Serdica J. Computing **10** (2016), No 2, 133–166

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

MODERN TRENDS IN THE AUTOMATIC GENERATION OF CONTENT FOR VIDEO GAMES

Boyan Bontchev

Abstract. Attractive and realistic content has always played a crucial role in the penetration and popularity of digital games, virtual environments, and other multimedia applications. Procedural content generation enables the automatization of production of any type of game content including not only landscapes and narratives but also game mechanics and generation of whole games. The article offers a comparative analysis of the approaches to automatic generation of content for video games proposed in last five years. It suggests a new typology of the use of procedurally generated game content comprising of categories structured in three groups: content nature, generation process, and game dependence. Together with two other taxonomies – one of content type and the other of methods for content generation – this typology is used for comparing and discussing some specific approaches to procedural content generation in three promising research directions based on applying personalization and adaptation, descriptive languages, and semantic specifications.

ACM Computing Classification System (1998): F.1.1, F.2.2, K.8.0.
Key words: video games, content generation, descriptive, personalization, semantics.

1. Introduction. In recent decades, computer games have enjoyed a diverse global audience and had a great impact on the new multimedia culture worldwide. They started as simple text-based games in the early seventies of the last century and, with the development of modern 2D and 3D video and game console technologies, passed through several generations of video games [1]. Nowadays, young, middle-aged and old people play video games, whereby the average game player is 35 years old and plays mostly social, action and puzzle/board games [2]. Computer games are an effective and interactive means for retaining the interest of an audience by attracting attention for much more time than traditional media. Playing video games results in many benefits regarding the development of cognitive skills (such as enhanced attention, creativity and problem solving), motivation (persistence to failures and learning by failure), emotional skills (flow experience, mood management and adaptive emotional regulation), and prosocial behavior and civic engagement [3]. This is valid not only for games for entertainment but also for serious games applied to education and training, defense, scientific modelling, engineering, health care, advertising, politics, etc. [4].

The strong penetration of video games into the everyday life of modern society fosters a multi-billion market of game hardware, accessories, and content, where game content takes two-thirds of the whole [2]. The need of novel and attractive game content will continue playing the most important role for keeping players highly motivated and emotionally engaged in virtual interactive worlds [5]. Although players started customizing game content, there is a gap between the manual content production and the demand for new game content fueled by an exponential growth of both the gamer community and the production costs. Allocating more human resources at game companies misses scalability and tends to be rather expensive; therefore, it cannot serve as a solution to the problem of ever-increasing demand for game content. On the other hand, Procedural Content Generation (PCG) provides a viable alternative to manual content production, because it results in automatization of the production of a specific type of content based on a set of input parameters [6]. PCG has been applied in many popular commercial games for generation of various types of content [7, 8] such as dungeons in *Rogue* (Toy and Wichman, 1980) and *Diablo* (Blizzard Entertainment, 1996), star systems

in *Elite* (Acornsoft, 1984), maps in *Civilization* (MicroProse 1991), vegetation in *SpeedTree* (Interactive Data Visualization, 2003), terrains in *Dwarf Fortress* (Bay 12 Games, 2006), weapons and shields in *Borderlands* (Gearbox, 2009), and the world of *Minecraft* (Mojang, 2011), in order to enhance the game by adding more realism, variety and complexity to the virtual fantasy world. At the same time, PCG has been used in serious games for education and training [9, 10] in order to foster the immersion, flow, and learnability of such games.

PCG for games is defined as "application of computers to generate game content, distinguish interesting instances among the ones generated, and select entertaining instances on behalf of the players" [11]. Players may have indirect or direct control over PCG—for example, procedural generation of game stories can automatically branch the main story according to player choices [12]. Some authors [13, 6] treat game content as referring to all aspects of a game that affect gameplay excluding behavior of non-player characters (NPC), which includes gaming issues such as rules, dynamics, character attributes, user interface, sound, level design, maps, terrain, story, quests, and player's inventory such as health, weapon, and munitions. In other studies [11, 14], generated game content applies to NPC in order to help the creation of believable characters and social agents. It may include automatic generation of tactics [14], dynamic dialogs with context generated from both episodic memory and emotional valence of previous social interactions [15], and agent behavior generated from planning graphs including natural language generation [16].

This article tries to summarize achievements in procedural game content generation concerning all game aspects affecting gameplay including the behavior of NPC. The considerations are outlined in the scope of two comprehensive taxonomies proposed in [11]—one of game content including six layers: bits, space, systems, scenarios, design, and derived, and one of the common methods for PCG. The article proposes a new typology of use of PCG for video games comprising categories structured into three main groups: content nature, generation process, and game dependence. Together with the taxonomies of content type and common methods for PCG, this typology is applied for comparing and discussing specific approaches to procedural content generation, with a focus on their importance regarding type, methods, and

usage of procedurally generated game content. The comparative study is conducted within three research directions identified as most promising:

1. personalized procedural content generation;
2. content generation using game descriptive languages;
3. content generation using semantic world representations.

Along with the comparison, we discuss the open problems of the methods for PCG and recapitulate their advantages and disadvantages. The article concludes with some remarks about the future trends of automatic generation of content for video games.

2. Procedural content generation in games. When summarizing practices and experiments for PCG, game designers have to answer at least the simple questions: "What?", "How?", and "Where?". With this purpose, taxonomies of procedurally generated game content, of generation methods and of use of generated game content are crucially important. Hendrikx et al. [11] surveyed both game content types and methods used for procedurally generating game content. Their effort resulted in two taxonomies outlined below—one of procedurally generated game content and another of methods of PCG. Since no classification of use of generated game content had been proposed to date, we created a new one extending the preliminary distinctions suggested in [7] for clarifying the role of search-based PCG.

