Designer Modeling for Personalized Game Content Creation Tools

Antonios Liapis[1], **Georgios N. Yannakakis**[1,2], **Julian Togelius**[1]
1: Center for Computer Games Research, IT University of Copenhagen, Copenhagen, Denmark
2: Institute of Digital Games, University of Malta, Msida, Malta
Mail: anli@itu.dk, georgios.yannakakis@um.edu.mt, julian@togelius.com

Abstract

With the growing use of automated content creation and computer-aided design tools in game development, there is potential for enhancing the design process through personalized interactions between the software and the game developer. This paper proposes designer modeling for capturing the designer's preferences, goals and processes from their interaction with a computer-aided design tool, and suggests methods and domains within game development where such a model can be applied. We describe how designer modeling could be integrated with current work on automated and mixed-initiative content creation, and envision future directions which focus on personalizing the processes to a designer's particular wishes.

Introduction

A game's aesthetic quality can be determined by several factors, including its art style, the pacing of its interactions, the emotions evoked by in-game or pre-scripted events, or the departures from typical games of its genre. All of these elements can be traced back to creative members of the design and art team: game designers, level designers, visual and sound artists or writers. The design process of these creators differs depending on the type of creative task, the structure of the design team and ultimately on the preferences and vision of each individual. However, a unifying trait in modern game development is the almost ubiquitous use of sophisticated computer-aided design tools. From game editors such as *Unreal Development Kit* (Epic Games 2009) to graphic design tools such as *Photoshop* (Adobe 1990) and from brainstorming tools such as *Mindjet* (Mindjet 2012) to procedural generators such as *SpeedTree* (IDV 2002), computer-aided design tools are used in almost all creative tasks in order to minimize development time and cost, reduce human effort, support collaboration among members of a design team or elicit a user's creativity.

Current tools for game development, however, do not afford personalized interaction. It stands to reason that creative tools used during game development would benefit from being able to identify the designer's intentions, preferences, design patterns and affective reactions and accommodate them in the artifacts they generate. Using a computer-aided design tool that is knowledgeable of its user or users can reduce designer effort and speed up content creation; moreover, it can make suggestions which are counter-intuitive to the designer's thought process in order to spur the designer's creativity via lateral thinking (De Bono 1970). This paper introduces the term *designer modeling* as the mechanism which identifies the style, preference, intentions and affective states of designers and artists in a similar fashion that player modeling attempts to identify behavioral, cognitive and affective patterns of a player (Yannakakis et al. 2013). Designer modeling adopts key principles from *user modeling* (Benyon and Murray 1993) and applies them directly to the interaction between a designer and an authoring tool during the creative process. While user modeling traditionally models the end-user interaction in order to inform a designer, there are a lot of potential synergies between the two fields; for instance the Goals, Operators, Methods and Selection rules model from user modeling (John and Kieras 1993) is very similar to the modeling of designer goals, preference and process suggested in this paper. One interesting difference between these endeavors is that while user modeling is mainly intended to enhance the efficiency of a process, designer modeling is also intended to augment the creativity of designers, and thus qualitatively increase the results of their work.

In this paper we draw the connections and the distinctions between designer modeling and player modeling and suggest different methods of performing designer modeling. The paper also envisions game development tasks where designer modeling can be applied. Finally, we propose how a designer model can also be used to enhance creativity and reduce designer fixation.

What Designer Modeling Is and Is Not

A reliable designer model can be used to enhance the creative processes via a computer-aided design (CAD) tool. Existing CAD tools for games (Smith, Whitehead, and Mateas 2011; Smelik et al. 2011) support co-creation at large, but do not discern the preferences, emotions or goals of their user as a human collaborator would (Lubart 2005). We view the designer model as an important computational element between human and computational creation (see Fig. 1). Information collected during the human design process is fed

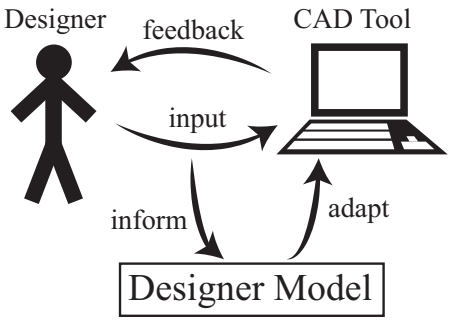

Figure 1: The designer model incorporated in a computer-aided design (CAD) process.

to the designer model, which adapts *online* as the interaction takes place or *offline* based on past interaction data. The model, in turn, provides a set of heuristics that drive the generative process or the search of the computational creator.

