Game Design: A Preliminary Review on Techniques and Tools

Jorge Ruiz Quiñones and Antonio J. Fernández-Leiva

Departamento de Lenguajes y Ciencias de la Computación,
Universidad de Málaga (Spain).
jorge.ruiz@alu.uma.es, afdez@lcc.uma.es

Abstract. This paper is a preliminary attemps to serve as a reference of the state of the art of the techniques and tools related to '*Game Design*' (GD). We analyze the relationship existing between Game Design and the main subareas related to Artificial and Computational Intelligence in Games. Here, we focus particularly in Techniques and Tools that help in any way to designers involved in game development.
Keywords: *Game Design, Tools, Artificial Intelligence*

1 Introduction

This paper focuses on Game Design and, in particular, in the tools helping game designers in any way. It is a first step in the search of a more ambitious objective consisting in '*Automating the Generation of Complete Games*'. The paper describes our first considerations to determine and define the elements that are already in use or under investigation as a possible source to reach this goal.

Here, we provide a preliminar analysis of the current situation of Game Design under different perspectives. In a first approach (section 2.1), a brief summary of the most used technique in the field of Artificial Intelligence in Games (AIG) is presented. Second (Section 2.2), we analyze how the 10 main Game Research subareas of AIG [1, p. 317] are interconnected and related with Game Design. Then, from the point of view of Tools oriented to Game Design, some of the most relevant existing works up to date are mentioned and classified according to the AIG subarea that it covers as well as to their commercial purposes. We also provide a brief description of the Tools described. The recognition and enumeration of these tools may be useful later on in the investigation where this document is framed into.

Inside AIG, much of the publications related with Game Design have been oriented to Procedural Content Generation (PCG) helping designers to build different kind of contents for games such as levels, bitmaps, maps, items, etc. Nowadays, other areas such as Narrative, have become a promising research field, with many surveys to take into account (PCG is still the leader in this aspect) but including important advances in helping tools like Twine [2] or Inform 7 [3]. Although we can find works oriented to generate elements not coupled to a

concrete game (PCG, Narrative or any other area, see section 3), usually they are oriented to a concrete game or platform. Commercial 2D platform games such as Mario Bros or first person shooters like Unreal (with about 50 relevant publications each in lasts years) are just a couple of examples of games/engines that have been included in lots of articles and publications. As this survey is framed into an investigation whose goal is to generate complete games in the Design level, we focus here in general purpose tools and techniques.

This paper enumerate used techniques and current tools focused in Game Design from a generic point of view but, although we focus in general/abstract tools oriented to Game Design, it is remarkable that the importance of Game Design Tools is apparent on today's games as far as there are several examples of commercial games that include their own tools (mainly focused to create maps or levels). For mentioning some of them, the World Editor included in Sid Meier's Civilization Saga, the recently released Super Mario Maker for different consoles platforms or Warcraft III world editor are some examples to be taken into account. These features give the game an extra value and highlights that is important nowadays for the general public to have this kind of tools even in commercial games.

2 AI Techniques

Game AI, in general, and more concretely AI Assisted Game Design can be defined in part as a group of techniques of AI that are applied to the complete development process of a Game including not only processes related to developing phases but as well related to the realtime gameplay (for instance, decisions or pathfinding for agents). Techniques are composed by methods, processes and algorithms, all of them related in a particular way with AI. All of them can be applied solely for a concrete task but may be gathered for obtaining better performance during the process of Game Design. For instance, use Evolutionary Computation (EC) for finding the best solution representation in consonance with Search techniques for exploring efficiently a wide range of possible solutions.

2.1 List of techniques

The most relevant techniques used up to date in AI Game Design are:

Bio-Inspired computing: In a general way, we consider bio-inspired computing as a technique that tries to apply any working example in Nature to solve complex problems using algorithms and representations that mimic real scenarios in Nature and reproduce them as a computing problem. There are some examples of this kind of algorithms used in AI in games, and more concretetly, in Game Design. Evolutionary Computation can be considered as a family of algorithms inspired by biological aspects or elements such as evolution, natural selection, animals behaviors, mutations, etc...based in a population that reflect a concrete representation of singular elements and the solution of the problem.

