Monte-Carlo Tree Search for Persona Based Player Modeling

Christoffer Holmgård[1], **Antonios Liapis**[2], **Julian Togelius**[1,3], **Georgios N. Yannakakis**[1,2]
1: Center for Computer Games Research, IT University of Copenhagen, Copenhagen, Denmark
2: Institute of Digital Games, University of Malta, Msida, Malta
3: Department of Computer Science and Engineering, New York University, New York, USA

Abstract

Is it possible to conduct player modeling without any players? In this paper we use Monte-Carlo Tree Search-controlled procedural personas to simulate a range of decision making styles in the puzzle game *MiniDungeons 2*. The purpose is to provide a method for synthetic play testing of game levels with synthetic players based on designer intuition and experience. Five personas are constructed, representing five different decision making styles archetypal for the game. The personas vary solely in the weights of decision-making utilities that describe their valuation of a set affordances in MiniDungeons 2. By configuring these weights using designer expert knowledge, and passing the configurations directly to the MCTS algorithm, we make the personas exhibit a number of distinct decision making and play styles.

Introduction

This paper investigates new methods for automatic play-testing in games. Play-testing is an integral step of iterative game development. It allows game designers to test their assumptions about player behavior and to observe dynamics of the game system (Fullerton, Swain, and Hoffman 2004). In a sense, it is a partial mapping of the game space itself through observing where human players are capable of and interested in going within that space. As crucial as human play-testing is, it is also time consuming and potentially expensive. Further, it does not necessarily support a quick iterative loop when game designers are creating or fine-tuning new content for a game (e.g. game levels). Level designers can only test their levels with human players so many times, and for many developers minor changes cannot realistically mandate human play-testing. Instead, the designer informally infers or formally analyses the expected atomic and holistic impacts on each minor change in a level design, imagining what players might do, observing her own behavior in the level, or testing internally with her team. While this works for current game development practices we propose that there is a potential for supporting the level design process with generative player models. Generative player models acting as agents which play the game in lieu of players may provide game designers with surrogate play-traces to inform their design decisions or to integrate in their content creation systems. When a designer defines such an agent for a particular game we call them *procedural personas*. As player types akin to those described by Bartle (1996) and the play personas described by Tychsen and Canossa (2008) and Canossa and Drachen (2009), they describe archetypal ways of interacting with the game. They are formal representations of the game designer's assumptions about her players.

Each persona may be used interactively or automatically in the level design process. Interactively, a level designer can inspect different interaction patterns (e.g. play-traces or completion statistics) in the level and iteratively adapt either the level or the persona behavior (Yannakakis, Liapis, and Alexopoulos 2014). Automatically, a procedural content generation system can use the personas as critics that evaluate and change the generated content (Liapis et al. 2015) as part of a search based procedural content generation loop (Togelius et al. 2011). The purpose of procedural personas is thus to create easily configurable artificial game playing agents that believably simulate a variety of human decision making styles.

Inspired by decision theory the behaviors of procedural personas are controlled with simple utility functions (Mark 2009) that designers can easily interpret, change and use to describe archetypal decision making styles. By assigning utility weights to game affordances, the designer prescribes what the different personas should prioritize. Methods for modeling player behavior from observed play traces and methods for creating AI agents that function as believable opponents are research areas that have seen much attention and progress in the last decade (Hingston 2012). However, methods that encode and model designer's notions and expectations of player behavior from simple parameters, and realize these as observable behavior in-game, is still an under-explored part of Game AI (Smith et al. 2011; Yannakakis and Togelius 2014).

In this paper we contribute to the development of AI agents as generative models of expected player behavior by demonstrating the use of procedural personas, configured by utility weights and controlled by Monte-Carlo Tree Search (MCTS), in our test-bed game *MiniDungeons 2*. We start out by grounding our work in decision theory, followed by a description of the MiniDungeons 2 test-bed game. Afterwards, we describe our specific implementation of an MCTS con-

Copyright © 2015, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

troller for MiniDungeons 2. This controller is then used with 5 different procedural persona configurations. The personas are used to play a number of MiniDungeons 2 levels, and their behaviors are recorded and contrasted to test whether meaningful differences arise from the different persona configurations.

Related Work

Two fundamental assumptions about games, drawn from decision theory, underlie the method suggested here.