Designer modeling is related and analogous to *player modeling*, but there are some important distinctions. Player modeling is the study of computational means for capturing behavioral, cognitive and affective patterns of players (Yannakakis et al. 2013); player models, in turn, are either directly used to influence the game (Hunicke 2005) or inform designers about a player's properties and experiences (Yannakakis and Togelius 2011). Designer modeling is a far more complex and challenging task as it can be viewed as a *second-order player modeling* process; that is, a successful designer model incorporates the intentions of the designer to satisfy the player. In a sense, a designer model evaluates the designer's evaluation of player experience and playstyle.

However, a designer is not solely concerned about the challenge of the game or the pacing of its interactions. Other typical concerns include the presence of a unified style (or theme) within the visual, aural and gameplay elements of the game, or an intended 'message', emotional response or learning outcome of the game, especially in purposeful games such as those intended for education. Creating models of such designer concerns is very different from player modeling, but falls under general machine learning. As a more mature academic domain, machine learning can provide significant insight on how to actualize designer modeling for the purposes of game development; in the next section, connections are drawn between the computational challenges of modeling preferences, goals and process and larger machine learning problems. It can be argued that successfully modeling all relevant aspects of a designer's intent, style and capability would be so complex as to be AI-complete; however, this is an empirical question and even achieving more limited versions of designer modeling could prove very useful.

What Can Be Modeled

As the interaction between a designer, the CAD software and the designed object is multi-faceted, there are many aspects of the process that could be modeled — often at different levels. In the following paragraphs we analyze which aspects could be modeled in a meaningful way.

Modeling Preferences

As any human user, so do designers have their own preferences on the types of content they wish to create. Such preferences may be personal, such as visual taste or favored playstyle, or they can depend on the game's setting, storyline or intended player experience: one game may require a frantic pace, while another may require vibrant visuals. Even within the same game, designer preference could pertain to the design patterns of a specific level, to the appearance of an alien race, or to the challenge level of a specific puzzle. A designer model could identify all of these preferences, regardless of how specific they are; once such preferences are identified, new content which aligns to these preferences can be generated. How such a design model is constructed depends on whether preferences are pertinent to the task at hand or a matter of personal taste; in the former case the model can be trained online, adapting solely to the user's current interactions, while in the latter case the model can be trained offline based on a large body of past interactions. This aspect of designer modeling shares the same computational challenges as *preference learning* (Fürnkranz and Hüllermeier 2010). Beyond machine learning, modeling designer preferences for CAD tools also needs to address challenges pertaining to human-computer interaction, since intrusive tools which use learned preferences in a way that limits a user's creative control can have adverse effects (Liapis, Yannakakis, and Togelius 2012a).

Modeling Style

Many designers have a particular style which can be easily identified across different levels of the same game or even across different games. Telltale signs include visual details such as rounded edges or particular color schemes as well as more abstract features such the presence or absence of open spaces, rhythm and pace, player choice, explicit or implicit reward mechanisms etc. As an example, consider the classic level designs of Shigeru Miyamoto in several early Nintendo games such as *Super Mario Bros* (Nintendo 1985) and *The Legend of Zelda* (Nintendo 1986), where the designer's preference for hidden features and quirky passages that reward replay and exploration can be seen across dissimilar games. Such designer style can be modeled as a form of preference in the sense that, when a design problem occurs, the designer's style is implemented as their preference for one among several possible solutions to that problem; faced with the problem of how a player can progress through a door, Miyamoto might prefer hiding a key behind a crack in the wall whereas another designer might prefer putting a strong enemy in front of the door. Since style is ultimately a form of preference, the same computational challenges of modeling designer preferences apply; however, since style is more pervasive across games or genres, larger datasets may be necessary to discern styles, moving the modeling of a designer's style closer to the larger problems of data mining, clustering and pattern recognition.