2.1. Taxonomies of procedurally generated game content and of methods of PCG. The taxonomy of procedurally generated game content includes five main classes of content that can be generated procedurally for using strictly inside games, and an additional class of generated content derived from a game in order to be used for attracting players further to the game world [11]. Each class contains several identified subclasses of content, which can be either abstract or concrete. As shown in Fig. 1, the taxonomy can be structured as a six-layer pyramid where upper classes may include content from the lower classes. The six layers include sub-classes as follows:

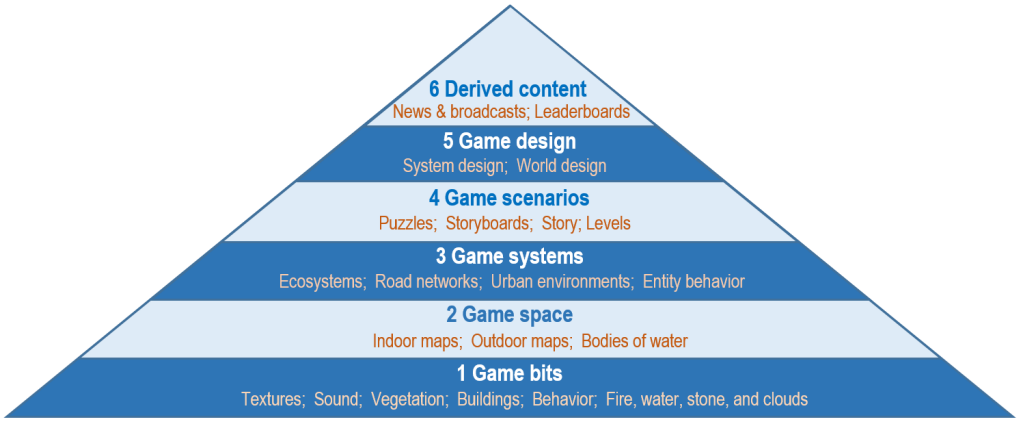

Fig. 1. Taxonomy of procedurally generated game content (after [11])

- Layer 1: Game bits—represent elementary game assets, which are concrete (interactive items of the game world) or abstract (such as textures and sound used for the creation of concrete bits). Identified game bits include textures, sound, vegetation, buildings, behavior (of objects interacting with each other or with the environment) and also fire, water, stone, and clouds.
- Layer 2: Game space—denotes the game environment and can be defined in a concrete way (as in a multi-leveled dungeon) or an abstract one (e.g., the board in backgammon). The game space sub-classes are indoor maps, outdoor maps, and bodies of water.
- Layer 3: Game systems—make the game more attractive and realistic, therefore immersive and believable, and include ecosystems, road networks, urban environments, and entity behavior (e.g., behavior of an NPC based on player actions and interactions).
- Layer 4: Game scenarios—describe the order in which game events evolve and foster player motivation and engagement; may be presented in the game in an abstract way (e.g., object interrelations) or a concrete one (for example, game narratives) way. The challenge is to generate automatically break-through stories with branching based to player input choices [12]. Identified game scenarios are puzzles, storyboards, story, and levels.

- Layer 5: Game design—here, Hendrikx et al. [11] referred to the game design vision of [17][1], which is comprised of system design (includes mathematical patterns and game rules) and world design of concrete setting, story, and theme. The authors found no commercial games applying procedural generation of game design.
- Layer 6: Derived content—defined as "content that is created as a side-product of the game world" [11] and including news, broadcasts, and leaderboards (player ranking tables). In light of the growing interactions among gamers and game stakeholders in social networks, we think social media posts about concrete video games should be included here as well.

Beside the taxonomy of procedurally generated game content, Hendrikx et al. [11] defined a taxonomy of common methods for PCG. All the methods studied by the authors had been applied successfully in commercial games and applications using virtual worlds. They identified five fundamental classes of methods for PCG, as follows:

1. Pseudo-random number generators.
2. Generative grammars—Lindenmayer-systems, split grammars, wall grammars, and shape grammars.
3. Image filtering—binary morphology and convolution filters.
4. Spatial algorithms—tiling and layering, grid subdivision, vectorization, fractals, and Voronoi diagrams.
5. Modeling and simulation of complex systems—cellular automata, tensor fields, agent-based simulation, and other complex systems and theories.

[1] Another popular vision of game design appears to be the MDA model (standing for Mechanics, Dynamics, and Aesthetics). MDA comprises a paradigm coined by Hunicke, LeBlanc and Zubek [18], where: (1) mechanics means game formal rules, their enforcement mechanisms, data representation and algorithms embedded within game components; (2) dynamics describes the run-time behaviour of the mechanics (i. e., interactions between mechanics and the player's input); (3) aesthetics refers to desirable emotional responses evoked in players by the dynamics like excitement, frustration or motivational intensity.

6. Artificial Intelligence (AI)—genetic algorithms, artificial neural networks, and constraint satisfaction and planning based on using PDDL, i.e., Planning Domain Definition Language [19].

When speaking of methods for procedurally generated game content, it is worth to distinguish approaches based on pure imperativeness of content generation from others applying declarative semantic methods and models for specification of individual problems. Imperative methods for procedural content generation create virtual worlds on the basis of structured geometric models of shapes, textures, and orientation concerning visual representation of objects in these worlds. On the other hand, semantic methods for procedural content generation make use of declarations of type, role, multiplicity, and relationships of the objects (entities). For example, the semantic model of a tree presented in [5] provides information about the attributes of that tree such as age, soil preferences, and its in-forest relationships.