The first assumption is that players' decisions in games can be understood as being driven by utility. Whenever a player is making a move in a game she is trying to maximize her expected utility. The utility function is personal in the sense that different players will have different preferences when playing games. Their preferences need not be perfectly aligned with the rules of the game and may even change over the course of the game — although this problem is not treated here, as we try to express static utility functions. The utility function is expressed through play style or, more specifically, decision making style.

The second assumption is that players try to optimize their utility in a boundedly rational fashion, using analytic thinking to solve parts of the decision making task while using heuristics to solve other parts of the decision making task (Gigerenzer and Selten 2002; Kahneman 2011). The extent to which the player applies each kind of thinking depends on the player's ability to do so — the player's analytic skill and experience. It also depends on the player's interest in devoting cognitive resources to solving the decision task in the first place, and the extent to which the player possesses specialized heuristics (from expertise) that can replace analytic effort (Gigerenzer and Gaissmaier 2011). MCTS may be a well-suited algorithm for approximating this decision making process in a simulation, since it supports utility weights directly, is composed of an analytical and a heuristic part, and its performance can be varied by changing its computational budget (Lorentz 2011; Bourki et al. 2010; Zook, Harrison, and Riedl 2015). In this paper we explore the feasibility of using MCTS for generating different decision making styles in our test-bed game MiniDungeons 2. Before describing MiniDungeons 2, we briefly describe its predecessor that guided its design: MiniDungeons 1.

MiniDungeons 1 Game Design

Earlier attempts at modeling human decision making were made using the game *MiniDungeons 1* (MD1) (Holmgård et al. 2014b). As a predecessor to the MiniDungeons 2 game, described in this paper, MD1 had a much simpler rule set and a smaller set of affordances which affected human decision making. MD1 levels were laid out on a grid of 12 by 12 tiles, and tiles could be impassable walls, or passable tiles which could be empty or contain potions, treasures, or monsters. Unlike MiniDungeons 2, its predecessor only had one type of monster, which did not move. Combat in MD1 was stochastic: if a hero moved into the position of a monster, the monster would deal a random number of hit point (HP) damage and then die (with the hero moving to its tile). MD1 therefore revolved around the calculated risk of combat: could the hero survive another monster fight and what reward did the monster guard? Personas in MD 1 attempted to model how players pursued various affordances while deciding under uncertainty.

In contrast, MiniDungeons 2 acts as a test-bed for modeling combinations of analytical and heuristic decision making, moving closer to the puzzle genre. For this reason, monsters in MiniDungeons 2 move in a completely deterministic manner and their combat damage is easily predictable. Therefore, MiniDungeons 2 (with larger levels and more complicated mechanics) challenges the decision making skill of a player not in her decision making under uncertainty, but on the long-term risk of the level play-through, which is harder to analyze than in the smaller, simpler MD1 levels.

MiniDungeons 2 Game Design

MiniDungeons 2 is a deterministic one-and-a-half player game with full game state information available to the player (Elias, Garfield, and Gutschera 2012). The game is designed specifically to have a high decision density, meaning that every action matters, while requiring little to no manual skill. It is a turn-based puzzle game where a *hero* travels from the *entrance* of a dungeon level to the *exit* (which loads the next level), similarly to many games in the rogue-like genre. Within the level, there are *potions*, *treasures* and *monsters* of different types. The monsters move in response to the hero's action. Fighting happens when characters collide or when the hero or certain monsters conduct ranged attacks by throwing a *spell* attack or, in the case of the hero, a *javelin*. The hero may kill most monsters by fighting, but crucially, a special monster called the *minitaur* cannot be killed, but can only be knocked out for 3 turns through fighting. Every turn the minitaur will move directly toward the hero along the shortest path as determined by A* path-finding, if it is not knocked out. This helps drive the game toward a terminal state, though it does not guarantee it. A state in MiniDungeons 2 will typically have 3-4 actions available, depending on the level in question. A typical level in MiniDungeons 2 takes 15-30 turns to play, depending on which goals a player chooses to pursue. Depending on monster setup some levels allow the player to play indefinitely, if she so chooses, by running away from the minitaur forever. Figure **??** shows an example MiniDungeons 2 level; the detailed rules of MD2 are described in Holmgård et al. (2015).

Persona Design for MiniDungeons 2

As described above, the personas for MiniDungeons 2 are defined by their individual utility functions and mediated by their computational budget.

