have so far been limited to the impact of NPC behavior [6] and the adjustment of NPC behavioral parameters for maximizing satisfaction in games [14]. To the best of our knowledge, game content such as rules or environments has not been generated based on empirically derived models. We consider better modeling of player experience to be of utmost importance for making automatic content generation techniques more sophisticated and usable.

The work we describe in this paper is concerned with the construction of computational models of player experience, derived from gameplay interaction, which can be used as fitness functions for game content generation. We use a modified version of a classic platform game for our experiments and collect player data through the Internet.

In the following, we describe the game used for our experiments; which player interaction data was collected and how; the preference learning method we used to construct player experience models; how feature selection was used to reduce the number of features used in the model; results of an initial statistical analysis; and the results of training nonlinear perceptrons to approximate the functions mapping between selected gameplay features and aspects of player experience. Finally, we discuss how the induced models will be used for automatically generating game content. This paper significantly extends [15] in which only the core ideas

The authors are with the Center for Computer Games Research, IT University of Copenhagen, Rued Langgaards Vej 7, DK-2300 Copenhagen S, Denmark. Emails: gammabyte@gmail.com, {juto, yannakakis}@itu.dk

of the methodology proposed are outlined.

II. Testbed platform game

The test-bed platform game used for our studies is a modified version of Markus Persson's *Infinite Mario Bros* (see Fig. 1) which is a public domain clone of Nintendo's classic platform game *Super Mario Bros*. The original Infinite Mario Bros is playable on the web, where Java source code is also available[1].

The gameplay in Super Mario Bros consists of moving the player-controlled character, Mario, through two-dimensional levels, which are viewed sideways. Mario can walk and run to the right and left, jump, and (depending on which state he is in) shoot fireballs. Gravity acts on Mario, making it necessary to jump over holes (or gaps) to get past them. Mario can be in one of three states: Small (at the beginning of a game), Big (can crush some objects by jumping into them from below), and Fire (can shoot fireballs).

The main goal of each level is to get to the end of the level, which means traversing it from left to right. Auxiliary goals include collecting as many as possible of the coins that are scattered around the level, clearing the level as fast as possible, and collecting the highest score, which in part depends on the number of collected coins and killed enemies.

The presence of gaps and moving enemies are the main challenges of Mario. If Mario falls down a gap, he loses a life. If he touches an enemy, he gets hurt; this means losing a life if he is currently in the Small state, whereas if he is in the Big and Fire state he shifts to Small and Large state respectively. However, if he jumps so that he lands on the enemy from above, the outcome is dependent on the enemy: Most enemies (e.g. goombas, fireballs) die from this treatment; others (e.g. piranha plants) are not vulnerable to this and proceed to hurt Mario; finally, turtles withdraw into their shells if jumped on, and these shells can then be picked up by Mario and thrown at other enemies to kill them.

Certain items are scattered around the levels, either out in the open, or hidden inside blocks of brick and only appearing when Mario jumps at these blocks from below so that he smashes his head into them. Available items include coins which can be collected for score and for extra lives (every 100 coins), mushrooms which make Mario grow Big if he is currently Small, and flowers which make Mario turn into the Fire state if he is already Big.

No textual description can fully convey the gameplay of a particular game. Only some of the main rules and elements of Super Mario Bros are explained above; the original game is one of the world's best selling games, and still very playable more than two decades after its release in the mid-eighties. Its game design has been enormously influential and inspired countless other games, making it a good experiment platform for player experience modeling.

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 (such as blocks, gaps and opponents) according to certain heuristics. In our modified version of Infinite Mario Bros most of the randomized placement of level features is fixed since we concentrate on a few selected game level parameters that affect game experience.

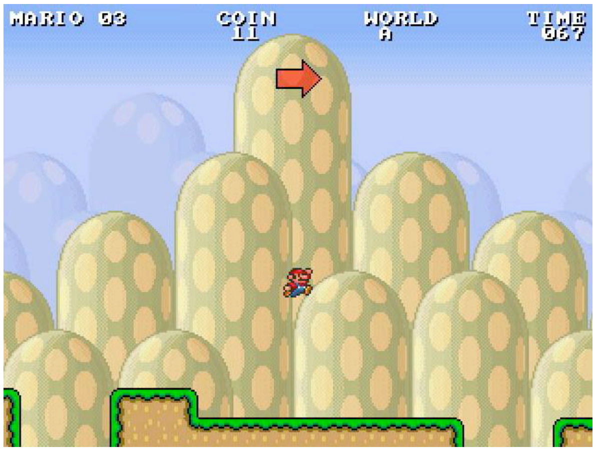


Fig. 1. Test-bed game screenshot, showing Small Mario jumping over a piece of flat terrain surrounded by two gaps.