2.2. Typology of PCG use in games. A typology[2] of ways of using PCG was proposed in [7] where "any particular example of PCG can be placed closer to one or the other extreme". This continuum of use of PCG was designed for clarifying especially the role of search-based PCG. Next, Shaker et al. [21] presented a modified version of the same typology, which includes five non-orthogonal distinctions:

- Online versus offline generation (we call it **generation mode**)—game content may be generated statically (offline, i.e., before running the game) or dynamically (i.e., at runtime during playing the game). For example, the interior layout of given room can be generated offline before the game is shipped, or on-the-fly (online for Web-based games) at the moment of entering the room.
- Necessary versus optional content (we call it **necessity**)—generated content can be really necessary, e.g., for answering a generated question in order to open a door or continue traversing a maze [22]; other objects

[2] We prefer to use the term "typology" instead of "taxonomy", because its classification categories "are neither exhaustive nor mutually exclusive" and "are descriptive rather than explanatory or predictive" [20].

not directly related to the gameplay (i.e., the player can omit considering them) can be generated optionally. It is important to note that necessary generation should be always correct for the game.

- Random seeds versus parameter vectors (called **degree of parameterization** [21])—while a random seeds algorithm generates a random number as input for the content generator, parameterized algorithms might receive one or several multidimensional input vectors. The level of granularity of game designer control over PCG [5] depends on the phase in the modelling process.
- Stochastic versus deterministic generation (we call it **determinism**)—in contrast with stochastic generation, deterministic algorithms always generate the same content given the same input parameters.
- Constructive versus generate-and-test algorithms (we call it **constructiveness**)—while constructive algorithms generate correct content once, generate-and-test approaches like genetic algorithms should test and prove the correctness of created content according to some criteria; in case of failure the candidate content is discarded, and new content is generated and tested again.

In order to supplement the continuum defined in [7] and [21] up to a general typology of PCG use considering not only search-based PCG, we add to it several other categories (note that not all of them are mutually exclusive), as follows:

- **Multiplicity** (generated content having single or multiple instances)—PCG is still predominantly applied to single player games considering an individual player's experiences [23]. The author stresses the fact that when used for multiplayer game design (e. g., in *Civilization IV*), PCG usually creates a single instance content, which is the same for all the players. Only in a few multiplayer games (such as *Galactic Arms Race*) is PCG used to create multi-instance content at runtime, i.e., unique content for each player visiting the same virtual space.
- **Player modelling** (non-personalized/non-adapted content versus personalization and adaptation of PCG to an individual player)—when

game designers strive to achieve unique playing experiences, gaming content should be generated in real-time and tailored to the expectations, needs and emotions of each player. Yannakakis and Togelius [13] proposed experience-driven procedural content generation by means of introducing a framework for PCG driven by computational models of user experience. In the scope of the ADAPTIMES[3] (ADAPTIve player-centric serious video gaMES) project, Bontchev [22] proposed style-based content selection in an educational game for learning strategic management, where game tasks and quests were dynamically selected according to playing style recognized within another video game played beforehand.

- **Player control** (whether and how the player can control the PCG process)—all content generators do not necessarily require any player control over the generation process. In cases of adapted content generation such as experience-driven procedural content generation [13] and style-based adaptation of game content [22], the player has implicit control over the generation process through the affective feedback loop of content creation [24]. On the other hand, he/she could start intentionally expressing a given playing experience (e.g., specific emotional input for the generator) in order to change the generated content, i.e., he/she might start controlling it in an indirect way via biofeedback. In cases of personalized content generation like stories branching according to player choices [12], the player is supposed to have direct control over the generation process.
- **Game industry**: content generated for entertainment games (called also games for fun) or for serious (or so-called applied) games—the differentiation is important because of different specifics of serious games compared to entertainment games such as skepticism of government-funded institutions like schools or military organizations, and lack of a "particularly large library of finished games", "large-scale statistical success", and "systematic improvement in this industry" [25]. Because of

[3] `http://adaptimes.eu/`

many factors (discussed later in the article), not all the types of methods for PCG can be applied to content generation for serious games.

- **Game genre**—specific types of generated content and generation methods are applied to different game genres such as platformers, puzzles, racing games, strategy games, and many others (refer to [26] for a description of game genres). Comparisons of commercial games of various genres applying generated content are given in [11] and [6]. The importance of the generated content depends strongly on the design of a given game and its storyline—for example, content with low quality can make a game less credible and realistic if one of the chief objectives is visual realism [28].
- **Derivation** (content built in a game or derived from a game)—while built-in content is used directly in the game and, therefore, applies to the bottom five levels of content type shown in Fig. 1, content drawn from the game is represented by the uppermost level of the pyramid (including also social media posts about concrete video games).

The extended version of the typology presented above provides a systematic basis for a comparison of the use of content generation in various genres and types of games. The classification can be visualized by grouping of the categories as shown in Fig. 2, as follows:

A. Content nature:
   - Multiplicity—single instances versus multiple instances.
   - Necessity—necessary versus optional content.
   - Derivation—built-in versus derived content.

B. Generation process:
   - Generation mode—online versus offline generation.
   - Degree of parameterization—random seeds versus parameter vectors.
   - Determinism—stochastic versus deterministic generation.
   - Constructiveness—constructive versus generate-and-test algorithms.

- Player modelling—non-personalized versus personalized generation.
- Player control—controlled versus non-controlled content.

C. Game dependence:

- Game industry—entertainment versus serious games.
- Game genre—dependence on the specific genre of a game.