The sources of utility in MiniDungeons 2 are defined by the authors, acting as the designers of the game. They are comprised of what we consider to be the most important seven events that can occur in the game: spending a turn, moving toward the exit, killing a monster, collecting a treasure, drinking a potion, dying (and losing the level), and exiting (and completing the level). Table 1 shows the configu-

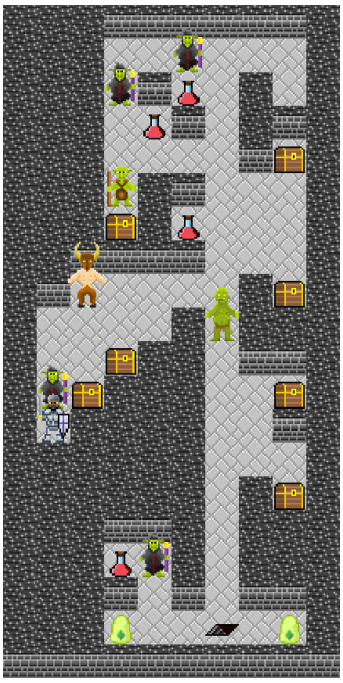

Figure 1: Map 7 from MiniDungeons 2.

	Utility weights of affordances						
Persona	Tu	Di	M	Tr	P	D	E
Exit	-0.01	1					0.5
Runner	-0.02	1					0.5
Survivalist	-0.01	0.5			0.5	-1	0.5
Monster K.	-0.01	0.5	0.5				0.5
Treasure C.	-0.01	0.5		0.5			0.5

Table 1: All personas tested in the experiments. The five different personas are awarded utility from seven affordances in MiniDungeons 2: Taking a turn (Tu), reducing the distance to the exit (Di), killing a monster (M), collecting a treasure (Tr), drinking a potion (P), dying (D) or reaching the exit of a level (E).

rations of the five personas that were defined for MiniDungeons 2. The personas represent different imagined play styles: *Exit* (E) simply tries to finish the level. So do the other personas, but they have auxiliary goals shaping their decisions: *Runner* (R) tries to complete the level in as few turns as possible, *Survivalist* (S) tries to avoid damage and collect potions, *Monster Killer* (MK) tries to kill as many monsters as possible, and *Treasure Collector* (TC) tries to collect as many treasures as possible.

Monte Carlo Tree Search for Persona Control

Monte Carlo Tree Search (MCTS) is a stochastic tree search algorithm that has seen considerable success in some board games and digital games (Browne et al. 2012; Jacobsen, Greve, and Togelius 2014; Champandard ; Perez et al. 2015). It works by expanding each state depending on how promising it is (based on its reward) and on an estimate of how under-explored it is. Reward is calculated by repeatedly playing the game from the given state until a terminal state, and averaging the terminal state reward (win or loss). When using the algorithm for games where random play-outs are not guaranteed (or even likely) to lead to a terminal state within a reasonable time (e.g. many arcade games), the algorithm needs to be modified (Jacobsen, Greve, and Togelius 2014). A common modification is to only perform the play-out for a set number of actions, and then evaluate the end state using some heuristic. We use the standard UCB1 formulation of MCTS (Browne et al. 2012), but only perform play-outs for a maximum length of 10,000 actions and then use the utility function of the persona to evaluate the state. In practice, however, very few play-outs reach the 10,000 action limit and reach a terminal state long before. The utility function is applied to terminal states as well, and terminal states are specifically flagged as such by the game logic. The intent of our work is not only to develop agents that play the game well, but to develop agents that can model player preferences and skill as personas. Skill can be represented through the computational budget allocated to MCTS play-outs. We investigate this by varying the computation time available to the personas. It is important to note that the different utility functions require different information and therefore have marginally different complexities. However, all necessary information for all utility functions is always calculated by the game logic during game-play and play-outs. Therefore, we assume that the personas have the same amount of computation time available and are playing at similar "skill" levels. In spite of this we do not assume that the different persona preferences are equally easy or hard to enact. Being a Monster Killer may be a more difficult decision making style than being a Runner. While this is an interesting question in itself, this paper focuses on behavioral differences between personas with identical computational budgets and performance differences within personas across different computational budgets. Below, we briefly outline our strategy for comparing the behaviors of the various personas and their performances under varying computational budgets.

Metrics

In this section we describe the three different types of metrics we use to evaluate the implemented personas as procedural representations of imagined decision making styles: Action agreement ratios, summary statistics, and heat-maps.