III. Data collection

Before any modeling could take place we needed to collect data to train the model on. We collected three types of data from hundreds of players over the Internet:

1) Controllable features of the game, i.e. the parameters used for level generation, and affecting the type and difficulty of the level. These were varied systematically to make sure all variants of the game were compared.
2) Gameplay characteristics, i.e. how the user plays the game. We measured statistical features such as how often and when the player jumped, ran, died etc. These features cannot be directly controlled by the game as they depend solely on the player's skill and playing style in a particular game level.
3) The player's experience of playing the game, measured through a 4-alternative forced choice questionnaire administered after playing two pairs of games with different controllable features and asking the players to rank the games in order of emotional preference.

Below we describe in detail which features were collected for each type of data.

A. *Controllable features*

We modified the existing level generator to create levels according to four controllable parameters presented below. Three of these parameters are dedicated to the number, width and placement of gaps. The fourth parameter turns a new function, the direction switch, on or off.

- The number of gaps in the level, G.
- The average width of gaps, $E\{G_w\}$.
- The spatial diversity of gaps which is measured by the entropy of the number of gaps appearing in a number of

[1] http://www.mojang.com/notch/mario/

G (equally-spaced) segments of the level. The entropy of gap-placements H_g in the G segments is calculated and normalized into $[0,1]$ via (1):

$$H_g = \left[-\frac{1}{logG} \sum_{i=1}^{G} \frac{g_i}{G} log\left(\frac{g_i}{G}\right) \right] \qquad (1)$$

where g_i is the number of gap-placements into level segment i. If the gaps are placed in all G level segments uniformly then $g_i = 1$ for all G parts and H_g will be 1; if all gaps are placed in one level segment H_g is zero. This controllable feature provides a notion of unpredictability of gaps and, therefore, jumps in the level. Unpredictability has been identified as an important factor of playing experience [16].

- Number of direction switches, S. No direction switch means that the player needs to move from left to right in order to complete the level, as in the original Super Mario Bros. If one or more direction switch is present, the level direction will be mirrored at certain points, forcing the player to turn around and go the other way, until reaching the end of the level or the next switch.

The selection of these particular controllable features was done after consulting game design experts, and with the intent to find features which where common to most, if not all, platform games.

Two states (low and high) for each of the four controllable parameters above are investigated. The combinations of these states result in $2^4 = 16$ different variants of the game which are used in the user study designed. In the Super Mario Bros game investigated here the number of coins, opponents, coin blocks, powerup blocks and empty blocks are fixed to 15, 3, 4, 2, and 8 respectively.

B. Gameplay features

Several statistical features are extracted from playing data which are logged during gameplay and include game completion time, time spent on various tasks (e.g. jumping, running), information on collected items (e.g. type and amount), killed enemies (e.g. type, amount, way of killing) and information on how the player died. The choice of those specific statistical features is made in order to cover a decent amount of Super Mario Bros playing behavior dynamics. In addition to the four controllable game features that are used to generate Super Mario Bros levels presented earlier, the following statistical features are extracted from the gameplay data collected and are classified in five categories: jump, time, item, death, kill and misc.

Jump: difference between the total number of jumps, J, minus the number of gaps, G; number of jumps over gaps or without any purpose (e.g. to collect an item, to kill an opponent), J'; difference between J' and the number of gaps, G; and a jump difficulty heuristic, J_d, which is proportional to the number of Super Mario deaths due to gaps, number of gaps and average gap width.

Time: completion time, t_c; playing duration of last life over total time spent on the level, t_{ll}; percentage of time that the player: is standing still, t_s, running, t_r, is on large Mario mode, t_l, is on fire Mario mode, t_f, is on powerup mode, t_p, is moving left, t_L, is moving right, t_R, and is jumping, t_j.

Item: number of collected items (coins, destroyed blocks and powerups) over total items existent in the level, n_I; number of times the player kicked an opponent shell, n_s; number of coins collected over the total number of coins existent in the level, n_c; number of empty blocks destroyed over the total number of empty blocks existent in the level, n_{eb}; number of coin blocks pressed over the total number of coin blocks existent in the level, n_{cb}; number of powerup blocks pressed over the total number of powerup blocks existent in the level, n_p; and the sum of all blocks pressed or destroyed over the total number of blocks existent in the level $n_b = n_{eb} + n_{cb} + n_p$.