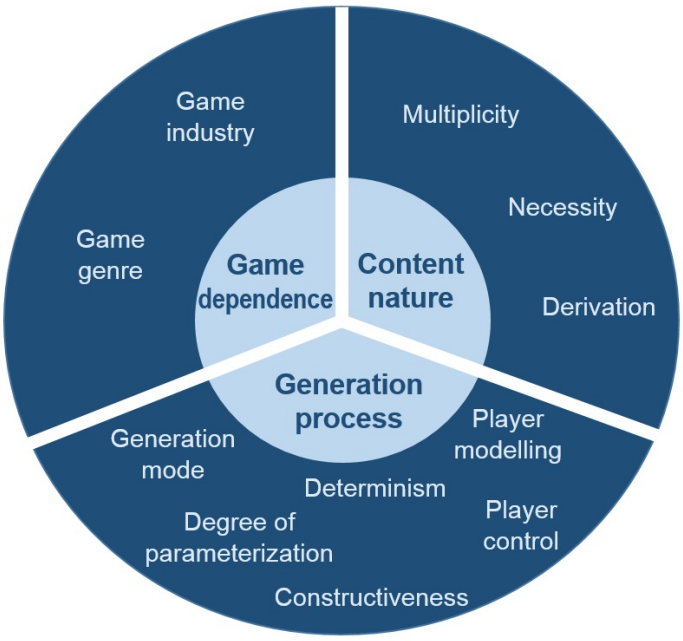

Fig. 2. Typology of use of procedurally generated game content

3. Procedural content generation in games. This section presents selected examples of procedural content generation for entertainment or serious games applied for education and training. Its objective is not to provide a detailed survey on PCG such as [11] but rather to outline specific approaches to procedural content generation and to stress their importance regarding type, methods, and use of procedurally generated game content. We have identified three promising directions of research and practical outcomes in the modern development of PCG based on applying personalization and adaptation, descriptive languages, and semantic specifications.

3.1. Personalized procedural content generation. Yannakakis and Togelius [13] outlined two challenges of personalized PCG for video games consisting in effective player modelling (presenting the emotional and cognitive experience of individual players) and efficient measuring of the quality of generated content in order to optimize player experience. They proposed a basic framework of experience-driven PCG, which consists of four components linked successively: Player Experience Modeling (PEM), assessor of content quality, content representation, and content generator. The PEM models player experience as a function of game content and players' cognitive, affective and style-based responses by means of subjective (self-report), objective (model-based like arousal-valence dimensions of emotions, or model-free like annotations of facial expressions), gameplay-based, or hybrid approaches. Next, the quality of the generated content is evaluated according to the player's experience in a direct, simulation-based, or interactive way. While the direct way maps specific content features to content quality using theory-driven or data-driven functions, the simulation approach involves an AI agent (static or dynamic) playing the game with content under evaluation and, on the other hand, the interactive way relies explicitly or implicitly on player interactions. After the assessment, content should be represented directly or indirectly in a form suitable for optimal efficacy, performance, and robustness of the generation process. Finally, it is sent to the content generator [28]. Based on various examples, Yannakakis and Togelius [13] conclude that "the quantification of player experience and the assessment of content quality based on a computational model of player experience" are the main challenges of the experience-driven PCG.

A further development of the ideas of experience-driven PCG is presented by Roberts and Chen [29], who consider an approach of learning-based PCG. They address three main problems of content quality in respect of optimised player experiences, namely: (1) how to avoid unacceptable content and how to categorize the content in the content vector space, (2) how to exploit potentially unreliable information acquired from the public (including player type/style), and (3) how to deal with player's preference changing during playing a game (so called concept-drift). The learning-based approach to PCG tries to learn from the developers about the content space and from

the beta-testers during public tests in order to gain knowledge about player behaviour. Thus, when the target players play the game, it can do player-centric online content adaptation resulting in minimized interruptions to player experience. For this purpose, the experience-driven PCG comprises three stages and submodels that are trained during each stage, as presented in Fig. 3. The models of Initial Content Quality (ICQ) and of Content Categorization (CC) are trained during the development stage by game developers for addressing problem (1), while the models of Generic Player Experience (GPE) and Play-log Driven Categorization (PDC) are trained during the public tests by beta testers for addressing problem (2). Finally, the model of Individual Preference (IP) is applied at the adaptation stage by monitoring their player log files, addressing problem (3).

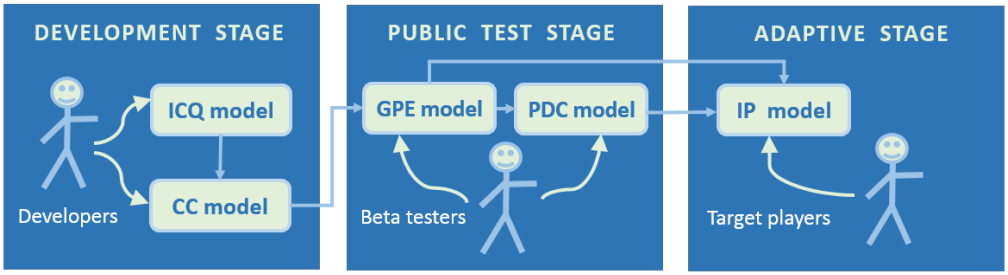

Fig. 3. Learning-based procedure content generation framework (upon [29])

To collect experimental data for the learning-based PCG system, Roberts and Chen [29] chose the popular first-person shooter (FPS) game *Quake* for the public test via active learning distributed equally into five categories of difficulty. They reported promising results about generating content appealing to target game players.

As of now, most experiments considering personalized content generation are run mainly for entertainment games. On the other hand, serious games also need generated learning content that can be tailored to the needs and preferences of a given group of learners [30]. An approach to the generation of content adapted to learning and playing styles was proposed in [22] making use of automatic generation of maze video games for training. The work addresses the lack of free software platforms allowing easy creation of simple but attractive customizable educational video games by professionals in

areas other than information technology. With this purpose, in the scope of the ADAPTIMES project there was developed a software design tool for formal description, customization, generation and management of 3D video labyrinths. The tool was based on Brainstorm's eStudio[4] platform and was designed to support game-based learning in various learning domains. Teachers can design 3D video mazes with a desirable degree of connectivity through textual or graphic labyrinth editors as well as customize the maze nodes and the transitions between them (Fig. 4).