Action Agreement Ratio

In order to establish that the personas are in fact enacting different decision making styles, we apply an agreement metric across individual actions. The metric we use to evaluate persona likeness, developed in previous work (Holmgård et al. 2014a) on persona/human comparison, is the *action agreement ratio* (AAR). AAR considers each step of a play-trace a distinct decision, in line with the high decision density of MiniDungeons 2. To produce the AAR between two personas, all distinct game states of a persona play-trace, the original, are reconstructed. For each game state, the other

persona being compared for agreement is inserted into the game state and queried for the next preferred action, essentially asking: "What would you do?". If the two personas choose the same action, one point is registered. Finally, the AAR is computed by dividing the number of points with the number of decisions in the original play-trace. A perfect AAR score of 1.0 represents two personas that agree on every single decision.

In *Results* we describe the experiments we ran to generate persona behaviors and examine their performances and differences through summary statistics and through AAR.

Summary Statistics and Heat-Maps for Persona Identification and Skill Evaluation

Since the different personas are constructed to pursue different affordances in MiniDungeons 2 it is difficult to evaluate all of them using a simple unidimensional score system. Instead, we summarize the number of affordances reached during game-play and interpret these as indications of whether a persona is enacting a desired style. While this arguably is a subjective approach to determining whether personas enact a desired decision making style, the fundamental persona concept is subjective to the designer. Future work will focus on determining if personas are recognizable across observers, but here we directly interpret the summary statistics from a game design perspective. Further, we examine a number of heat-maps representing persona play-traces to identify patterns that characterize the behavior of the various personas.

For evaluating whether varying the computational budget has an impact on persona performance we apply a straightforward operationalist approach: For each persona we determine whether variation in the computational budget impacts the interaction with affordances that factor into the particular persona's utility function. If a Monster Killer manages to kill more monsters or if a Treasure Collector manages to collect more treasure, we interpret this as an indication of greater skill. In the following section we first present the experiments we ran to obtain results and then analyze each of the metrics.

Results

The five MCTS personas (Exit, Runner, Survivalist, Monster Killer and Treasure Collector) were tested on 10 handcrafted maps in MiniDungeons 2. The levels were crafted to represent varying degrees of difficulty and to allow for the expression of all five decision making styles. The levels were informally tested to ensure that this was the case. Five test conditions were defined with $10ms$, $100ms$, $1s$, $10s$, and $60s$ of computation time, respectively. Each controller was run 20 times on each map in order to take into account the effect of stochasticity, and the means of the scores are reported here. All experiments were run on an Intel Xeon E5-2680 Ivy Bridge CPU at 2.80GHz.

Action Agreement Ratios

AAR values were obtained by cross-comparing all personas against each other with $1s$ of computation time. For each persona, the other four personas as well as a new instance of the same persona were presented with all $1s$ play-traces from the original persona (20 per map) and asked to evaluate each decision, yielding an AAR value. Each play-trace was evaluated 20 times to account for stochasticity. The average AAR values resulting from this process are presented in Table 2.

	E	R	S	MK	TC
Exit	**0.65**	0.49	0.42	0.40	0.43
Runner	0.43	**0.65**	0.59	0.46	0.56
Survivalist	0.42	0.56	**0.61**	0.45	0.51
Monster Killer	0.47	0.49	0.48	**0.64**	0.46
Treasure Collector	0.44	0.58	0.52	0.40	**0.68**

Table 2: Action Agreement Ratios (AAR) between all personas. The AARs range from 0 to 1 and indicate the extent to which two personas agree on atomic in-game decisions. The results indicate that personas agree with themselves to a larger extent than with other personas. However, it is also evident that some variation happens within personas, likely due to the stochasticity of the MCTS algorithm. All AARs were calculated with $1s$ of decision making time per action.

The results indicate a number of interesting things. No personas achieve perfect agreement, not even with themselves. This may be due to the stochasticity of the MCTS algorithm. The state space of MiniDungeons 2 is too large to be fully explored by MCTS. Hence, each instance of a persona may end up exploring slightly different parts of the state space and reach different conclusions about the best course of action. In particular, sometimes an agent walks back and forth while still implementing an overall optimal path. Secondly, all personas agree more with themselves than with any other persona, with a mean AAR within personas of 0.65 in contrast to a mean AAR between personas of 0.48. This difference is significant as measured by Welch's t-test ($t = 128.3, p < 0.01$), showing that the designer-defined utility weights for the different affordances clearly and consistently affect persona behavior.