This population can evolve (and mutate) as it tries to find the best most optimal solution for the problem.

Machine Learning: This term was coined in 1959 [4] and is known as the ability of a computer (process) to learn without being programmed explicitily. Algorithms based in Machine Learning are those who can learn from a data set or subset and are able to make predictions using that data. It can be divided in several different approaches but, regarding to AI in Games, we can highlight:

- Reinforcement Learning (RL): The way that a process can take decisions in order to maximize the reward of a concrete solution. Actions are associated to a positive or negative reward. Related to Game Design, RL is applied for evolving strategies or Non Player Characters (NPCs), trying to maximize any of the concepts relative to the game itself (score, winning conditions, duration of game, etc...)

- Supervised Learning: Learning through well known and labeled example data that produces a concrete output, the process aim is to infer a function that fits (maps) the training data and can be used to map new example data not included in the initial study.

- Unsupervised Learning: Contrary to the previous point, unsupervised learning tries to infer a function using example data which is not categorized or labeled previously. While Supervised Learning refers to learning a model that maps instances of datasets to target values, Unsupervised Learning tries to find patterns in datasets that do not have target values. Regarding to Game Design, Machine Learning technique is mainly used in NPC and Player Modelling.

Search and planning: Search and Planning techniques are related directly with AI in Games subareas such as PCG or NPCs. This technique may be used in an inferior rate that Evolutionary Computation is, but has a relevat role as well mainly in exploring solutions space in an optimized way. Planning is used often basically to build processes that generate any kind of '*plans*' such a path from one point to another or a sequence of transitions between states. And consider as well Searching as an important extra contribution to other techniques, most often used to search efficiently the most promising elements (may be gencoded algorithm solution in evolutionary computation) for obtaining the next generation of possible solutions, combining the best parts of all ancestors.

2.2 Techniques applied to Game Design

Artificial Intelligence assisted Game Design: The Artificial Intelligence takes a main role in Game Desgin due to its capacity to contribute with tools that support almost all aspects of Game Design. It's been around 35 years from the first conference related to video games [5] and during the earlier years of study (and not so far away in time), AI was mainly focused on NPCs and, a bit later on, pathfinding/strategies. Artificial Intelligence in computer games used to covered mainly the behavior and decision processes and populate the environment with characters that require any kind of human intelligence and behavior [6]. This situation has changed strongly and more recently AI has been redefined [7] to a new much more general concept related to Game Design. As

we already mentioned, nowadays it's not only AI as simply generating NPCs and strategies (the first historic approach) but also it's a key concept and taking part in almost all the rest of subareas. Moreover, its current scope has been divided into four subareas, namely, General Game AI, AI as benchmarks, AI in comercial games and **AI assisted Game Design** (AIGD).

AIGD may be the most promising research area [7, 1] and also the one that concentrates numerous researches (thousands of publications in the last three years in the most important channels including IEEE Xplore, ACM Digital Library). From the point of view of Game Design tools, we can consider AI as a tool itself, making possible to achieve advances in other areas providing tools that will help in the process of designing. Not only that but, maybe out of scope of this paper, AI contributes with several tools that are directly oriented to be used by developers and apply AI to games, organizations and general purpose products We'll focus on those related to Game Design.

Regarding to Techniques, it is obvious that Artificial and Computational Intelligence provides a wide range of techniques usually applied to generate game content, NPCs or agents, game narrative and stories, etc. Just to point out several of them, we can highlight bioinspired algorithms (like Evolutionary and Co-evolutionary Computation, Genetic algorithms, ants colony search), machine learning (like Reinforcement, supervised and unsupervised) or Planning/Search algorithms (like Montecarlo Tree search). The main techniques involved with AIGD are mainly Evolutionary Computation and Planning. This methods are used in realtime for obtaining strategies and behaviors (to be applied to NPC, for instance) and may be mixed as well with Learning methods (usually RL) to evolve a population representing game elements, such a strategy, NPC AI and, with less relevance, may be applied to other aspects of the game as game rules.