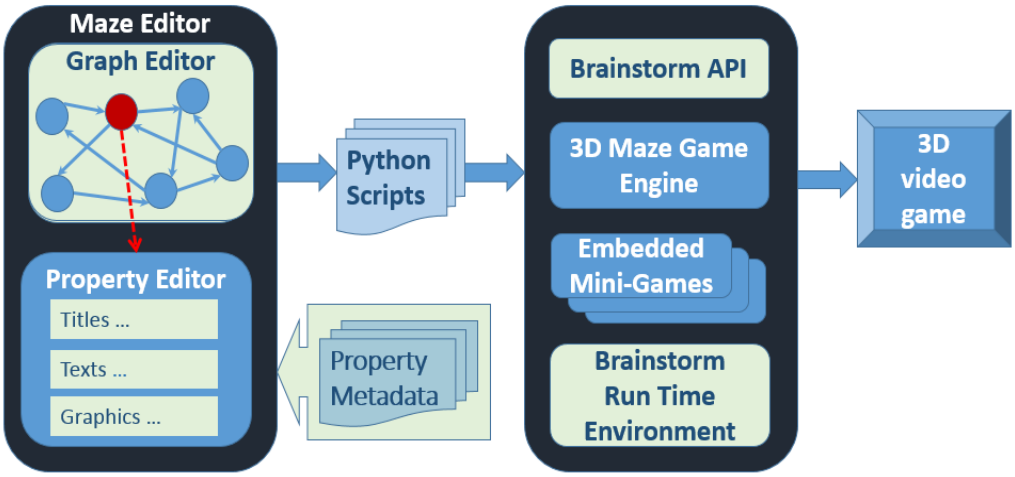

Fig. 4. Software architecture of a maze design platform

The property editor uses either a predefined fixed set of properties for each node (a room with specific disposition of learning tables with multimedia content; textures, shapes, and colors of walls; sound and audio arrangements; didactic and hidden 3D objects; test questions for opening doors, etc.), or properties set by metadata. The formal descriptions, along with all the data and learning content of a game, serve for the generation of a Python script, which is executed by the game management platform in the Brainstorm graphical environment. Besides general rooms connected by tunnels with quizzes of various types, the generated 3D mazes can also include two mini-games: a 3D Quiz with animated questions for learner assessment after passing

[4] http://www.brainstorm.es/products/estudio/

through a part of the maze and a 3D Zoom mini-game for ordering a stack of scattered images. The learning content of both the 3D Quiz and the 3D Zoom game was generated along with the labyrinth using the formal game description. The maze design platform was applied in practical experiments for generation of maze games for entrepreneurship training with adaptation based on playing/learning styles, where styles were recognized implicitly and dynamically through playing an action-adventure game [31]. The playing style-based adaptation was used to adapt game tasks generated by the platform for each room and each tunnel of the maze for entrepreneurship training (Fig. 5).

Fig. 5. A style-adapted game task in the Mission room of a maze for learning strategic management

3.2. Content generation using game descriptive languages. Game descriptive languages (GDLs) constitute a challenging research area. They allow definition of games understood by computers for a specific range of games. The Stanford GDL [32] is a declarative language using first order logic defined for general game playing covering turn-based, competitive games like chess and backgammon. Browne & Maire [33] proposed Ludi as a formal system for playing, measuring and synthesizing combinatorial games within the

scope of GDL. Another simple description language similar to GDL is PuzzleScript proposed by Stephen Lavelle [34] for easy prototyping of turn-based, keyboard-controlled puzzle games. The General Video Game Playing (GVGP) proposed in [35] extends the general game playing with other types of games like arcade games and is conceived for AI agents playing unknown video games by receiving the current state of the game and actions applicable to it. A Video Game Description Language (VGDL) was designed especially for the GVGP [36] for supporting the core mechanics and behavior of classical 2D video games including PCG and automatic game generation. VGDL descriptions consist of a map, objects, player definitions, avatars, physics, events, and rules. Tom Schaul [37] designed Python VGDL (PyVGDL) as a simple high-level GDL and applied it for specification of many popular 2D video games such as *Space Invaders, Lunar Lander, Pac-Man, Sokoban, Legends of Zelda*, and others. Perez-Liebana et al. [38] ported the implementation of PyVGDL to Java and thus created the GVG-AI framework able to load games and levels described in VGDL and to expose the formal game model to agent controllers.

The descriptive languages outlined above have been used together with other approaches for specification and generation of game content and various video games. One of the most promising applications of both the VGDL and GVG-AI framework is for automatic generation of game levels. Khalifa et al. [39] proposed a GVG-LG framework for level generators for games specified in VGDL and playable by some AI player, which "builds any required number of different levels for that game which are enjoyable for humans to play". Together with the GVG-LG framework, they designed three sample level generators: for creating sprites at random empty positions, for generation of avatars, solid, harmful, collectible, and other sprites using a constructive approach, and a search-based level generator based on a generic algorithm. A pilot study compared the levels produced by these generators through testing with human players and revealed that humans were unable to distinguish between the constructive and random generators but prefer the search-based generator [39].

An arguing approach to evaluation and automatic generation of general video games using a description language is suggested in [40]. The authors

developed a system that automatically finds out solutions for various video games described in PuzzleScript and having different game mechanics, rules, level designs, and winning conditions. They applied a set of level state heuristics for estimating the proximity of a given game level to the solution and, as well, a set of ruleset heuristics for defining the game's mechanics and assessing its playability. Next, they generated playable rulesets from scratch using an evolutionary approach and thus proved that PuzzleScript can be used for general design evaluation and generation.

