Personalized Procedural Content Generation to Minimize Frustration and Boredom Based on Ranking Algorithm

Hong Yu and Tyler Trawick
Georgia Institute of Technology
85 Fifth Street NW, Atlanta, GA 30308
{hyu8,gtpd}@gatech.edu

Abstract

A growing research community is working towards procedurally generating content for computer games and simulation applications with various player modeling techniques. In this paper, we present a two-step procedural content generation framework to minimize players' frustration and/or boredom according to player feedback and gameplay features. In the first step, we dynamically categorize the player styles based on a simple questionnaire beforehand and the gameplay features. In the second step, two player models (frustration & boredom) are built for each player style category. A ranking algorithm is utilized for player modeling to address two problems inherent in player feedback: inconsistency and inaccuracy. Experiment results on a testbed game show that our framework can generate less boring/frustrating levels with very high probabilities.

Introduction

Both offline and online procedural content generation (PCG) research have attracted great interest in recent years, for both gaming and simulation applications. Offline PCG has been applied to make the development process more efficient, while online PCG has been applied to improve the replay value and adapt difficulty level dynamically (Doran and Parberry 2010; Hecker et al. 2008; Pedersen, Togelius, and Yannakakis 2009). The research on personalized and player-adaptive procedural content generation is an emerging field in recent years (Yannakakis and Togelius 2011; Schuytema 2007). This paper seeks to procedurally generate personalized game levels in order to give players a better experience.

Player experience is critical in computer games. Our assumption is that a game that is much too difficult for a given player's skill level will cause her to be frustrated. Likewise, a game that is much too easy will cause boredom. A game will be the most fun when it minimizes both of these, and thus an approximation of player enjoyment can be gained from measuring two factors: boredom and frustration. The claim underlying this research is the game flow theory (See Figure 1) which provides a framework explaining the relationship between the emotion boredom and frustration (Sweetser and Wyeth 2005; Csikszentmihalyi 1990; Murray and Arroyo 2002; Vygotsky 1986). The game flow describes the relationship between game difficulty and player ability. The ideal situation is achieved when the player stays in the optimal flow channel throughout the game, without stepping into the zone of frustration or boredom, resulting in a game that is less frustrated, less boring and more fun.

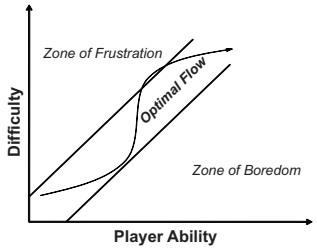

Figure 1: Graph illustration of the game flow theory. (Sweetser and Wyeth 2005)

In this paper we propose a two-step framework to attack this problem. In the first step, the players are classified into different player style categories according to *player features* which record their in-game behavior and a simple questionnaire collected beforehand. In the second step, we build two *player models* for every category based on *player feedback* and corresponding *gameplay features*: a frustration model and a boredom model. In this paper, the *player feedback* consists of frustration and boredom comparisons for pairs of levels the player has played. A ranking algorithm is utilized for player modeling in order to avoid the inaccuracy inherent in the players' feedback. With this framework, subsequent testing players will firstly be assigned to a category, and will then be presented with game levels generated from the player model for that category which are optimal in terms of minimizing frustration or boredom. In the experiments, we built a simple level game to test the framework.

Related Work

Procedural content generation has attracted increasing attention from computer game research area recently. The PCG can be used offline to help game designers and make the development process more efficient, such as game world or terrain generation (Parish and Mller 2001; Roden and Par-

berry 2004; Doran and Parberry 2010), character generation (Hecker et al. 2008), and so on. Such works typically focus on satisfying authorial intentions and constraints, and consequently do not explicitly model player behaviors. On the other hand, online PCG such as story generation (Riedl et al. 2008), level generation (Togelius, Nardi, and Lucas 2007; Shaker, Yannakakis, and Togelius 2010), and so on, usually include an explicit or implicit player model for procedural generation. This paper is a contribution to the latter efforts. Our work is also related to dynamic difficulty adjustment (Spronck, Sprinkhuizen-Kuyper, and Postma 2004), in which they change game difficulty by adjusting rule weights. In this paper we focus on adjusting frustration and boredom levels to give player a better experience.