EC is actually the most used in subareas such as Non Player Character (NPC) behavior learning, Player Modelling, Procedural Content Generation (PCG) and **AI Assisted Game Design**. Although it can be considered the most used technique, Evolutionary Computation is not the only example, as it can be found as well examples of Neuroevolution (neural networks, related as well with machine learning) like in FORZA Motorsport 6 racing game to create avatars that learn from players and others like Cognitive Modeling [8] focused in player strategies modeling.

Behavior Trees (BT) is a method often framed into Machine Learning (but also Planners) techniques and are used widely in Game Design tools, like for instance in Behaviour Bricks [9](work still in progress), a tool supporting and helping designers, in many cases without any programming skills, to participate actively in the behavior development process and not to be focused just in the previous design and subsequent adjustments. It implements a model of BT with special emphasis in reusing behaviors and using customizing parameters. This kind of trees is used as well in "Behavior Designer - Behavior Trees for Everyone", available in Unity Asset Store and focused in helping not only programmers but also designers and artists. It has lots of features, including creating believable Agents with a provided visual editor. Another well known example of using

behavior trees is the Unreal Engine, which is defined as a *powerful tool to create Artificial Intelligence* combining the AI memory (the blackboard) that keeps the BT values and the BT as is, in charge of taking decisions and performing actions accordingly.

Programming by demonstration (PbD), an example of Learning technique[1] is a technique also used in already mentioned Behavior Bricks for obtaining trained nodes based on end-user interactions and not only as a tool but also as an environment like Pong Designer [10], used even for building complete simple games. Kodu [11], can be settled as well as another example of PbD.

To mention other examples, for instance both neuroevolution and machine learning is also used in authoring tools like NERO [12] and trees search based techniques are mainly used in narrative tools like Twine [2] or Inform 7 [3]

Computational Narrative (CN): CN affects directly both the evolution of the game and in the player experience. So, it is not only about content or elements that will be generated during game design/development, but also elemnts that have influence during realtime game playing. So, it is important to think in tecniques that can be applied accepting realtime requirements in order to not affect the gameplay performance. Usually, decision trees [2] are used during design phase to apply the narrative aspects and define the game story. During gameplay, it will be a mix of pathfinding (performing any kind of heuristic search using A* algorithm) and player experience who decides mostly the game story.

Procedural Content Generation: PCG is highly interconnected with Game Design in a sense that the PCG is used often to create and generate different types of contents that are key concepts of the Game Design. For instance, we can mention elments such as maps, items, characters, enemies, bitmaps and so on... PCG, as any kind of Game content generation, could be any of mixed (techniques and tools) initiative AI assistance.

According to techniques, and contrary to AI Game Design which is the area with the most diverse and richest palette of AI methods [6], PCG is widely dominated by evolutionary computation. Thinking in PCG as a process which is usually executed offline (not during game playing), it seems that evolutionary computation fits so much with this concept, because it is usually involved in processes that are related with heavy and large computation tasks. Once the content has been generated, it can be used by any real-time process during gameplay, but the process itself to generate that content can be considered 'offline'.

Non Playable Character and Agents: Also known as NPC, refers to any character involved in the game, not controlled directly by a human, so is complete responsibility of computational processes to manage and control these elements. This subarea is related with GD in several ways. It's not only to define how these characters are going to be represented during the game but also their characteristics and **behavior** which is the most relevant concept related to NPC. Methods related with NPC behavior learning are Evolutionary Comptuation and Reinforcement Learning. This methods are focused in two main aspects. First, in finding a way to improve the NPC behavior so that the NPC can be competitive

[1] https://en.wikipedia.org/wiki/Programming_by_demonstration

during the game play and, second, trying to make the NPC as much believable as possible. In this subarea, we could mention Behaviuor Trees (deciding which of a set of behaviors should be chosen in a concrete situation of the game) or Pathfinding (automatically find out which is the best move in map according to the state and targets of the game/agent) as the most used techniques. To mention some examples of using of concrete algorithms, we find for example A* combined with Influence Map [13] or Layered Learning [14], as a particular case of machine learning technique used to train agents.