Other approaches to applying a game descriptive language for mechanic generation are based on PDDL [19], for declarative description of a game state and transitions. Zook and Riedl [41] proposed a generate-and-test game design process via mechanic generation. By means of a constraint solver, they generate mechanics meeting (1) given required or optimized formal design conditions and (2) adaptation requirements specifying additional playability or design requirements. Next, they test whether these mechanics meet the playability requirements. The game domain comprises a state model described in PDDL, and a PDDL transition model allowing simulation and planning checks. The approach was used for the representation of simple role-playing and platformer games.

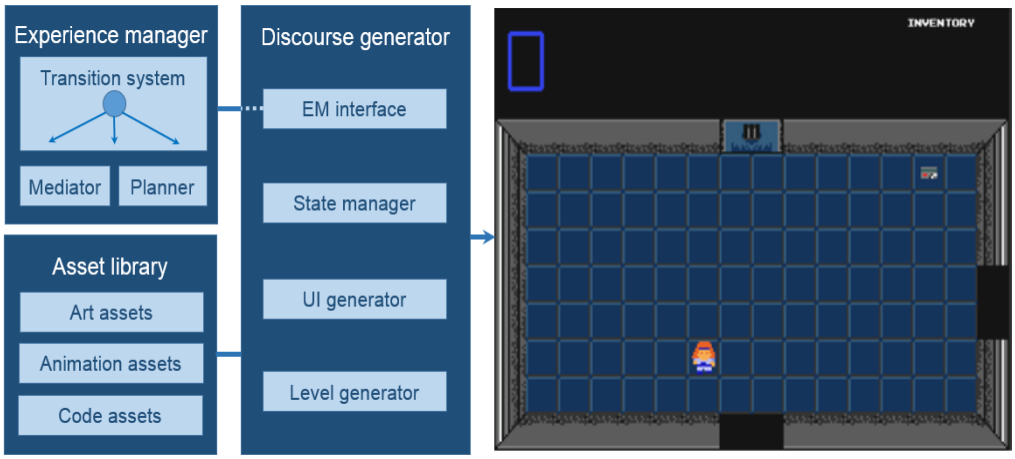

Fig. 6. The GME system pipeline from declarative representation and game assets to a game world, built upon [8]

Automated generation of gameplay including world mechanics, assets, states, virtual agents, and plot events was proposed by [8] on the basis of declarative world representations. The authors conceived a platform called General Mediation Engine (GME), which applies a PCG pipeline on top of an experience management framework. The PCG pipeline receives, from an experience manager, atomic formulae representing world states, preconditions and effects (i.e., post-conditions). Next, it applies them for producing a state-transition system of an interactive game, which serves for manipulating all the assets based on a given state. The construction of state transition systems is based on declarative PDDL descriptions of initial and goal states and conditioned action operators to be performed by agents for transforming world states. This transition system models the game world and is used by an experienced manager and discourse generator. The experience manager manipulates the game world by maintaining a desired experience plan including NPC character actions and monitoring the transition system. It consists of a state transition system, a planner, and a mediator used for maintenance of world states, plan updates and execution of NPC actions. The discourse generator generates a playable game world using the state-transition system and the asset library (Fig. 6) and, as well, creates and maintains world objects like the player, NPCs, and game items. It consists of several components: an experience management (EM) interface initializing the discourse generation system and receiving commands from the mediator; a game state manager responsible for maintenance of game assets based on the current state; a user interface (UI) generator creating and configuring the interface, game camera, and world layout; and a level generator responsible for building a high-level physical configuration graph by using locations and connections specified by PDDL.

GME has been implemented as a Unity General Mediation Engine. This game engine has been used to create a 2D sneaking game generated and maintained in a declarative way [8].

3.3. Content generation using semantic world representations. Semantic approaches to content generation for games started in the last twenty years with applying semantic information to techniques for automatic

generation of terrain erosion, distribution of vegetation, road networks, city maps, and interiors of buildings [42]. For example, procedural techniques were applied for generation of consistent buildings [43]. The first comprehensive declarative semantic model of a game world useful for PCG was proposed by Smelik [5], who defined four levels of abstraction of modelled game objects: geometric objects level (including 3D geometric meshes, textures, etc.), semantic objects level (represented by a set of generated objects and their features), structure level (including feature extent and structural objects), and specification level (outline shape and semantic attributes). Smelik developed a SketchaWorld prototype incorporating various semantic features of different objects situated on five predefined layers of the virtual world model: urban, road, vegetation, water, and landscape layer. For a particular semantic object, a combination of procedural methods was applied for generating all the comprised elements. The generated elements were combined with instances of semantic definitions into a semantic model of the object (Fig. 7). A semantic consistency moderator was added for ensuring the maintenance of the consistency of the semantic model of the content. This process of generic procedural generation assisted by semantic specification was applied in the SketchaWorld prototype for creating content for in-house developed simulators for training military personnel and for the *Levee Patroller* serious game for training levee inspectors [5].