Player experience modeling techniques usually fall into the following three categories according to (Yannakakis and Togelius 2011): subjective, objective, and gameplay based. Subjective modeling relies on the subjective feedback by players before, during or after gameplay (Tognetti et al. 2010). Very accurate models can be built based on these subjective data. But this approach is limited by the reliability of player feedback. Objective player modeling methods keep record of a number of real-time player reactions during gameplay, such as electrocardiography (ECG), electromyography (EMG) and so on, and study the relationship between these features and player emotion (Yannakakis, Martinez, and Jhala 2010). These features are more reliable but usually difficult to obtain, especially for commercial games (Yannakakis and Togelius 2011). The gameplay based approaches build the player model by collecting player features during gameplay, such as the player's death rate, weapon preference, and so on (Shaker, Yannakakis, and Togelius 2010; Pedersen, Togelius, and Yannakakis 2009; Hastings, Guha, and Stanley 2009). The gameplay based player modeling is the most computationally efficient and least intrusive among all these three but they usually require several strong assumptions that relate the player features to player preference (Thue et al. 2007). It is possible to combine them and build more efficient player models (Shaker, Yannakakis, and Togelius 2010). Our work can be viewed as a hybrid approach which combines the subjective and gameplay features. Neuro-evolution algorithms have been widely used for player modeling (Martinez, Hullett, and Yannakakis 2010; Pedersen, Togelius, and Yannakakis 2009). Martinez et al. use a similar two-step model to learn player preference. Here we apply a new player modeling algorithm in order to solve the uncertainty problems in player feedback.

AI Model

Our purpose is to generate personalized levels in order to minimize the frustration and/or boredom of the players. Different types of players usually find different things boring or frustrating, and thus we need to model different categories of players during the training process. Our AI model is composed of two steps: the player styles categorization and player modeling.

We classify all the *gameplay features* into two categories (Pedersen, Togelius, and Yannakakis 2009; Shaker, Yannakakis, and Togelius 2010): *controllable features* and *player features*. *Controllable features* control how the levels are generated, such as the parameters that affect the number of enemies and the difficulty of the levels. *Player features* record players' activities during gameplay, such as how many enemies the player killed, how many times the player died, how long did it take to finish the level, and so on. In the training process, all the gameplay features and the players' frustration and boredom feedbacks are collected. In the testing process, we generate the optimal combination of controllable features which are supposed to minimize the players' boredom/frustration levels.

Dynamic Player Style Categorization

Previous research on player modeling usually assumes several predefined player types and classifies new players according to these types (Peinado and Gervas 2004; Thue et al. 2007). This could possibly bring some problems, as these categories are based on pre-conceived notions of different player styles. It is difficult for the predefined categories to cover all possible player types, which could vary from game to game. We try to avoid the problem by acquiring the player style information from player features. In the training process, we cluster the player features of all the players into several categories which represent different player styles. Then in the testing process, a new player's style can be identified dynamically by computing its relationships to the cluster centers.

A naive Bayesian approach is applied to identify the player style. We use a soft clustering algorithm and assign the player to multiple categories with different probabilities, which are used as likelihood. Furthermore, we incorporate information from a simple questionnaire which is supposed to be related to the player styles. The questionnaire is treated as the prior knowledge and the player style category can be computed as the posterior with Bayesian theory as in Equation 1. The experiment results section will talk about the questionnaire in more detail.

$$p(C_i|x) \propto P(C_i)p(x|C_i) \tag{1}$$

where x is the player feature and C_i is player style category i. $P(C_i)$ is the prior probability for category C_i given player's response in the questionnaire. $p(x|C_i)$ is the likelihood computed from the soft clustering algorithm. In practice, we compute the prior $P(C_i)$ as in Equation 2.

$$\begin{aligned} P(C_i) &\doteq p(C_i|Q_1, Q_2, ...Q_n) \\ &\propto p(Q_1|C_i)p(Q_2|C_i)...p(Q_n|C_i)p_0(C_i) \end{aligned} \tag{2}$$

$Q_1, Q_2...Q_n$ represent the questions in the questionnaire and n is the total number of questions. Equation 2 takes the naive Bayesian assumption that all the questions are conditionally independent given the category i. $p_0(C_i)$ is the prior probability of the player in category C_i. $p_0(C_i)$ and $p(Q_j|C_i)$ can be computed directly from the training data by counting the corresponding appearing frequencies.

Ranking for Player Modeling

In the second step, we compute two player models for each category of players: a frustration model and a bore-

dom model. This is treated as a supervised learning problem: given the controllable features and player features, the player models are supposed to predict the corresponding boredom or frustration labels from the player's feedback.