3 AI and Tools in Game Design phases

The contributions on Game Design tools have increased significantly in lasts years [15]. These tool are centered in assisting the process of creation of game in any sense. It is important to notice that, currently, most of the tools are being developed for specific games and/or platforms/AI and is not easy to find general purpose tools oriented to game design [16], independents from the game or platform that they are going to be used with. Developing a general tool must take into account all tasks related to game design and be able to assist designers in all that tasks or procedures and also identify how to create a model of that tasks in order to improve it the way that the tool could be optimized.

In general, most relevant tools developed up to date are oriented to PCG helping designers to build new levels, worlds or stories (narrative) to be incorporated or used into a game, but also GD tools are used in other areas, such as mechanics (rules) or agents. Other areas less rich in provided Tools is NPC, contributing with tools helping to create Believable Agents for simulating playthroughs or generate agents automatically. So, this a promising area to research in.

As we have already seen in this document, AI techniques has a significant role in Game Design, taking part in several different processes related to Game development phases. Following, a brief summary of Game Design phases is presented and AI Techniques and Tools are identified for each of them. Also, Table 1 represents a summary of all mentioned tools with their main characteristics.

3.1 Technical specifications

This phase focuses mainly in determining technical aspects of the game, such as frame rate, screen resolution and color depth, and also in some game details like the number of players, NPCs or allowed game modes. Because of the idea of generating complete games, tools oriented to generate specifications automatically should include any way to *coding* this information and the capacity yo generate all those aspects using any kind of AI technique. This is a very promising and unexplored game design phase with no tools associated to it.

3.2 Defining Game Story - Narrative

The game main story, differents stories's timelines, secondary stories and also a storyboard for the game's intro and other video scenes during the game can

Tool - Phase	PCG	MR	NRR	AG	COMPG	COMM	RES	OS	FREE
Tanagra	✓						✓		
Sketcha World	✓					✓			
Sentient World	✓						✓		
Sentient Sketchbook	✓						✓		✓
SpeedTree	✓					✓			✓*
Charack	✓						✓		
NERO				✓			✓		✓
MetaGame		✓					✓		
RuLearn		✓					✓	✓	✓
ABL			✓	✓			✓		
TADS			✓				✓		✓
Inform 7			✓					✓	✓
Twine			✓			✓		✓	✓
Kodu					✓	✓			✓
Ceptre		✓					✓	✓	✓
PuzzleScript					✓			✓	✓
Pong Designer					✓		✓		
Ludi					✓		✓		
Machinations		✓						✓	✓
Total	6	4	4	2	4	4	12	6	10, 1*

Table 1. This table shows the most relevant tools related with Game Design according the Game AI areas they are focused in (that is to say: PCG: Procedural Content Generation, MR: Mechanics and Rules, NRR: Narrative, AG: Agents, COMPG: Complete Games) and according to commercial aspects (that is to say, COMM: Commercial purposes, RES: Research purpuses, OS: Open Source, FREE: Free license to use in game development). * denotes the existence of a Free evaluation version.

be framed into Narrative subarea. This phase becomes really important (if not the most important) in some cases like in adventure games and requires specific software helping designers during these tasks. Focused on Narrative, Twine [2] is a powerful free and open source tool helping creating stories for games, successfully used in "Howling Dogs" and "Depression Quest". Inform7 is another example to mention. Platform independent, is a visual tool used for creating in a easy way Interactive Ficition (IF) for games. Following, a list of the most interesting tools regarding to Narrative:

- TADS [17]: Text adventure development system (TADS) is a free authoring system for writing your own Interactive Fiction (IF). It offers a complete set of programming tools for creating high-quality IF.
- Inform 7 [3]: Inform is a design system for interactive fiction based on natural language. It is a radical reinvention of the way interactive fiction is designed, guided by contemporary work in semantics and by the practical experience of some of the world's best-known writers of IF.

– Twine [2]: Twine is an open-source tool for telling interactive, nonlinear stories. There's no need to write any code to create a simple story with Twine, programmers simply extend stories with variables, conditional logic, images using simply CSS, and JavaScript..