Modeling Goals and Intentions

A CAD tool is expected to help the designer achieve their goals. While *goal recognition* via machine learning has been

identified early on (Goodman and Litman 1992) as a significant enhancement to user/system interactions, there has been limited interest in incorporating it in tools for game development. A prediction of a designer's current goals can be made from past interactions; assuming that the designer has a consistent style, the system can predict the designer's next steps by matching the user's current creative step with previously seen interactions. The end result of such previous interactions can be shown, as advice, in order to speed up the creative process. Alternatively, the system can suggest likely next steps based on a probabilistic model of cognitive associations (Wang 2008), which is trained on past design sessions. There are however several challenges in the prediction of goals: for instance, the system is likely to present no new content, as it uses past artifacts more or less directly. This is as likely to stifle creativity and enhance designer fixation as it is to speed up the creative process. Additionally, the goals of the designer are rarely known in advance, even to themselves; it is therefore difficult to evaluate if the tool succeeds in predicting or helping achieve the designer's goals.

Modeling Process

Different designers follow different design processes: some level designers start with a rough sketch of the map layout, others start from identifying the intended challenge level while yet others have fleshed out specific portions of the level (i.e. a chokepoint, a sniper location or a secret passage) before worrying about the big picture. A challenging goal of designer modeling, therefore, would be to identify such different processes and integrate them into their support mechanisms. Following the level designer example, a design tool could alternate between generating suggestions which increase the 'resolution' of a rough sketch (Liapis, Yannakakis, and Togelius 2013d), suggestions which maintain the intended challenge level, and suggestions which add details only to the portions of the level which are not fully fleshed out yet. The design process of the human and the computational designer can be even more closely matched, such as for instance identifying if the level designer is currently placing resources or obstacles. Modeling a designer's process can be viewed as a short-term *plan recognition* problem (Kautz 1987), and thus actions such as resource placement can be clustered together for the purposes of activity recognition (van Kasteren, Englebienne, and Kröse 2011).

Modeling Affective States

As emotion plays a critical role in human decision making (Damásio 1994), it is central to the drive of creativity. Arguably, a designer's emotion is far more complex to identify and measure when compared to modeling behavioral patterns of interaction. However, research in affective computing (Picard 1997) has shown the potential of reliably modeling user affect via emotional manifestations. We argue that similar to modeling player's affect during gameplay interaction through head motion (Asteriadis et al. 2012) or physiology (Martínez, Garbarino, and Yannakakis 2011), affective computing methods can be applied to detect designer frustration or surprise which are critical during creative processes. Conceivably, using a designer model that predicts which content will surprise a designer, a computational creator could generate and present surprising suggestions in order to break designer fixation.

Individual versus Collective Designer Modeling

As in player modeling, there is the potential to model both individual designers and groups of designers such as the members of a game studio. A designer modeling tool that captures the preferences, goals and processes of a large community of designers would be useful, as it could learn much more from the abundance of interaction data than it could learn from a single designer. Having a good initial model would also be beneficial to novice designers, who might need more help in the design process. Modeling individual designers on the other hand allows for personalization, especially considering that designers often have highly divergent styles and ways of working, as noted above. Conceivably, an effective designer model would include both collective and individual models, with the former describing a baseline and the latter deviations from that baseline. Classical supervised learning algorithms would likely be useful for collective models based on large amounts of data whereas instance- or case-based models could be used for individual designer modeling.

Modeling within Game Development

In a typical game company, developers with creative tasks include game designers, level designers, content creators (visual and sound artists) and in many cases story writers. In this section we propose methods for accommodating each of these designer types via creativity-enhancing tools which adapt to fit their users.

It should be noted that the proposed methods of designer modeling do not solely depend on the type of creator, but also on the size and hierarchical structure of the development team. For instance, an indie developer working alone to design, program and create assets can benefit from a highly personalized model while a large team of artists may require a collective model of visual preferences. Even so, such large teams of artists often have an art director who strives for a unified art style; a designer model can therefore be personalized to the director and disseminated to the artists in order to notify them of deviations from the art director's vision. In looser hierarchical structures, a group of game designers may use a collective designer model in a collaborative environment, interweaving the preferences of other designers with their own and drawing inspiration from the design processes modeled after other members of the team.