Death: number of times the player was killed: by an opponent, d_o; by jumping into a gap, d_g; by jumping into a gap over the total number of deaths d_j.

Kill: number of opponents died from stomping over the total number of kills, k_s; number of opponents died from fire-shots over the total number of kills, k_f; total number of kills over total number of opponents, k_T; number of opponent kills minus number of deaths caused by opponents, k_P; and number of cannonballs killed, k_c.

Misc: number of times the player shifted the mode (Small, Big, Fire), n_m; number of times the run button was pressed, n_r; number of ducks, n_d; number of cannonballs spawned, n_{cs}; and whether the level was completed or not C (boolean).

C. Reported player experience and experimental protocol

We designed a game survey study to solicit pairwise emotional preferences of subjects playing different variants of the test-bed game by following the experimental protocol proposed in [7]. Each subject plays a predefined set of four games in pairs: a game pair of game A and game B played in both orders. The games played differ in the levels of one or more of the four controllable features presented previously. For each completed pair of games A and B, subjects report their emotional preference using a 4-alternative forced choice (4-AFC) protocol:

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

Where E is the emotional state under investigation and contains *fun*, *challenging*, *boring*, *frustrating*, *predictable* and *anxious*. The selection of these six states is based on their relevance to computer game playing and their popularity when it comes to game-related user studies [17]. In this initial investigation of player experience we focus only on three emotions: *fun*, *challenge* and *frustration*.

Data is collected over the Internet. Users are recruited via posts on blogs and mailing lists and directed to a web page containing a Java applet implementing the game and questionnaire[2]. As soon as the four games are played and the questionnaire is completed, all the features (controllable, gameplay and player experience) are saved in a database at the server hosting the website and applet. Data collection is still in progress and at the moment of writing, 181 subjects have participated in the survey experiment. The minimum number of experiment participants required is determined by $C_2^{16} = 120$, this being the number of all combinations of 2 out of 16 game variants. The experimental protocol is designed in such a way that at least 2 preference instances should be obtained for each pair of the 16 game variants played in both orders (1 preference instance per playing order). The analysis presented in this paper is based on the 240 game pairs (480 game sessions) played by the first 120 subject participants.

IV. Preference Learning for Modeling Player Experience

Based on the data collected in the process described above, we try to approximate the function from gameplay features (e.g. number of coins gathered) and controllable game level features (e.g. number of gaps) to reported emotional preferences using neuroevolutionary preference learning.

The data is assumed to be a very noisy representation of the underlying function, given the high level of subjectivity of human preferences and the expected variation in playing styles. Together with the limited amount of training data, this makes overfitting a potential hazard and mandates that we use a robust function approximator. We believe that a non-linear function such as an artificial neural network (ANN) is a good choice for approximating the mapping between reported emotions and input data. Thus, a simple single-neuron (perceptron) is utilized for learning the relation between features (ANN inputs) — selected from feature selection schemes presented in Section V — and the value of the investigated emotional preference (ANN output) of a game. The main motivation for using a single neuron instead of a multi-layered perceptron (MLP) in this study is that we want to be able to analyze the trained function approximator. While an MLP can potentially approximate the function investigated with a higher accuracy, it is much easier for a human to understand the obtained function when represented as a single-neuron ANN.

The single neuron uses the sigmoid (logistic) activation function; connection weights take values from -5 to 5 to match the normalized input values that lie in the [0, 1] interval. Since there are no prescribed target outputs for the learning problem (i.e. no differentiable output error function), ANN training algorithms such as back-propagation are inapplicable. Learning is achieved through artificial evolution by following the preference learning approach presented in [18]. A generational genetic algorithm (GA) is implemented, using a fitness function that measures the difference between the subject's reported emotional preferences and the relative magnitude of the corresponding model (ANN) output.

V. Feature Selection

We would like our model to be dependent on as few features as possible, both to make it easier to analyze, and to make it more useful for incorporation into future games for purposes of e.g. content creation. Therefore, feature selection is utilized to find the feature subset that yields that most accurate user model and save computational effort of exhaustive search on all possible feature combinations. The quality of the predictive model constructed by the preference learning outlined above depends critically on the set of input data features chosen. Using the extracted features described earlier the *n best individual feature selection* (nBest), the *Sequential Forward Selection* (SFS) and the *Perceptron Feature Selection* (PFS) schemes are applied and compared.