3.3 Generating Game Content

All concepts related to game content such as bitmaps, characters, weapons, maps, levels, environments, music, etc can be generated using AI in any way. Procedural Content Generation (PCG) has been pioneer and historically the subarea with the most contributions in research aspects and is as well the subarea with more game desing oriented tools dedicated to. As comented in the Introduction section of this document, content generation is being more and more important as far as many games include their own software for adding or editing content, and not only that but also games fully dedicated to 'create' that content and real-time play with it. Following, a list of the most interesting tools regarding to PCG:

– Tanagra [18]: A prototype 2D games generic Level Design Generator tool helping designers to create levels and manipulate levels. It is a mixed initiative (PCG-Assisted game Design) focused in improving the designer experience.
– SketchaWorld [19]: Integrated and very accessible modelling tool which combines 3D modelling and semi-automated techniques for building a world.
– Sentient World Simulation [20]: Builder of a sythetic mirror of the real world with automated and continous calibration with respect to current real-wolrd information. SWS consists of components capable of capturing new events as they occur anywhere in the world, focus on any local area of the synthetic world offers sufficient detail. In other words, the set of models that make up the synthetic environment encompass the behavior of individuals, organizations, institutions, infrastructures and geographies while simultaneously capturing the trends emerging from the interaction among entities as well as between entities and the environment.
– Sentient Sketchbook [21]: A tool which supports a designer in the creation of game levels.
– Speedtree [22]: Render Software for generating animated plants and trees for games (but also used in other areas). It has been used in many games in lasts 10 years fully integrated with Unity platform.
– Charack [23]: A tool able to generate pseudo-infinite virtual worlds with different types of terrains. Using a combination of algorithms and content management methods, Charack is able to create beaches, islands, bays and coastlines that imitates real world landscapes.

3.4 Defining Rules and Mechanics

Game Mechanics and rules usually governs the logical and functional evolution of a game. There are some references that highlights the importance of this

area, naming them as *the core of a game* [24]. Even in some cases, generating game's rules has been considered, as a generation of games [25]. Taking them into account and using tools based in AI techniques would be powerful to combine and generate new sets of rules or mechanics, and so, we could say *new* games as far as the game rules are different. Exploring using any of the search methods such as Tree Search and combining it with bioinspired algorithms [26], may result in good solutions to evaluate and to start from to find new promising rule sets to generate new games. Some examples of tools based on AI and oriented to generate game mechanics are:

- Meta Game [27]: Meta-Game Playing (Metagame) is a paradigm for research in game-playing in which programs can be designed to take in the rules of unknown games and play those games without human assistance. Strong performance in this new paradigm is evidence that the program, instead of its human designer, has performed the analysis of each specific game.
- RuLearn [28]: An open-source toolkit for the automatic inference of rules for shallow-transfer machine translation from scarce parallel corpora and morphological dictionaries. RuLearn will make rule-based machine translation a very appealing alternative for under-resourced language pairs because it avoids the need for human experts to handcraft transfer rules and requires, in contrast to statistical machine translation, a small amount of parallel corpora (a few hundred parallel sentences proved to be sufficient).
- Ceptre [29]: Ceptre is a rule specification language intended to enable rapid prototyping for experimental game mechanics, especially in domains that depend on procedural generation and multi-agent simulation.
- Machinations [30]: Machinations is a conceptual framework and diagram tool that focuses on structural qualities of game mechanics. An interactive and visual tool is provided with the framework to allow drawing and run Machinations diagrams.