Summary Statistics and Heat-Maps

Table 3 shows summary statistics for each persona, summed across all levels when different computational budgets are allocated. The results show similar patterns across all computational budgets and highlight the similarities and differences in the behaviors of the personas.

In general the personas fail at completing the levels. In a few instances they do succeed in reaching the exit, mainly in the case of the Exit persona, but in general they die before completing any levels. This may be due to a general weakness in the MCTS controller design. The controller does not implement domain knowledge in the play-outs, which are completely random. This can lead to most play-outs providing little information which in turn can hamper performance. It can, however, also be due to the fact that the levels are somewhat hard. Future work comparing persona skill to human player skill should investigate this.

In relative terms, the Exit persona is by far the most successful of the personas. Depending on the computational

budget, it completes between 15% and 39% of the levels. However, it may be argued that the Exit persona has the simplest decision making style, as it only has to optimize for reducing the distance to the exit, reaching the exit, and taking as few turns as possible; all of these goals align well within the rules of the game. The Runner persona, on the other hand, is typically unsuccessful in reaching the exit. The only difference to the Exit persona is a higher cost to spending a turn, but this changes the behavior to be unsuccessful. Notably, the Runner pursues almost no potions which stands in contrast to the Exit persona which seems to exploit potions to make it all the way to the exit. The Survivalist seems to focus shortsightedly on potions, which it collects to a great extent, but the number of potions in the levels seems to be too low to make this a viable strategy for also reaching the exit. The Monster Killer kills by far the most monsters of any persona, and collects a large number of potions too, to enable this mission. Finally, the Treasure Collector collects more treasure than anyone else, but typically fails at reaching the exit. This could either be because it prefers treasures over finishing the levels or the controller can not look ahead far enough to avoid trapping itself in fatal situations.

Figure 2 shows persona behaviors on various maps. The maps are chosen to highlight the behavior of each persona as different maps cater to different personas' decision making styles. The green color indicates tiles visited by the persona and the red color indicates tiles onto which the persona threw the javelin. Each map is shown in its final state.

All together the results indicate that the personas exhibit variation in behavior based on their utility functions. The differences in AAR values and summary statistics support this and the heat-maps serve as demonstrations. At the more general level, the personas exhibit a somewhat weak ability to play the game successfully, although this might be addressed by changing details in the MCTS controller implementation and by fine-tuning the personas' utility functions.

Discussion

In this paper we demonstrated that combining the idea of varying simple utility functions with Monte-Carlo Tree Search for agent control allows us to express archetypally different decision making styles in MiniDungeons 2. The decision making styles modeled in this paper do not represent models of observed human players but instead represent formal models of archetypal players imagined by game designers. On the one hand, it is possible to criticize this as being altogether different from the usual conception of data-driven player modeling. On the other hand, it is our belief that game designers always work with informal, imaginary player models when designing games, and that helping the designer formalize these assumptions procedurally may be valuable to the design process, in line with play persona theory (Canossa and Drachen 2009). The use of MCTS itself brings new possibilities to embedding synthetic play-testing in iterative level design processes. In previous work focusing on MiniDungeons 1 we have shown how other agent control methods, including reinforcement learning (Holmgård et al. 2014b) and neural networks configured through evolution (Holmgård et al. 2014a), can be used for the same purpose.

10 milliseconds					
	E	R	S	MK	TC
Turns	5368	3853	4106	4386	3982
Deaths	170	195	197	200	197
Monsters	721	531	571	741	531
Minitaurs	718	589	634	627	631
Treasures	111	90	78	86	175
Potions	57	11	34	32	14
Javelins	359	354	317	327	311
100 milliseconds					
	E	R	S	MK	TC
Turns	5017	2645	3328	4210	2778
Deaths	144	194	194	196	196
Monsters	800	481	554	931	490
Minitaurs	580	415	507	539	427
Treasures	113	77	84	92	210
Potions	75	0	57	65	3
Javelins	291	274	282	278	249
1 second					
	E	R	S	MK	TC
Turns	4116	2263	2831	3907	2513
Deaths	138	195	200	197	199
Monsters	761	482	570	969	476
Minitaurs	416	358	418	461	377
Treasures	110	57	68	88	203
Potions	45	0	70	63	5
Javelins	283	224	237	287	235
10 seconds					
	E	R	S	MK	TC
Turns	4066	2250	2576	3648	2603
Deaths	123	197	196	200	195
Monsters	769	455	544	1002	478
Minitaurs	459	372	375	403	417
Treasures	106	56	60	59	190
Potions	58	0	67	62	5
Javelins	243	233	225	237	217
60 seconds					
	E	R	S	MK	TC
Turns	3885	2110	2579	3611	2336
Deaths	122	197	199	198	192
Monsters	759	473	577	967	493
Minitaurs	408	337	392	406	349
Treasures	89	58	49	70	201
Potions	53	0	85	52	3
Javelins	230	202	215	253	223

Table 3: Play summary statistics summed across the 10 maps under different computational budgets: 10ms, 100ms, 1s, 10s, and 60s. Each condition totaled 200 runs per persona, 20 per map.