A. nBest

nBest feature selection ranks the features used individually in order of model performance; the chosen feature set of size n is then the first n features in this ranking. The nBest method is used for comparative purposes, being the most popular technique for feature selection.

B. SFS

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 SFS method has been successfully applied to a wide variety of feature selection problems, yielding high performance values with minimal feature subsets [7], [19]

C. PFS

The third method we investigate is Rosenblatt's perceptron as a methodology for selecting appropriate feature subsets. Our algorithm which is similar to [20] is adjusted to match preference learning problems. Thus, the perceptron used employs the sigmoid activation function in a single output neuron. The ANN's initial input vector has the size of the number of features examined. The perceptron feature selection (PFS) procedure is as follows:

Step 1 Use artificial evolution to train the perceptron on the pairwise preferences (see Section IV). Performance of the perceptron is evaluated through 3-fold cross-validation. The initial input vector consists of all features extracted $\mathcal{F}$ (40 in this paper).

Step 2 Eliminate all features $\mathcal{F}'$ whose corresponding absolute connection weight values are smaller than $E\{|\mathbf{w}|\} - \sigma\{|\mathbf{w}|\}$, where $\mathbf{w}$ is the connection weight vector.

Step 3 If $\mathcal{F}' = \emptyset$ continue to Step 4, otherwise use the remaining features and go to Step 1.

[2] The game and questionnaire are available at www.bluenight.dk/mario.php

Step 4 Evaluate all feature subsets obtained using the neuro-evolution preference learning approach presented in Section IV.

Note that all three methods are incomplete. Neither is guaranteed to find the optimal feature set since neither searches all possible combinations (they are all variants of hill-climbing). To evaluate the performance of each input feature subset, the available data is randomly divided into thirds and training and validation data sets consisting of 2/3 and 1/3 of the data respectively are assembled. The performance of each user model is measured through the average classification accuracy of the model in three independent runs using the 3-fold cross-validation technique on the three possible independent training and validation data sets. Since we are interested in the minimal feature subset that yields the highest performance we terminate the SFS selection procedure when an added feature yields equal or lower validation performance to the performance obtained without it. On the same basis, we store all feature subsets selected by PFS and explore the highest performing subset starting with the smallest feature subset generated.

VI. Statistical Analysis

This section describes testing for correlations between playing order, controlled features and gameplay features and the reported emotions of fun, challenge and frustration.

To check whether the order of playing Super Mario game variants affects the user's judgement of emotional preferences, we follow the order testing procedure described in [6] which is based on the number of times that the subject prefers the first or the second game in both pairs. The statistical analysis shows that order of play does not affect the emotional preferences of fun and frustration; however a statistically significant effect is observed in challenge preferences (p-value $= 0.006$). The effect reveals a preference for the second game played which implies the existence of random noise in challenge preference expression. On the other hand, the insignificant order effects of fun and frustration, in part, demonstrate that effects such as a user's possible preference for the very first game played and the interplay between reported emotions and familiarity with the game are statistically insignificant.

More importantly, we performed an analysis for exploring statistically significant correlations between subject's expressed preferences and extracted features. Correlation coefficients are obtained through $c(\mathbf{z}) = \sum_{i=1}^{N_s}\{z_i/N_s\}$, where N_s is the total number of game pairs where subjects expressed a *clear preference* for one of the two games (e.g. $A \succ B$ or $A \prec B$) and $z_i = 1$, if the subject preferred the game with the larger value of the examined feature and $z_i = -1$, if the subject chooses the other game in the game pair i. Note that, N_s is 161, 189 and 151 respectively, for reported fun, challenge and frustration

The variation of the N_s numbers above indicates, in part, the difficulty in expressing a clear emotional preference on different game variants. The percentage of $A \succ B$ and $A \prec B$ selection occurrences over all 240 preference instances for different emotional states varies from 78.7% (challenge) to 62.9% (frustration). These percentages provide some first evidence that the selected game level and rule parameters have an dissimilar impact on the emotional states investigated. For instance, challenge appears to be very much affected by varying the selected parameters whereas frustration, on the contrary, does not appear as an emotion which is directly affected by variations in the game.

TABLE I
Top ten statistically significant (p-value $< 1\%$) correlation coefficients between reported emotions and extracted features.