3.5 Strategies and Agents

- NERO [12] [31]: Authoring Tool NERO is a machine learning game in which the player uses real-time neuroevolution to train a team of robotic soldiers for combat. Training takes place in a sandbox, where the player can place obstacles and enemies, and change the fitness function according to the skills that should be learned. After training, players can pit their teams against other teams in combat.
- ABL [32]: A Behavior Language for Story-Based Believable Agents (ABL) is a programming language explicitly designed to support programming idioms for the creation of reactive, believable agents. ABL has been successfully used to author the central characters Trip and Grace for the interactive drama Facade (Mateas and Stern, 2003). The ABL compiler is written in Java and targets Java; the generated Java code is supported by the ABL runtime system

3.6 Generating complete games

We have to notice that nowadays exist general purpuse tools designed for helping in all game design phases and generating complete games, called usually *Game Engines*. For just naming a couple of them, Unity 3D, Unreal Engine,or CryEngine.Although these tools/engines are supporting game developers to create games, are out of the scope of this document as far as is the developer/Designer who is in charge of build the game and, usually, doesn't generate games automatically but programatically.

What about generating complete games automatically? It's clear that this task should be much more complex and ambitious that simply focus in a concrete desgin phase and difficult to realize. When generating complete games, should be necessary as well an evaluation method (also called fitness function) that allows to evaluate every game individually in order to have an idea of *how good this game is*. There are some examples of researches that are focused in this path, but with some restrictions. Usually, tools for generating complete games are focused in a concrete type of games, for instance, board games [25] or tile based games [33] and may be difficult (if not impossible up to date) to find a tool to generate games not tied to any kind of games. Difficult as well is to find any kind of fitness function applied to these games, so we could say this is an unexplored and very promising area. Following, example of tools oriented to generate complete games:

- PuzzleScript [33]: PuzzleScript is a game engine designed to help game developers to make tile-based puzzle games.
- Kodu [11]: Kodu lets designers create games on the PC and Xbox via a simple visual programming language. Kodu can be used to teach creativity, problem solving, storytelling, as well as programming.
- Ludi GDL [25]: Language that allows designer to define game's rule sets and provides a player (Ludi General Game Player) that parses those rule sets allowing to play games between human players and NPC agents.
- Pong Designer [10]: An environment for developing 2D physics games through direct manipulation of object behavior.

4 Conclusions

As seen in this survey, we find that there are two main concepts to focus. First, the idea is that game design tools are mainly oriented to one specific game. We feel that it is really interesting to analyze the role of AI for developing new games or tools that are not tied to an specific game or platform (that is to say, a generic perspective for AIG). Second, the majority of tools related with game design are oriented to generate specific content for games. This situation is a good starting point for our research taking into account that complete game generation is one of the most interesting (and hard) challenges for the application of Artificial Intelligence in Games.

Game processes as well as realtime gameplay are also elements well studied that attract the attention of the research community. In this context, it is mandatory to take into account the process of **Game Evaluation**. This is a very unexplored area that offers many possibilities of investigation from the point of view of techniques to be applied and possibly tools that help designers and players to evaluate, compare and rate automatically Games. This concept is really important if we think in generating games automatically, because it is needed for accomplishing the complete process of generation without any human intervention.

Usually, most of the publications up to date related with Game Design tools are framed and tied to a concrete platform or game. Thinking in how to open the mind in this aspect will guide us to find more powerful tools to help build games in any sense. Some tools mentioned in this document, oriented to a more general concept of automatic games generation are very tied to a concrete type of games (board or tiles games, puzzles, etc...). Generating games not tied to any specific genre is a challenge from the research point of view.

We have presented some tools aimed to generate complete games, but with restrictions as far as they are oriented to a particular game type such as tile or board games. The first phase in game development that involves technical decisions (i.e., determining frame rate, game resolution, type of games, number of players, etc.) corresponds to the Technical Definition phase in Game Design. We feel that there is a big gap related to both tools and ways to generate game elements automatically. The tools already mentioned in this document usually obviate this phase. For us, this phase represents a a starting point in the search of a schema/model to generate games completely.