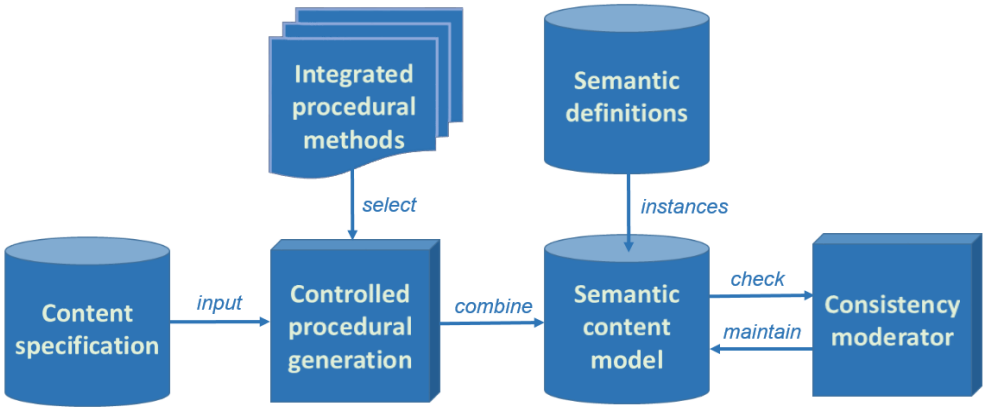

Fig. 7. A generic process declarative procedural generation (after [5])

A semantic-based framework for enabling procedurally generated game content by the player's behavior and gameplay experience was suggested by [44]. They applied reusable gameplay semantics defined by the designer for matching the content generation with the player's behavior and experience. Knowledge about gameplay experiences, player behavior features and involved game actors was imported from a semantic library and used to control and constrain automatic content generation. Knowledge containers encoded valid combinations between semantic entities (e.g., a car ramp) and player features such as preferences, skills, style, and experiences. On the other hand, the player model was observed for retrieval of dynamic values of player features used "to synthesize such player-matching content into a meaningful game world (segment)". The authors integrated the semantic-based framework into an existing 3D car game (*Stunt Playground*) and used it, together with a specific model of player behavior and experience, in order to generate player-matching game worlds at gaming time.

Educational games are very appropriate for personalized and adaptive e-learning. Here, semantic structuring and organization of learning content facilitate greatly automatic content extraction. Bontchev [10] proposed a design and delivery workflow (Fig. 8), along with a software framework for the construction of simple single-user word and logic games based on the automatic extraction of the semantic organization of educational content provided in the form of learning objects. Several word games such as hangman, anagram, memory and association games (with optional use of intelligent agents) have been created with an ability to generate learning content by personalized extraction from courseware organized in an ontology. The course instructor managed the personalization and adaptation of game content with respect to agents' behavior. The games were applied for adaptive e-learning in XML technologies, with adaptation based on the learning style of the individual learner. Practical experiments conducted by using the ADOPTA platform [30] proved benefits resulting from a personalisable and adaptable instantiation of puzzle games with didactic content generated from semantically structured courseware, whereby games were automatically inserted into a storyboard graph by means of adaptation rules addressing both the learning style and the results of the learner.

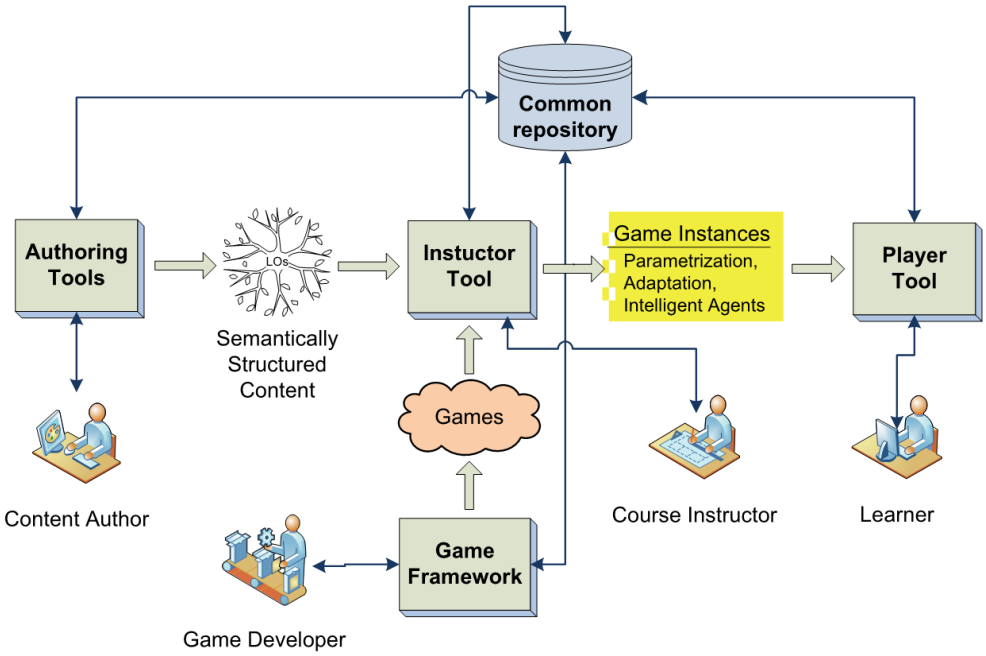

Fig. 8. Workflow of educational game creation and delivery [10]

4. Discussion. The section offers a discussion of the selected studies of personalized, descriptive, and semantic-based procedural generation of content for video games presented above. Table 1 presents a comparison of all the ten approaches ordered as outlined in section 3. The table compares seven games for fun and three serious games according to the classifications of content, methods, and use of PCG. The content type varies from indoor/outdoor objects to levels, rules, and courseware, while the majority of the generation methods use declarative languages. Only the last three studies use semantic modelling for generation of content. All approaches based on player modelling generate multiple contents tailored according to the preferences, skills, style, or experience of an individual player, with indirect control. They apply built-in and necessary content generated either online of offline, using high-level parameterization, in a deterministic or stochastic way, mostly for 2D games. While some approaches follow a deterministic generation process, others make use of stochastic or combined processes.