However, the methods used for that game would not perform well in MiniDungeons 2, which has a dynamic environment (monsters move); the models learned for MiniDungeons 1 presumed a static world. Other recent work has shown how game playing agents that play across multiple games (while not incorporating any explicit notions of style or preference) can be produced using a combination of reinforcement learning and neural networks (Mnih et al. 2015) or MCTS (Perez et al. 2015). While off-line machine learning-based

(a) Exit (b) Runner (c) Survivalist (d) Monster Killer (e) Treasure Collector

Figure 2: Heat-maps of end game states exemplifying differences in the personas' behaviors, taken from personas given $1s$ of decision making time. Different maps are chosen to best showcase the personas' decision making styles. Green indicates tiles a persona moved to and red indicates a javelin attack. In sub-figures (a) and (b) the Exit and Runner personas manage to finish the level, while in sub-figures (c), (d), and (e) the personas die before reaching the end of the level. The Survivalist pursues two potions, but ultimately dies, and the Monster Killer manages to kill all monsters in the level, but does not survive. Finally, the Treasure Collector gets all the treasure, but dies in the process.

player modeling methods perform well when trained for a sufficiently long time and with a sufficient amount of training material, the need for training also constitutes their major disadvantage. If a game designer fundamentally changes the rules of a game, previously trained agents or personas may be rendered invalid in an instant. In contrast, on-line search based methods like MCTS are capable of adapting to unseen problems within the rules of a game (e.g. new maps) or to rule changes. If we add a new monster with an entirely new behavior to MiniDungeons 2 and include this in the simulations of the MCTS personas, it is likely that the personas would retain their individual characteristics while responding to the new element of the game. Prior work in MCTS for general game playing supports this assumption (Finnsson and Björnsson 2008), but future work should focus on investigating this for the particular case of MiniDungeons 2. The scalable aspects of MCTS allow for a straightforward way of simulating player skill for personas. The results in this paper suggest that skill may be represented through computational budgets, and recent work has provided deeper insight into MCTS and skill for other games (Zook, Harrison, and Riedl 2015). It is an open question to what extent MCTS based personas can replicate the behavior of a human player on an unseen map in MiniDungeons 2. The current personas are defined by hand, and are in a sense extremes within the space of strategies that can be represented with the current set of primary utilities. When sufficient player data has been collected from players of MiniDungeons 2, we should try to learn personas directly from player behavior, by varying the utility weights of the MCTS agents until the best possible fit is obtained. Future work revolves around using these personas as level design critics in procedural content generation tools for MiniDungeons 2.

Conclusion

In this paper we have demonstrated how a number of simple utility functions, representing game designers' assumptions about player behavior, can be used with a Monte-Carlo Tree Search implementation to provide game playing personas that enact a gallery of decision making styles. We also argued that MCTS has a number of advantages for modeling player decision making styles: It matches contemporary high-level models of the human decision making process, it accepts utility functions in a straightforward manner, and can be adjusted to represent different skill levels by changing its computational budget. These characteristics make MCTS a good candidate for constructing generative player models across a wide range of games. Future work will focus on building MCTS-based data-driven generative player models from human play-traces, comparing these to the expert-knowledge-driven ones built from game designer expectations, and eventually applying these player models as critics for procedural level generation.

Acknowledgments

The research was supported, in part, by the FP7 ICT project C2Learn (project no: 318480), the FP7 Marie Curie CIG project AutoGameDesign (project no: 630665), and by the Stibo Foundation Travel Bursary Grant for Global IT Talents.

References

Bartle, R. 1996. Hearts, Clubs, Diamonds, Spades: Players who Suit MUDs. *Journal of MUD research* 1(1):19.