Game designers

The core contribution of a game designer is shaping the overall gameplay experience by defining the game's rules and mechanics. Game designers are often concerned about game balance, as well as maintaining a challenging and engaging experience throughout the entire game.

Togelius and Schmidhuber (2008) created movement and collision rules for abstract video games, evaluating them on how easily artificial controllers could learn to play them,

while Cook et al. (2013) altered and combined video game rulesets targeting a longer sequence of actions for solving a level (using a breadth-first search controller). As such methods use an artificial controller to play the game, designer modeling can be realized by adapting the artificial controller to match the designer's intended playstyle. Assuming a simulation-based evaluation of game quality (Togelius et al. 2011), training the artificial controller to match designer-sanctioned gameplay patterns will indirectly affect the evaluation score of such games, and thus types of games are favored. As an example, Togelius, De Nardi, and Lucas (2007) trained artificial neural networks to drive like individual human players, which were used to evaluate new racing tracks personalized to those human players; this could be considered designer modeling if the player is considered the track designer.

Browne and Maire (2010) evolved board game layouts and rules and evaluated them on 57 criteria aggregated into a weighted sum; the impact of each criterion to 'human interest' was determined through a user survey with 57 participants. That study tried to accommodate all 57 participants, and thus could be considered collective designer modeling; the same method could also be applied to personalize the impact of such criteria to a single game designer. The user study of Browne and Maire required each participant to play a pair of games and identify the most interesting one; choice-based interactive evolution (Liapis et al. 2013) could similarly be used to adapt the impact of these criteria based on the designer's preference, and evolve a new pair of games allowing the designer to continuously adapt and refine both the game rules and the model of quality which evaluates them.

Level designers

Level designers create the gameworld where the game takes place; in most cases such a gameworld is separated into levels played in succession, although open-ended games usually consist of a single massive and highly-detailed world. Procedurally generated levels have seen extensive use in the game industry, but their quality is rarely evaluated. Researchers have proposed evaluation functions based on e.g. on the rhythm of platformer levels (Smith, Whitehead, and Mateas 2011), the fairness of strategy game levels (Liapis, Yannakakis, and Togelius 2013c), the speed variance of racing tracks (Togelius, De Nardi, and Lucas 2007) or the allowed solutions of puzzle games (Smith, Butler, and Popović 2013); the impact of such engineered measures can be adjusted to designer preferences.

As an example of how designer modeling could be implemented, we envision a future version of the Sentient Sketchbook level design tool which integrates a designer model. In Sentient Sketchbook (Liapis, Yannakakis, and Togelius 2013b), the tool constantly measures the qualities of the current state of the level design through several metrics related to exploration, area control and balance (Liapis, Yannakakis, and Togelius 2013e), and provides real-time feedback to the designer as well as suggestions for changes that the designer can choose to apply or ignore. One way of modeling designer preferences could be implemented by learning preferred weightings on the various quality metrics based on the comparison of accepted suggestions with suggestions that were ignored during the design process. This could be represented as a linear or non-linear function, such as a weighted sum or a multilayer perceptron respectively. A baseline model of level quality could be learned from the entire community of users of this tool, which could be used to help discern and quickly adapt to individual deviations. Individual or collective designer process could be modeled by recording sequences of actions and storing them in a case base. These could then be accessed by a nearest-neighbor retrieval procedure so that when a designer starts a new map sketch, the first steps are matched to the corpus of existing design sequences.

Visual artists

Visual artists include several different professions such as concept artists, 3D modelers, animators and art directors. Such artists mostly use computer-aided design tools such as *Photoshop* (Adobe 1990) or *3D Studio Max* (Autodesk 1996); however, automated generators such as *FaceGen* (Singular Inversions 2001) and *SpeedTree* (IDV 2002) are also used as shortcuts for visual asset creation within the game industry. Evolutionary algorithms have been used to create both 2D art and 3D art, where a visual artist can explicitly select which visual patterns they wish to favor (Wannarumon 2010) or by selecting which artifacts should be allowed to evolve (Secretan et al. 2011). While such 'direct' designer modeling can result in highly customizable artifacts, a less intrusive method for modeling the artists' preferences is also possible. By adjusting the prevalence of specific visual patterns according to the appearance of the artist's preferred artifacts, the generator can create new artifacts which accommodate the artist's style. This adaptive model of visual taste has been demonstrated in the domain of 2D spaceship design (Liapis, Yannakakis, and Togelius 2011; 2012b).