However, there are two issues associating with the boredom/frustration labels which prevent us to treat the learning problem as a traditional classification or regression problem. The first issue is that these subjective labels are *inconsistent* among players. By inconsistent we mean that different players could have completely different labeling criterions. Suppose that we use the numbers 1 to 5 as labels for frustration, where a higher number corresponds to more frustration. A frustration label 3 given by player A does not necessarily mean that player A feels more frustrated than player B even if B labeled the same game level by 2. The direct comparison of the labels across player usually makes little sense. The second issue with the labels is that they are *inaccurate*. *Inaccurate* means that even the label values given by the same player are not accurate. For example, if the player feels a little frustrated, she could label the game level by 4, or 3 sometimes. In another case, when a player labels level one by 4 and level two by 3, she is supposed to feel more frustrated with level one than level two. But she is also likely to label the two levels by 4 and 2. In other words, it is inaccurate when the players try to quantitate their feelings into numbers.

In order to overcome the two difficulties resulted from the noise inherent in player feedback, we utilize a ranking algorithm for the player modeling. Notice that when the player leaves a higher frustration label for level A than level B, it is probable that she feels more frustrated with level A than B. In other words, the information from *relative comparisons* of labels given by the same player is more reliable than the label values themselves. Some ranking algorithms such as RankSVM (Joachims 2002) can be used to address the problem. It only takes relative comparisons between feature data points as input.

In the test process, we perform an exhaustive search in the level space and present the best levels to the player which are supposed to minimize frustration or boredom. The exhaustive search can be done by firstly generating all the possible combinations of controllable features. The corresponding ranking model built in the training step is used to rank all these feature combinations. The one obtaining the lowest ranking score (corresponds to the least boring or frustrating) is then selected.

The whole AI model can be summarized as follows. Given the features and player feedbacks in the training step, we perform:

- Cluster the features from all the players into different categories C_i.
- For every category i, compute $p_o(C_i)$ and $p(Q_k|C_i), k = 1,2...n$
- For every category i, train a boredom ranking model B_i and a frustration ranking model F_i.

Then in the testing step, after we obtain the testing player features x and the questionnaire:

Figure 2: The gameplay of the test-bed game.

- Compute $p(x|C_i)$ using a soft cluster.
- Compute $p(C_i|x)$ according to Equation 1 and 2. Identify the player category k.
- Use B_k or F_k to rank the controllable feature space and find the optimal controllable features to minimize boredom or frustration.

Testbed Game Description

We build a top-down level game similar to Gradius as our testbed game. Figure 2 shows a screen shot of the gameplay.

In the game, the main goal for the player is to traverse to the right end of the level. The player can move freely up, down, left and right in the game world. But unlike Gradius, we do not shift the game screen continuously. The whole level consists of several discrete screens, one of which is exhibited in Figure 2. Once the player has entered the next screen, she cannot come back to the previous one. The player cannot preview the terrain in the following screens.

The number and type of Non Player Characters (NPCs) in every screen are controlled by the controllable features. There are three types of NPCs in the current game: melee-attacking enemies, ranged-attacking enemies and the green balls which can kill the player by collision. The NPCs will attack the player if they feel threatened. Every NPC has its own aggression threshold which determines how willing the NPC is to attack the player. Each NPC then associates a threat level with the player, calculated by the aggression threshold, the distance to the player, the player's action towards the NPC (i.e. attack), and so on. The player can choose to attack any NPC with two types of weapons: melee attacking or bullets. Once the player kills an NPC, there is a chance that the NPC drops a weapon power-up, which can upgrade the player's bullet weapon. Four types of power-ups are implemented in current game: fire, bolt, poison and evil. The fire power-up can add special effect to burn the NPCs. The bolt power-up can hurt the NPCs within a certain area. The poison power-up will slow down the NPCs. And the evil power-up will scare the NPCs and decrease their aggression thresholds. All the weapon power-ups in current screen can be picked up by the player as she wishes. However, the

NPCs will grow stronger as the player acquires more powerful weapons according to a certain probability.

The following list shows all the controllable features c_i. The game engine we built can generate a new level with any combination of the controllable features.