Fun		Challenge		Frustration	
n_s	0.345	C	-0.600	C	-0.826
n_{cb}	0.311	n_p	-0.480	n_p	-0.815
k_T	0.256	d_j	0.469	n_{cb}	-0.688
n_r	0.253	d_g	0.447	d_g	0.578
t_L	0.237	J_d	0.439	d_j	0.564
k_P	0.222	$E\{G_w\}$	0.409	n_I	-0.544
t_r	0.192	n_d	-0.368	t_s	0.520
		t_{ll}	-0.312	k_f	-0.515
		n_{cb}	-0.292	t_{ll}	-0.513
		G	-0.287	n_c	-0.511

A. Fun

Statistically significant correlations are observed between reported fun and seven features: number of times the player kicked a turtle shell, proportion of coin blocks that were "pressed" (jumped at from below), proportion of opponents that were killed, number of times the run button was pressed, proportion of time spent moving left, number of enemies killed minus times died, and proportion of time spent running. All of these were positive correlations.

Such correlations draw a picture of most players enjoying a fast paced game that includes near-constant progress, plenty of running, many enemies killed and many coins collected from bouncing off the coin blocks. One might argue that this picture fits with the concept of *Flow*, in that the player makes unhindered progress [3]. However, the Flow concept also includes a certain level of challenge, and no features that signify challenge are associated with fun in this case. It might be that players enjoy when the game is easy — at least when they only play a single level of the game.

The feature that correlates the most with fun preferences is kicking turtle shells. Kicking a turtle shell is a simple action which often results in the unfolding of a relatively complex sequence of events, as the shell might bounce off walls, kill enemies, fall into gaps etc. The fun inherent in setting of complex chains of events with simple actions is something many players can relate to and which features prominently in many games, but which is to our knowledge not part of any of the "established" theories of what makes games fun.

B. Challenge

Eighteen features are significantly correlated with challenge. The ten most highly correlated are (+/− in parenthesis signifies positive or negative correlation): whether the level was completed (−), proportion of power-up blocks pressed (−), proportion of Mario deaths that were due to falling into a gap (+), number of times Mario died from falling into a gap (+), jump difficulty (+), average width of gaps (+), number of times Mario ducked (−), proportion of time spent in the last life (−), proportion of coin blocks that were pressed (−), and the number of gaps (−). In addition, a weaker but still significant positive correlation was found between gap entropy, H_g, and challenge.

A first observation is that it is obviously much easier to predict challenge than to predict fun — many more features are significantly correlated, and the correlations are stronger. It also seems that challenge is somehow orthogonal to fun, as almost none of the features that are correlated with challenge are correlated with fun. The exception is the proportion of coin blocks pressed, but while this feature is positively correlated with fun it is negatively correlated with challenge. (This is somewhat expected: if the level is so hard that the player has to struggle to survive it, she does not have time to make detours in order to collect more coins.)

Most of the correlations are easy to explain. That a level is perceived as less challenging if you complete it should not come as a surprise to anyone. Likewise, we can understand that players think a level is hard when they repeatedly die from falling into gaps. Three particularly interesting correlations are those between the controllable features and challenge: increase in gap width, $E\{G_w\}$, and gap entropy, H_g, imply increased challenge whereas increased number of gaps, G, implies a linear decrease of challenge. These effects suggest that challenge can be controlled to a degree by changing the number, width and distribution of gaps.

The correlation with number of ducks would have been easy to explain — if it was positive. The main reason for ducking in Super Mario Bros (at least in the tested levels) is to avoid cannonballs. To the authors, cannons are perceived as some of the most difficult elements on a level. However, players reported lower challenge on levels where they ducked many times. We have yet to find an explanation for this.

C. Frustration

Twenty-eight features are significantly correlated with frustration, and some of the correlations are extremely strong. Of the top ten correlated features, most are also in the top ten list for features correlated with challenge, and correlated in the same way. The exceptions are proportion of collected items (−), time spent standing still (+), proportion of killed opponents that were killed with fireballs (−), and proportion of coins collected (−).

From these new features, it seems that a frustrated player is most likely one that spends time standing still and thinking about how to overcome the next obstacle; is far too busy overcoming obstacles to collect coins and power-ups; and as a result of not collecting power-ups is rarely in the Fire Mario state (necessary to shoot fireballs). But frustration can also be very well predicted from not winning the level and from falling into gaps often, just like challenge.