Sound artists

While often overlooked, sound is an important element of games and can greatly affect the player's experience; sound artists create the background music and sound effects of the game. There has been limited research interest in the creation of music and sound effects for games, although musical metacreation has seen several important developments in the last years, including the creation of a workshop[1]. Similar to the visual arts, artificially evolved music is driven directly by human selections (Hoover, Szerlip, and Stanley 2011) or by objective functions (Eigenfeldt and Pasquier 2013); a model which adapts these objective functions (Liapis et al. 2013) could be applied to personalize the generated music to the designer's wishes. A more direct form of designer modeling could be used to learn from a corpus of designer-authored (or designer-favored) sound pieces; new sound pieces could be created from such a corpus via probabilistic models such as Hidden Markov Models (Pardo and Birmingham 2005).

[1] 1st International Workshop on Musical Metacreation (2012) at http://www.metacreation.net/mume2012

Writers

The term 'writer' in this paper is used loosely to include the story writers which flesh out the main story arc as well as writers of quests and NPC dialogue. Not all games require writers if they favor short interactions (e.g. casual or sandbox games); however, as many AAA games rely on an engrossing plot, narrative generation has potential applications in game development as demonstrated by the game-like interactive drama prototype Façade (Mateas and Stern 2005). Events or complete stories for games can be authored via mixed-initiative tools for story generation (Skorupski et al. 2007; Thomas and Young 2006); designer modeling can be accommodated directly by allowing human authors to create a corpus of "agent types, mediation strategies and narrative macros" (Thomas and Young 2006). Machine learning applied to this corpus of past stories or elements of stories can be used to identify popular tropes and predict the writer's goals. Another possibility is to use computational models of coherence, novelty and interestingness (Pérez y Pérez and Ortiz 2013) or plot point flow (Weyhrauch 1997) to evolve story elements as implemented by Giannatos et al. (2011); these models can be adapted via designer modeling based on the preferences or general style of the writer.

Breaking the model

While the primary purpose of designer modeling — as presented in this paper — is to accommodate the computer-generated artifacts to the designer's preferences, goals and processes, a personalized model can also be used to 'break' the designer's common practices. Designer fixation on specific gameplay properties in level design, specific visual patterns in 2D art or specific tropes in writing can lead to banality in the artifacts created; a CAD software which accommodates such fixations would do little to inspire creativity. However, as the designer model identifies those exact fixations, it can purposefully create artifacts that break them. Showing unexpected suggestions which the designer was unlikely to come to by following their current preferences, goals and process is likely to inspire the designer by showing them how to 'think outside the box' (Kuhn 1962). Divergent search algorithms, such as novelty search (Lehman and Stanley 2011), can be used to break designer fixations by adapting the characterization of divergence to the designer's preferences.

It should be noted that breaking the model in a way that designers would not dismiss as 'random' or 'wrong' is a challenging problem. Following the requirements that creative processes should result in artifacts that are both novel and useful (Boden 1990), this problem can be somewhat mitigated by creating surprising suggestions which however fulfill some minimal criteria of usefulness, such as levels which are playable. Constrained novelty search algorithms (Liapis, Yannakakis, and Togelius 2013a) could be used to ensure novel content which remains useful.

Conclusion

Designer modeling has the potential of greatly enhancing the creative process of game developers, by accommodating a designer's intentions for game difficulty, by adapting a visual asset generator to an artist's preferred style or by creating surprising plot developments for a writer. The task of designer modeling is challenging due to the fact that it needs to accommodate a designer seeking to accommodate, in turn, a player, but also due to the fact that a designer's task has unclear and immeasurable goals. While only the very first steps towards actual implementation of these concepts have been taken, we expect that the increasing interest of game companies (and game AI researchers) in intelligent tools for game development will lead to the exploration of several of the concepts presented in this paper.