References

1. Georgios N. Yannakakis, J.T.: A panorama of artificial and computational intelligence in games. IEEE Transactions on Computational Intelligence and AI in Games **7**(4) (2015)
2. Klimas, C.: Twine, https://twinery.org/ (2009)
3. Inform: Inform 7 - design system for interactive fiction, http://inform7.com/ (2014)
4. Samuel, A.L.: Some studies in machine learning using the game of checkers. IBM Journal of Research and Development **3**(3) (1959) 210–229
5. Mitchell, E.: Video games visit harvard yard, Antic, 2:24 (1983)
6. Artificial Intelligence in Games, International Journal of Soft Computing and Engineering (IJSCE) (2012)
7. Yannakakis, G.N.: Game ai revisited. In: Proceedings of the 9th Conference on Computing Frontiers. CF '12, New York, NY, USA, ACM (2012) 285–292
8. Camerer, C.F., Ho, T., Chong, J.K.: A cognitive hierarchy model of games. The Quarterly Journal of Economics **119**(3) (2004) 861–898
9. Olivenza, I.S.: Aplicación de técnicas de aprendizaje automático supervisables por el diseñador al desarrollo de agentes inteligentes en videojuegos (2017)
10. Mayer, M., Kuncak, V.: Game programming by demonstration. In: Proceedings of the 2013 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software. Onward! 2013, New York, NY, USA, ACM (2013) 75–90

11. Microsoft-Research: Kodu game lab, https://www.kodugamelab.com/ (2009)
12. NEROGame.org: Nero game, http://nerogame.org/ (2017)
13. Prihatmanto, A.S., Prasetyanto, F., Harsoyo, A.: Strategy and behavior models of non-playable character using computational intelligence approach. In: 2013 IEEE 3rd International Conference on System Engineering and Technology. (2013) 411–414
14. Mondesire, S., Wiegand, R.P.: Evolving a non-playable character team with layered learning. In: 2011 IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MDCM). (2011) 52–59
15. Yannakakis, G.N., Togelius, J.: A panorama of artificial and computational intelligence in games. (2014)
16. Togelius, J., Yannakakis, G.N.: General general game ai. In: IEEE Conference on Computational Intelligence and Games (CIG'16), Press (2016) 468–473
17. Vicarious: Tads - text adventure development system (2017)
18. Smith, G., Whitehead, J., Mateas, M.: Tanagra: a mixed-initiative level design tool. In: Proceedings of the Fifth International Conference on the Foundations of Digital Games, New York, NY, USA, ACM (2010) 209–216
19. TNO: (Sketchaworld: From scketch to virtual world, https://www.tno.nl/en/focus-areas/defence-safety-security/missions-operations/sketchaworld-from-sketch-to-virtual-world/)
20. Tony Cerri, D.A.C.: Sentient world simulation (sws) (2006)
21. Antonio Liapis, J.T., Yannakakis, G.N.: Sentient sketchbook: Computer-aided game level authoring. (2013)
22. Interactive Data Visualization, I.I.: Speedtree (2016)
23. Bevilacqua, F., Pozzer, C.T., d'Ornellas, M.C.: Charack: Tool for real-time generation of pseudo-infinite virtual worlds for 3d games. In: 2009 VIII Brazilian Symposium on Games and Digital Entertainment. (2009) 111–120
24. Zook, A., Riedl, M.O.: Automatic game design via mechanic generation. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada. (2014) 530–537
25. Browne, C.: Automatic generation and evaluation of recombination games (2008)
26. Shaker, N., Togelius, J., Nelson, M.J.: Procedural Content Generation in Games: A Textbook and an Overview of Current Research. Springer (2016)
27. Pell, B.D.: Strategy generation and evaluation for meta-game playing. Technical report (1993)
28. Sánchez-Cartagena, V.M., Pérez-Ortiz, J.A., Sánchez-Martínez, F.: Rulearn: an open-source toolkit for the automatic inference of shallow-transfer rules for machine translation (2016)
29. Martens, C.: Ceptre: A language for modeling generative interactive systems. (2015)
30. Dormans, J.: Machinations: Game feedback diagrams (2017)
31. Stanley, K.O., Cornelius, R., Miikkulainen, R., D'Silva, T., Gold, A.: Real-time learning in the nero video game. In: Proceedings of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. AIIDE'05, AAAI Press (2010) 159–160
32. Mateas, M., Stern, A.: A behavior language for story-based believable agents. IEEE Intelligent Systems **17**(4) (2002) 39–47
33. Lavelle, S.: open-source html5 puzzle game engine, http://www.puzzlescript.net/ (2017)