D. Controllable features and intra-correlations

When only looking at linear correlations, it would appear that fun is not connected to any of the four controllable features. Fun is also less strongly correlated with gameplay features than is the case for challenge and frustration. The latter two emotions are easier to model with linear models, and are also strongly correlated with controllable features, namely gap entropy, gap width and number of gaps.

The three emotions are also all significantly (p-value $<$ 0.001) correlated to each other. The correlation coefficient $c(\mathbf{z})$ between challenge and fun and between challenge and frustration is 0.346 and 0.462, respectively, while the corresponding correlation between fun and frustration is -0.284. The positive effect of challenge on fun and frustration, combined with the negative effect of fun on frustration indicate the nonlinearity (and possibly complexity) of those emotions' interrelationships.

VII. Nonlinear Preference Modelling

The correlations calculated above provide linear relationships between individual features and reported emotions. However, these relationships are most likely more complex than can be captured by linear models. The aim of the analysis presented below is to construct non-linear computational models for reported emotions and analyze the relationship between the selected features and expressed preferences.

For this purpose we evolve weights for nonlinear perceptrons as described in Section IV. The weights of the highest performing networks are presented in Table II. All evolved networks performed much better than networks with random weights, which reached chance level prediction accuracy.

TABLE II

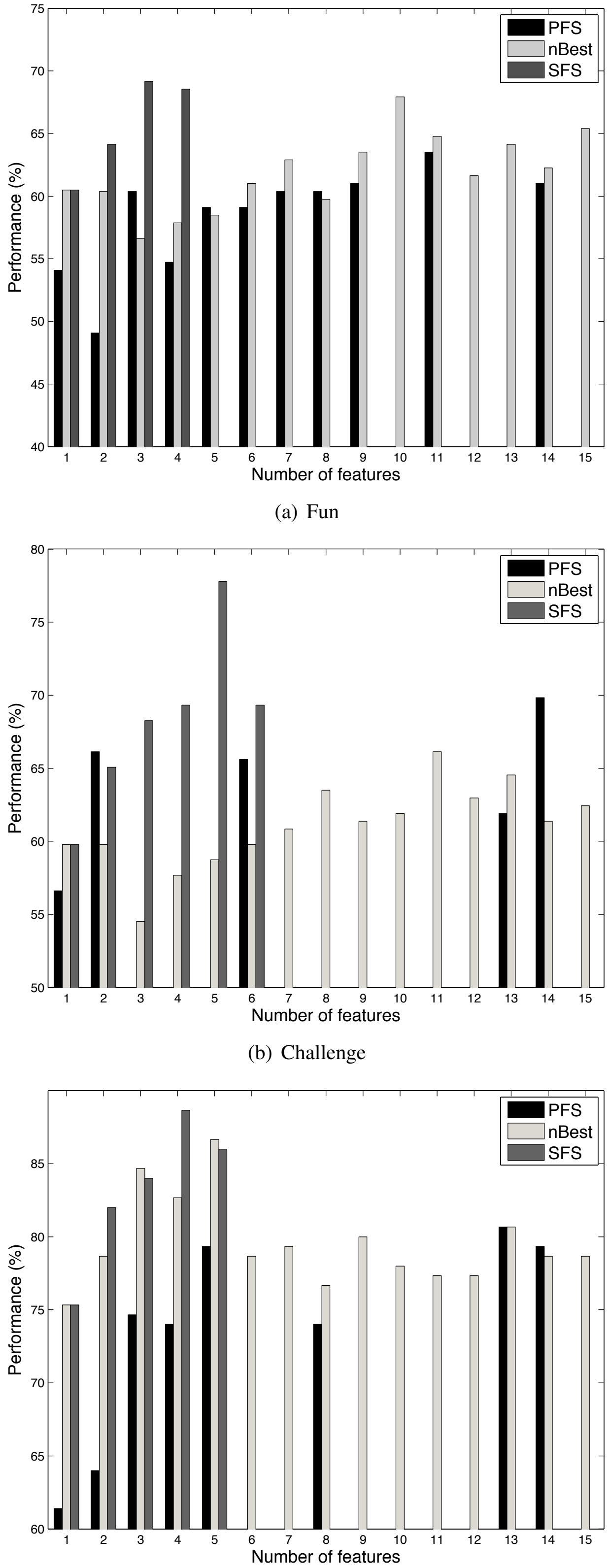
Learning from Preferences: Features and Corresponding Connection Weights for Highest Performing ANNs

Fun		Challenge		Frustration	
t_L	4.905	t_s	-1.703	t_s	3.267
k_s	0.942	J_d	3.805	t_{ll}	-1.851
L	-3.873	n_{eb}	-1.502	J_d	-0.995
		k_c	1.073	d_g	0.233
		k_s	-0.189		

A. Fun

In the comparison between the three different selection mechanisms applied it is evident that SFS has advantages over nBest and PFS for fun preferences (see Fig. 2(a)). nBest achieves a satisfactory performance (67.92%) but requires 10 features as inputs to the ANN. PFS generates the lowest classification accuracies; its best network has an accuracy of 63.52% with a selected subset of 11 input features.