Acknowledgments

This research was supported, in part, by the FP7 ICT project SIREN (project no: 258453) and by the FP7 ICT project C2Learn (project no: 318480).

References

Asteriadis, S.; Karpouzis, K.; Shaker, N.; and Yannakakis, G. N. 2012. Towards detecting clusters of players using visual and gameplay behavioral cues. In *Proceedings of the International Conference on Games and Virtual Worlds for Serious Applications*, 140–147.

Benyon, D., and Murray, D. 1993. Applying user modeling to human computer interaction design. *Artificial Intelligence Review* 7(3-4):199–225.

Boden, M. 1990. *The Creative Mind*. Abacus.

Browne, C., and Maire, F. 2010. Evolutionary game design. *IEEE Transactions on Computational Intelligence and AI in Games* 2(1):1–16.

Cook, M.; Colton, S.; Raad, A.; and Gow, J. 2013. Mechanic miner: reflection-driven game mechanic discovery and level design. In *Proceedings of Applications of Evolutionary Computation*, volume 7835, LNCS, 284–293. Springer.

Damásio, A. 1994. *Descartes' Error: Emotion, Reason, and the Human Brain*. Putnam Publishing.

De Bono, E. 1970. *Lateral thinking: creativity step by step*. Harper & Row.

Eigenfeldt, A., and Pasquier, P. 2013. Evolving structures in electronic dance music. In *Proceedings of Genetic and Evolutionary Computation Conference*.

Fürnkranz, J., and Hüllermeier, E. 2010. *Preference Learning*. Springer-Verlag New York, Inc.

Giannatos, S.; Nelson, M. J.; Cheong, Y.-G.; and Yannakakis, G. N. 2011. Suggesting new plot elements for an interactive story. In *Proceedings of the AIIDE workshop on Interactive Narrative Technologies*.

Goodman, B. A., and Litman, D. J. 1992. On the interaction between plan recognition and intelligent interfaces. *User Modeling and User-Adapted Interaction* 2:83–115.

Hoover, A. K.; Szerlip, P. A.; and Stanley, K. O. 2011. Interactively evolving harmonies through functional scaffolding. In *Proceedings of Genetic and Evolutionary Computation Conference*.

Hunicke, R. 2005. The case for dynamic difficulty adjustment in games. In *Advances in Computer Entertainment Technology*, 429–433.

John, B. E., and Kieras, D. E. 1993. The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction* 3(4):320–351.

Kautz, H. A. 1987. *A formal theory of plan recognition*. Ph.D. Dissertation, Bell Laboratories.

Kuhn, T. S. 1962. *The Structure of Scientific Revolutions*. University of Chicago Press.

Lehman, J., and Stanley, K. O. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation* 19(2):189–223.

Liapis, A.; Martínez, H. P.; Togelius, J.; and Yannakakis, G. N. 2013. Adaptive game level creation through rank-based interactive evolution. In *Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG)*.

Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2011. Optimizing visual properties of game content through neuroevolution. In *Proceedings of the AAAI Artificial Intelligence for Interactive Digital Entertainment Conference*.

Liapis, A.; Yannakakis, G.; and Togelius, J. 2012a. Limitations of choice-based interactive evolution for game level design. In *Proceedings of the AIIDE Workshop on Human Computation in Digital Entertainment*.

Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2012b. Adapting models of visual aesthetics for personalized content creation. *IEEE Transactions on Computational Intelligence and AI in Games* 4(3):213–228.

Liapis, A.; Yannakakis, G.; and Togelius, J. 2013a. Enhancements to constrained novelty search: Two-population novelty search for generating game content. In *Proceedings of the Genetic and Evolutionary Computation Conference*.

Liapis, A.; Yannakakis, G.; and Togelius, J. 2013b. Sentient sketchbook: Computer-aided game level authoring. In *Proceedings of the ACM Conference on Foundations of Digital Games*.

Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2013c. Generating map sketches for strategy games. In *Proceedings of Applications of Evolutionary Computation*, volume 7835, LNCS, 264–273. Springer.

Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2013d. Sentient world: Human-based procedural cartography. In *Proceedings of Evolutionary and Biologically Inspired Music, Sound, Art and Design (EvoMusArt)*, volume 7834, LNCS, 180–191. Springer.

Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2013e. Towards a generic method of evaluating game levels. In *Proceedings of the AAAI Artificial Intelligence for Interactive Digital Entertainment Conference*.

Lubart, T. 2005. How can computers be partners in the creative process: classification and commentary on the special issue. *International Journal of Human-Computer Studies* 63(4-5):365–369.

Martínez, H. P.; Garbarino, M.; and Yannakakis, G. N. 2011. Generic physiological features as predictors of player experience. In *Proceedings of the 4th International Conference on Affective Computing and Intelligent Interaction*, 267–276.

Mateas, M., and Stern, A. 2005. Procedural authorship: A case-study of the interactive drama façade. In *Digital Arts and Culture*.

Pardo, B., and Birmingham, W. P. 2005. Modeling form for on-line following of musical performances. In *Proceedings of the AAAI*. AAAI Press.

Pérez y Pérez, R., and Ortiz, O. 2013. A model for evaluating interestingness in a computer-generated plot. In *International Conference on Computational Creativity*.

Picard, R. W. 1997. *Affective computing*. MIT Press.

Secretan, J.; Beato, N.; D'Ambrosio, D. B.; Rodriguez, A.; Campbell, A.; Folsom-Kovarik, J. T.; and Stanley, K. O. 2011. Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary Computation* 19(3):373–403.

Skorupski, J.; Jayapalan, L.; Marquez, S.; and Mateas, M. 2007. Wide ruled: A friendly interface to author-goal based story generation. In *International Conference on Virtual Storytelling*, 26–37.

Smelik, R. M.; Tutenel, T.; de Kraker, K. J.; and Bidarra, R. 2011. A declarative approach to procedural modeling of virtual worlds. *Computers & Graphics* 35(2):352–363.

Smith, A. M.; Butler, E.; and Popović, Z. 2013. Quantifying over play: Constraining undesirable solutions in puzzle design. In *Proceedings of ACM Conference on Foundations of Digital Games*.

Smith, G.; Whitehead, J.; and Mateas, M. 2011. Tanagra: Reactive planning and constraint solving for mixed-initiative level design. *IEEE Transactions on Computational Intelligence and AI in Games* (99).

Thomas, J. M., and Young, R. M. 2006. Author in the loop: Using mixed-initiative planning to improve interactive narrative. In *ICAPS Workshop on AI Planning for Computer Games and Synthetic Characters*.

Togelius, J., and Schmidhuber, J. 2008. An experiment in automatic game design. In *IEEE Symposium on Computational Intelligence and Games*.

Togelius, J.; Yannakakis, G.; Stanley, K.; and Browne, C. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games* (99).

Togelius, J.; De Nardi, R.; and Lucas, S. 2007. Towards automatic personalised content creation for racing games. In *Proceedings of IEEE Symposium on Computational Intelligence and Games*, 252–259. IEEE.

van Kasteren, T. L. M.; Englebienne, G.; and Kröse, B. J. A. 2011. Hierarchical activity recognition using automatically clustered actions. In *Proceedings of the Second international conference on Ambient Intelligence*, 82–91. Springer-Verlag.

Wang, H.-C. 2008. Modeling idea generation sequences using Hidden Markov Models. In *Proceedings of the 30th Annual Meeting of the Cognitive Science Society*.

Wannarumon, S. 2010. An aesthetics driven approach to jewelry design. *Computer-Aided Design and Applications* 7(4):489–503.

Weyhrauch, P. 1997. *Guiding Interactive Drama*. Ph.D. Dissertation, School of Computer Science, Carnegie Mellon University.

Yannakakis, G. N., and Togelius, J. 2011. Experience-driven procedural content generation. *IEEE Transactions on Affective Computing* 99.

Yannakakis, G. N.; Spronck, P.; Loiacono, D.; and Andre, E. 2013. Player modeling. *Dagstuhl Seminar on Game Artificial and Computational Intelligence*.