- The mean numbers of the three types of NPCs per screen c_1, c_2, c_3. The number of each type of NPCs in a screen takes a Gaussian distribution.
- The variances of the number of NPCs $c_4 ... c_6$.
- The moving speeds of the three types of NPCs $c_7 ... c_9$.
- The aggression thresholds of the three types of NPCs c_{10}, c_{11}.
- The attack frequency of NPCs c_{12}.
- The probability of NPCs dropping weapon power-ups c_{13}.
- The probabilities for the four types of power-ups $c_{14} ... c_{17}$.
- The amount and variance each power-up increases the player's attack c_{18}, c_{19}.
- The health points the NPCs will grow when the player's weapon gets powered up c_{20}.

The player's activities during the gameplay are recorded in the player features p_i. The player features in this paper are chosen inspired by (Pedersen, Togelius, and Yannakakis 2009) and are listed as follows.

- The player's death count p_1.
- The player's attacking count p_2.
- The player's killing count p_3.
- The time spent to finish the level p_4.
- The time spent/killing frequency/attacking frequency of the player in every screen $p_5 ... p_{10}$.
- The average life span of the player p_{11}.
- The percentage of time when the player is running p_{12}.
- The player's weapon preference, melee attacking or ranged attacking p_{13}, p_{14}.
- The player's weapon power-up frequency $p_{15} ... p_{18}$.
- The player's average life span p_{19}.
- The changing of NPCs' health points p_{20}.
- Fail or succeed p_{21}.

Experiments and Results

Data Collection

To examine our model, we perform a player study in which 41 college students have taken part in data collection process for the training step. The players firstly took a simple questionnaire about their player style preference which was used as the prior in the AI model section. The questionnaire contains two simple questions right now, what is the preferred difficulty level (easy or difficult) and what is the preferred weapon type (melee or ranged). Then after taking some time to familiarize themselves with the game interface, each player played three to five random levels. It usually took less than three minutes per level. At the end the players were required to compare and rank all the game levels they had played according to their frustration/boredom. We did not ask for specific frustration/boredom labels from the players due to the inaccuracy of label values according to our assumption.

In the current system, we collect a 41-dimension feature vector for every level which contains 20 controllable features and 21 player features as described in the last section. Each of these features is normalized to within zero and one before the clustering and ranking process.

NDCG (p=3)	k=1	k=2	k=3
Boredom Model	0.875	0.921	0.905
Frustration Model	0.888	0.917	0.908

Table 1: The average NDCG values of boredom and frustration ranking models built with different number of player categories.

Experiment Results

All the feature vectors are clustered into k player categories using the k-means algorithm. For every category, we build one boredom ranking model and one frustration ranking model with corresponding feedback labels and the feature vectors we collected. The Normalized Discounted Cumulative Gain (NDCG)(Jarvelin and Kekalainen 2002), which is a widely used metric for ranking algorithm performance, is utilized to compare the ranking models generated with different size of k. The NDCG value is always between 0 and 1 and a larger value means a better ranking list.

Table 1 shows the average NDCG performance at $p = 3$ for the ranking models. For every category, we randomly select 80% of the feature data and corresponding feedback labels in the category to train the ranking models, and the left 20% are used for testing. This random selection process is repeated ten times and an average NDCG is computed for each of the k categories. Notice that these values are higher than normal since there are at most five levels per query (player) and we are evaluating the ranking lists at $p = 3$. Generally NDCG evaluated at a larger p will be more meaningful. In Table 1, the best case in terms of NDCG performance is achieved when $k = 2$. One reason that the models generated with $k = 3$ do not perform better is that we only have limited amount of training data. With a large k, there are so few players in each category that the ranking algorithm does not obtain a decent training.

In the case of $k = 2$, another interesting phenomenon is that there exists a possible explanation for the two categories generated by the clustering algorithm. We compare the player feature centers of the two categories and notice that the players in category 1 are much more skilled at the game than category 2. The players in category 2 have much higher percent of failing rate (87%) than those in category 1

	Boredom Model	Frustration Model
Accuracy	91.6%	87.5%

Table 2: The testing accuracies of the frustration and boredom models when $k = 2$.

(12%). The death rate of category 2 is also much higher than category 1 (4.3 vs. 1.8 per level). What is more interesting is that in the questionnaire collected from the players before playing the game, 70.7% of the players in category 1 replied that they preferred difficult levels than easy ones, while only 40% of the players in category 2 had the same preference over difficult levels.