The best obtained perceptron model of fun preferences is designed by SFS. This model achieves a performance



(a) Fun

(b) Challenge

(c) Frustration

Fig. 2. Performance comparison of feature selection mechanisms on emotional preferences.

of 69.18% which is with a selected feature subset of size three. The selected perceptron input vector consists of the time spent moving left t_L, the number of opponents died from stomping over the total number of kills, k_s, and the controllable *switching* feature which is defined as the percentage of level played in the left direction L[3].

Fun is the least correlated of the three modeled emotions, and the hardest to model with a nonlinear perceptron as well. Still, it's remarkable that this complex emotion can be predicted to a moderate degree simply by observing that Mario keeps running left and kills enemies by stomping.

B. Challenge

The best-performing ANN for challenge prediction has an accuracy of 77.77%. It is more complex than the best fun predictor, using five features: time spent standing still $(-)$, jump difficulty $(+)$, proportion of coin blocks pressed $(-)$, number of cannonballs killed $(-)$ and proportion of kills by stomping $(-)$. While the jump difficulty heuristic has the largest corresponding weight — a testament to the central role of gap placement and size for challenge — it is also the only feature related to gaps used by this model, pointing to the adequateness of this particular heuristic.

C. Frustration

Our best evolved ANN for predicting frustration has an accuracy of 88.66%. We can predict with near-certainty whether the player is frustrated by the current game by just calculating the time spent standing still $(+)$, the proportion of time spent on last life $(-)$, the jump difficulty $(-)$, and the proportion of deaths due to falling in gaps $(+)$.

Somewhat surprisingly, time spent standing still counts *against* challenge, whereas it is a strong *positive* predictor of frustration. This observation could be valuable if trying to design a feedback system that keeps the game challenging but not frustrating. Another feature that has different effect on challenge and frustration is jump difficulty, where frustration is connected with *lower* jump difficulty. Maybe the player gets frustrated by falling into gaps that she knows are not that hard.

That the player feels frustrated when dying after a short time during his last life is understandable — many players feel that their last attempt should be their best. Additionally, a high frustration level can cause the player to care less about the game and play worse in her final life.

VIII. Discussion

While we have found relatively good predictors for all three emotions, two problems remain: the predictions (at least for fun and challenge) are still not as good as we would like them to be, and we cannot reliably predict fun from *controllable* features. As controllable features (such as level

[3] The L feature is there to correct for the fact that when the level direction switches, Mario moves right rather than left to move forward, and so t_L is diminished. This points to an oversight on our part when designing the gameplay features: we should have measured the time spent moving *towards the end of the level* rather than moving left.

design parameters) are those that we can vary, and therefore those that can be optimized by evolution or other global optimizers, we need to be able to predict emotions at least partly from controllable features.

This points to the need for better models and/or features. First of all, we will try to induce multi-layer perceptron models of emotional preferences. MLPs have the advantage of universal approximation capacity; in particular, combinatorial relationships (such as XOR) can be represented. We might very well have a situation were one controllable feature (such as gap width) can be both negatively and positively connected with an emotion (such as frustration) depending on the player's playing style, as measured through gameplay features (such as number of jumps). Such relationships can be captured by MLPs but not by nonlinear perceptrons.

Data collection is continuing at the time of writing, and probably at the time of reading (the reader is welcome to contribute by visiting the project's web site), the new data will be used to improve the accuracy of our predictions.

Depending on the success of finding predictors partly dependent on controllable features, we might need to design new controllable features or revise the existing ones. New features might include the number and type of enemies, the existence of dead ends in the level (forcing backtracking) etc.

After good models have been learned, evolutionary algorithms will be used to optimize the level design parameters (relating to gaps and switches) for different objectives. We hope to, this way, be able to generate levels that tailor the playing experience according to the needs of the game design (e.g. a challenging level combined with a non-frustrating experience). The success of our optimization attempts will be validated with further user studies.