Bourki, A.; Chaslot, G.; Coulm, M.; Danjean, V.; Doghmen, H.; Hoock, J.-B.; Hérault, T.; Rimmel, A.; Teytaud, F.; Teytaud, O.; et al. 2010. Scalability and Parallelization of Monte-Carlo Tree Search. In *Computers and Games*, 48–58. Springer.

Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4(1):1–43.

Canossa, A., and Drachen, A. 2009. Patterns of Play: Play-Personas in User-Centred Game Development. In *Proceedings of the 2009 DiGRA International Conference*.

Champandard, A. J. Monte-Carlo Tree Search in TOTAL WAR: ROME II's Campaign AI. AIGameDev.com: http://aigamedev.com/open/coverage/mcts-rome-ii/.

Elias, G. S.; Garfield, R.; and Gutschera, K. R. 2012. *Characteristics of Games*. MIT Press.

Finnsson, H., and Björnsson, Y. 2008. Simulation-Based Approach to General Game Playing. In *AAAI*, volume 8, 259–264.

Fullerton, T.; Swain, C.; and Hoffman, S. 2004. *Game Design Workshop: Designing, prototyping, and playtesting games*. Focal Press.

Gigerenzer, G., and Gaissmaier, W. 2011. Heuristic Decision Making. *Annual Review of Psychology* 62:451–482.

Gigerenzer, G., and Selten, R. 2002. *Bounded rationality: The adaptive toolbox*. MIT Press.

Hingston, P., ed. 2012. *Believable Bots*. Springer.

Holmgård, C.; Liapis, A.; Togelius, J.; and Yannakakis, G. N. 2014a. Evolving Personas for Player Decision Modeling. In *IEEE Conference on Computational Intelligence and Games*.

Holmgård, C.; Liapis, A.; Togelius, J.; and Yannakakis, G. N. 2014b. Generative Agents for Player Decision Modeling in Games. In *Foundations of Digital Games*.

Holmgård, C.; Liapis, A.; Julian, T.; and Yannakakis, G. N. 2015. MiniDungeons 2: An Experimental Game for Capturing and Modeling Player Decisions. In *Proceedings of the 10th Conference on Foundations of Digital Games*.

Jacobsen, E. J.; Greve, R.; and Togelius, J. 2014. Monte Mario: Platforming with MCTS. In *Proceedings of the 2014 conference on Genetic and evolutionary computation*, 293–300. ACM.

Kahneman, D. 2011. *Thinking, Fast and Slow*. Farrar, Straus and Giroux.

Liapis, A.; Holmgård, C.; Yannakakis, G. N.; and Togelius, J. 2015. Procedural Personas as Critics for Dungeon Generation. In *Applications of Evolutionary Computation*, volume 9028, LNCS. Springer.

Lorentz, R. J. 2011. Improving Monte–Carlo Tree Search in Havannah. In *Computers and Games*. Springer. 105–115.

Mark, D. 2009. *Behavioral Mathematics for Game AI*. Course Technology Cengage Learning.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-Level Control through Deep Reinforcement Learning. *Nature* 518(7540):529–533.

Perez, D.; Samothrakis, S.; Togelius, J.; Schaul, T.; Lucas, S.; Couëtoux, A.; Lee, J.; Lim, C.-U.; and Thompson, T. 2015. The 2014 General Video Game Playing Competition. *IEEE Transactions on Computational Intelligence and AI in Games*.

Smith, A. M.; Lewis, C.; Hullett, K.; Smith, G.; and Sullivan, A. 2011. An Inclusive Taxonomy of Player Modeling. *University of California, Santa Cruz, Tech. Rep. UCSC-SOE-11-13*.

Togelius, J.; Yannakakis, G. N.; Stanley, K. O.; and Browne, C. 2011. Search-Based Procedural Content Generation: A Taxonomy and Survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):172–186.

Tychsen, A., and Canossa, A. 2008. Defining Personas in Games Using Metrics. In *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, 73–80. ACM.

Yannakakis, G. N., and Togelius, J. 2014. A Panorama of Artificial and Computational Intelligence in Games.

Yannakakis, G. N.; Liapis, A.; and Alexopoulos, C. 2014. Mixed-Initiative Cocreativity. In *Proceedings of the 9th Conference on the Foundations of Digital Games*.

Zook, A.; Harrison, B.; and Riedl, M. O. 2015. Monte-Carlo Tree Search for Simulation-based Strategy Analysis. In *Proceedings of the 10th Conference on the Foundations of Digital Games*.