Another 12 college students took part in the testing process. Every tester was required to play a randomly generated level after he or she took the questionnaire and became familiar with the game. The testers were then clustered into the k categories using a soft version of k-means algorithm, and classified into one particular category according to Equation 1. Our framework would then present an adapted (the least frustrating or boring) level with the corresponding ranking model. The players were asked to compare their frustration or boredom between the randomly generated level and the adapted one. In order to eliminate the noise introduced by the sequence of levels presented to the players, we asked every participant to play another pair of levels in the sequence of firstly the adapted level and then the random one. Table 2 gives the accuracies of the models where $k = 2$. The accuracy is the percent of level pairs in which the players felt the personalized adapted level was less frustrating or boring than the random level. The results show that our ranking models can generate less frustrating or boring levels with high probabilities. Note that throughout the experiments, we did not ask for any digital ratings from the players, which are inaccurate according to our assumption.

The game flow theory suggests that the player will have the best experience if they feel neither frustrated nor bored. To test this, we combine the two ranking models (frustration and boredom) using a simple summation of the model outputs, and find the levels with the lowest total. Again the testers were required to play the two pairs of game levels in the sequence as above except that the second and third levels were adapted to minimize both frustration and boredom. For each pair of levels, the players were asked to compare which of the two was more fun. In 18 out of the 24 pairs, the testers found the adapted level more fun. Thus the final accuracy for the combined model is 75%.

Discussions

The experiment results on the testbed game show that our model can successfully capture the new player's style and generate a less boring/frustrating level with a high probability. This demonstrates that the ranking algorithm is one possible solution to avoid the inconsistency and inaccuracy problems inherent in player feedback. It also verifies that the players, at least some of them, do share a consistent criterion about comparing which level is more frustrating/boring, which is the basic assumption of our ranking model.

Intuitively, fun is a much more complex to describe and thus more difficult to estimate in computer games. Previous studies also show that the accuracies of directly generating more fun levels are usually much lower than generating more frustrating/boring ones (Pedersen, Togelius, and Yannakakis 2009; Shaker, Yannakakis, and Togelius 2010). Our combined model (minimizing both frustration and boredom) shows a better-than-chance capability of generating more fun content, although a naive combination method is applied currently.

The comparison of ranking models generated with different k implies that the dynamic player style categorization improves the ranking performance to some extent. Both the performance under $k = 2$ and $k = 3$ are better than only one player category. But more data are needed to draw a statistical reasonable conclusion. It is also impossible to make any inference on the effect of prior knowledge acquired from the questionnaire with such a small dataset size. But the prior at least can help alleviate the overfitting problem in the case of small dataset.

In our experiments, we encounter a problem which may be common to all the gameplay based player modeling (Yannakakis and Togelius 2011). The model we develop has no way of explicitly attributing the causes of frustration or boredom during gameplay, and assumes that the cause is included in the set of controllable features. This may not have been the case, as it is indeed possible that an external factor, such as the game controls, became frustrating during a particular level and caused a change in reported frustration. For any non-trivial game it would be infeasible to test every atomic element of the game's design in isolation, so this type of error can only be avoided by ensuring that the non-controlled features of gameplay are as useful as possible. Another problem is the reliability of the player feedback. Even with an initial familiarization with the game, the players will always become more skilled at the game during the training or testing process. Thus the quality of player feedback is inevitably affected. The same level could be less frustrating due to player learning.

Another interesting finding in these experiments is that the players could report an increase both in boredom and frustration from one randomly generated level to the next. By our initial assumption and the game flow theory this was impossible, as we assume a game can be one or the other, but not both simultaneously. These cases occurred when the game became extremely difficult for very mundane reasons, i.e. the level contained a large number of fast-moving green ball enemies which would kill the player on touch. The players became frustrated at the tedious nature of dodging these, and bored since dodging the enemies was what the gameplay had been reduced to. Such a scenario was not foreseen in our assumptions of gameplay.

Conclusions

In this paper, we are motivated by the need for automatically generating game levels customized to player's preferences from player feedback and gameplay features. We present a two step framework to address this: the dynamic player

style categorization and player modeling. Two problems are identified in the player feedbacks: the inconsistency and inaccuracy problem. The ranking based modeling algorithm is proposed to address these problems. In the experiments, we build a testbed game to examine our framework. Experiment results shows that the framework can generate less boring/frustrating levels with very high probabilities. We also combine the two models to minimize both frustration and boredom simultaneously. A decent accuracy is achieved on the current test dataset.