Another question concerns the generality of the results gathered here — do they apply to just the players and the particular game tested here, or do they have wider applicability? We venture that, as Super Mario Bros more or less defined the platform game genre, the results apply to some extent to all games of the same genre. Further, the population of experimental subjects is believed to be very diverse, but this needs to be verified. A possible critique is that the emotions reported are those that have been elicited after only a few minutes of play. It is possible that challenge or variety (gap entropy) would factor in more if play sessions were longer, so subjects would have had a chance of getting bored with the game.

IX. Conclusions

We designed a user study focused on a version of the Super Mario Bros platform game, in which a population of subjects played in a number of different versions (mainly differing in the game environments encountered). Controllable features and emergent gameplay features were correlated with reported emotions during gameplay. We found a large number of statistically significant correlations, and were able to train good predictors of player emotions using preference learning and neuroevolution. These results will be improved upon and form the basis for attempts to automatically generate environments for this game using artificial evolution with the induced player experience models as fitness functions.

Acknowledgments

The authors would like to thank Aki Järvinen and Markus Persson for insightful discussions, and all subjects that participated in the experiments.

References

[1] C. Bateman and R. Boon, *21st Century Game Design*. Charles River Media, 2005.

[2] K. Isbister and N. Schaffer, *Game Usability: Advancing the Player Experience*. Morgan Kaufman, 2008.

[3] M. Csikszentmihalyi, *Flow: the Psychology of Optimal Experience*. Harper Collins, 1990.

[4] R. Koster, *A theory of fun for game design*. Paraglyph press, 2005.

[5] J. Juul, *Half-real*. MIT Press, 2005.

[6] G. N. Yannakakis and J. Hallam, "Towards optimizing entertainment in computer games," *Applied Artificial Intelligence*, vol. 21, pp. 933–971, 2007.

[7] ——, "Entertainment modeling through physiology in physical play," *International Journal of Human-Computer Studies*, vol. 66, pp. 741–755, 2008.

[8] J. Togelius, R. De Nardi, and S. M. Lucas, "Towards automatic personalised content creation in racing games," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2007.

[9] K. Compton and M. Mateas, "Procedural level design for platform games," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*, 2006.

[10] M. J. Nelson, C. Ashmore, and M. Mateas, "Authoring an interactive narrative with declarative optimization-based drama management," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*, 2006.

[11] C. Browne, "Automatic generation and evaluation of recombination games," Ph.D. dissertation, Queensland University of Technology, Brisbane, Australia, 2008.

[12] J. Marks and V. Hom, "Automatic design of balanced board games," in *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*, 2007, pp. 25–30.

[13] J. Togelius and J. Schmidhuber, "An experiment in automatic game design," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 2008.

[14] G. N. Yannakakis and J. Hallam, "Real-time Game Adaptation for Optimizing Player Satisfaction," *IEEE Transactions on Computational Intelligence and AI in Games*, 2009, (to appear).

[15] C. Pedersen, J. Togelius, and G. Yannakakis, "Optimization of platform game levels for player experience," in *Proccedings of the Artificial Intelligence and Interactive Digital Entertainment International Conference (AIIDE)*, 2009.

[16] T. W. Malone, "What makes computer games fun?" *Byte*, vol. 6, pp. 258–277, 1981.

[17] R. L. Mandryk and M. S. Atkins, "A Fuzzy Physiological Approach for Continuously Modeling Emotion During Interaction with Play Environments," *International Journal of Human-Computer Studies*, vol. 65, pp. 329–347, 2007.

[18] G. N. Yannakakis and J. Hallam, "Game and Player Feature Selection for Entertainment Capture," in *Proceedings of the IEEE Symposium on Computational Intelligence and Games*. Hawaii, USA: IEEE, April 2007, pp. 244–251.

[19] G. N. Yannakakis, M. Maragoudakis, and J. Hallam, "Preference Learning for Cognitive Modeling: A Case Study on Entertainment Preferences," *IEEE Systems, Man and Cybernetics; Part A: Systems and Humans*, 2009, (to appear).

[20] M. Mejia-Lavalle and G. Arroyo-Figueroa, "Power System Database Feature Selection Using a Relaxed Perceptron Paradigm," in *Proceedings of 5th Mexican International Conference on Artificial Intelligence, LNCS*. Springer Berlin/Heidelberg, 2006, pp. 522–531.