There are many avenues for future work. The performance of the player style categorization is limited to the training size and the gameplay features we use. The selection of gameplay features are of importance to the process. It will also be interesting to study the relationship between the selected features and the categories. The influence of our prior knowledge is still unexplored until now. The design of prior questions itself requires plenty of presumptions. In addition, more priors such as the traditional pre-conceived categories can be added to alleviate the problem of small training dataset. In terms of generating more fun levels, we currently use a naive combination method which directly sums the outputs of the two models. More sophisticated combination techniques are worth trying and testing.

Acknowledgments

We wish to thank all the subjects participating in our experiments. Special thanks to Dr. Mark Riedl for his help and comments.

References

Csikszentmihalyi, M. 1990. Flow: The psychology of optimal experience. *Harper Perennial*.

Doran, J., and Parberry, I. 2010. Controlled procedural terrain generation using software agents. *IEEE Transactions on Computational Intelligence and AI in Games*.

Hastings, E.; Guha, R.; and Stanley, K. O. 2009. Evolving content in the galactic arms race video game. *Proceedings of the IEEE Symposium on Computational Intelligence and Games*.

Hecker, C.; Raabe, B.; Enslow, R.; DeWeese, J.; Maynard, J.; and Prooijen, K. 2008. Real-time motion retargeting to highly varied user-created morphologies. *ACM SIGGRAPH*.

Jarvelin, K., and Kekalainen, J. 2002. Cumulated gain-based evaluation of ir techniques.

Joachims, T. 2002. Optimizing search engines using clickthrough data. *ACM KDD*.

Martinez, H.; Hullett, K.; and Yannakakis, G. N. 2010. Extending neuro-evolutionary preference learning through player modeling. *IEEE Conference on Computational Intelligence and Games (CIG)*.

Murray, T., and Arroyo, I. 2002. Towards measuring and maintaining the zone of proximal development in adaptive instructional systems. *International Conference on Intelligent Tutoring Systems*.

Parish, Y., and Mller, P. 2001. Procedural modeling of cities. *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*.

Pedersen, C.; Togelius, J.; and Yannakakis, G. N. 2009. Modeling player experience in super mario bros. *Computational Interlligence and Games*.

Peinado, F., and Gervas, P. 2004. Transferring game mastering laws to interactive digital storytelling. *International Conference on Technologies for Interactive Digital Storytelling and Entertainment*.

Riedl, M. O.; Stern, A.; Dini, D.; and Alderman, J. 2008. Dynamic experience management in virtual worlds for entertainment, education, and training. *International Transactions on Systems Science and Applications, Special Issue on Agent Based Systems for Human Learning*.

Roden, T., and Parberry, I. 2004. From artistry to automation: A structured methodology for procedural content creation. *Proceedings of the International Conference on Entertainment Computing*.

Schuytema, P. 2007. Game design: A practical approach. *Charles River Media*.

Shaker, N.; Yannakakis, G.; and Togelius, J. 2010. Towards automatic personalized content generation for platform games. *Artificial Intelligence and Interactive Digital Entertainment*.

Spronck, P.; Sprinkhuizen-Kuyper, I.; and Postma, E. 2004. Difficulty scaling of game ai. *International Conference on Intelligent Games and Simulation*.

Sweetser, P., and Wyeth, P. 2005. Gameflow: a model for evaluating player enjoyment in games. *Computers in Entertainment*.

Thue, D.; Bulitko, V.; Spetch, M.; and Wasylishen, E. 2007. Interactive storytelling: A player modelling approach. *Artificial Intelligence and Interactive Digital Entertainment*.

Togelius, J.; Nardi, R. D.; and Lucas, S. M. 2007. Towards automatic personalised content creation for racing games. *Proceedings of the IEEE Symposium on Computational Intelligence and Games*.

Tognetti, S.; Garbarino, M.; Bonarini, A.; and Matteucci, M. 2010. Modeling enjoyment preference from physiological responses in a car racing game. *IEEE Conference on Computational Intelligence and Games*.

Vygotsky, L. 1986. Thought and language. *MIT Press Cambridge*.

Yannakakis, G., and Togelius, J. 2011. Experience-driven procedural content generation. *IEEE Transactions on Affective Computing*.

Yannakakis, G. N.; Martinez, H. P.; and Jhala, A. 2010. Towards affective camera control in games. *User Modeling and User-Adapted Interaction*.