Table 1. A comparison of selected approaches for personalized,

Study		[13, 28]	[29]
Content type		levels	monster and ammunitions
Method of PCG		neural networks	OBLIGE random level generator[5]
Content nature	Multiplicity	multiple	multiple
	Necessity	necessary	necessary
	Derivation	built-in	built-in
Generation process	Mode	online	online
	Parameterization	number, size and place of gaps and switching	skill level, N monsters, health packs, weapon, monster types
	Determinism	stochastic	stochastic
	Constructiveness	generate-and-test	generate-and-test
	Player modelling	subjective and gameplay player models	content adaptation according player experience
	Player control	implicit	learning from beta testers
Game dependence	Industry	games for fun	games for fun
	Genre	2D platformer	3D FPS

[5] `http://oblige.sourceforge.net/`

descriptive, and semantic-based procedural content generation

[22, 31]	[39]	[40]
indoor, courseware	levels	levels, rules
declarative	declarative (VGDL)	declarative (Puzzle-Script)
multiple	single	single
necessary	necessary	necessary
built-in	built-in	built-in
offline	offline	offline
no	priority and category of sprites	level state and ruleset heuristics
deterministic	deterministic/stochastic	stochastic
constructive	random, constructive, generate-and-test	generate-and-test
playing style	no	no
implicit	no	no
serious games	games for fun	games for fun
3D learning mazes	2D platformer	2D puzzle

Table 1. A comparison of selected approaches for personalized,

Study		[41]	[8]
Content type		avatar-centric mechanics, levels	world mechanics, assets, states, virtual agents, plot events
Method of PCG		declarative (PDDL)	declarative (PDDL)
Content nature	Multiplicity	single	single
	Necessity	necessary	necessary
	Derivation	built-in	built-in
Generation process	Mode	offline	online
	Parameterization	required / optimized design requirements; adaptation requirements	game state description
	Determinism	stochastic	deterministic
	Constructiveness	generate-and-test	constructive
	Player modelling	no	no
	Player control	no	no
Game dependence	Industry	games for fun	games for fun
	Genre	2D role-playing, platformer	2D puzzle

descriptive, and semantic-based procedural content generation *(continued)*

[5, 6]	[44]	[10]
urban objects, roads, vegetation, waters, and landscapes	stunt arenas	courseware
declarative semantic modelling	declarative semantic modelling	declarative semantic modelling
single	multiple	multiple
necessary / optional	necessary	necessary
built-in	built-in	built-in
offline	online	offline
semantic object attributes and relationships	semantic gameplay descriptions, player features	playing / learning style parameters
deterministic / stochastic	stochastic	deterministic
constructive / generate-and-test	constructive	constructive
no	preferences, skills, style, experience	playing / learning style
no	implicit	implicit
serious games	games for fun	serious games
3D games for military training	3D car racing	2D word puzzles

The comparison of the features of the approaches to PCG for video games presented in Table 1 provides a base for further discussion about the current state and the trends in the generation of game content. Here we will outline some considerations about them, as follows:

- The role of personalized and adapted content generation appears to be more and more important for the creation of appealing, engaging and immersive games. At Table 1 reveals, player modelling approaches of PCG can generate game content tailored according to player preferences, skills, experience, or learning/playing style. Tailored and surprising game environments created by such PCG methods increase game aesthetic properties such as challenge, discovery, and fellowship [23]. On the other hand, adapted PCG encourages communion and empathy among players, e.g., while commenting generated maps in *Civilization IV* (Firaxis Games, 2005). As well, the dynamic adaptation resolves the problem of changes in player's preferences over time (so called concept-drift) [29].
- There are many approaches to PCG allowing the player to control the generation process—all studies using player modelling for a personalized and adapted content generation provide player control over the generation. This control is performed in an implicit form thanks to the affective feedback mechanism of content creation [24].
- For the majority of the presented approaches to PCG, game developers have parameterized control over the generation process such as various levels of skills required to deal with the generated content (i.e., content difficulty), rulesets, type and state of health, ammunitions, monsters and other NPC, and so on. This control over the generations tends to use an advanced set of parameters allowing fine tuning and personalization of automatically created content according to various design purposes.
- Both constructive and generate-and-test methods for PCG are applied to offline and online content generation, although generate-and-test methods such as genetic programming cause problems with the genotype-to-phenotype mapping and require a large amount of content to be assessed [29].

- Applications of methods for PCG based on languages such as VGDL, PuzzleScript, and PDDL appear to be more popular. Such methods use descriptive languages for declarative representations of game levels, state, transitions, rules, mazes, etc. Some of them apply advanced semantic modelling in order to describe categorization of content objects and their properties and interconnections, in order to generate credible ecosystems for video game worlds [5, 6].

When discussing automatic content creation, we have to make a distinction between content generated for entertainment games and for serious (i.e., applied) games. Since the very beginning of their application, PCG methods are applied mainly for games for fun. As Table 1 confirms, the experiments of personalized content generation consider mainly entertainment games. Besides game content such as mazes, land shaft and levels, serious games also need automatically generated learning content that is able to be tailored to the needs and preferences of a given group of learners [30]. Generation of personalized and adapted game content can help to overcome three major types of obstacles hampering the massive penetration of serious games [45]:

- Pragmatic barriers consisting in the relatively high cost and long production time of a serious game, which establishes a baseline hard to overcome. Other pragmatic issues causing gaps between expectations and reality are time-scales, sustainability of the game-based training, and outsourcing of the development serious games.
- Performance barriers existing because of the fact that the overall quality of any serious game is lower than that of the modern entertainment games—serious games are not perceived as so funny and engaging as entertainment games. This perception raises an initial adoption barrier that shatters the use of serious games.
- Pedagogical barriers—because of the fact that it is hard to ensure high learnability [46] of a serious game and the lack of reliability of producing expected elements of valuable learning and relevant training (providing we have defined exactly what learning is required). Therefore, it is crucial

to understand what elements of learnability are to be measured for an adequate assessment of the game from the didactic point of view.

Therefore, effective generation of both game content and learning courseware appears to be very important and promising with respect to all three major barriers to the large-scale adoption of serious games [45].

5. Conclusions. Since their adoption, methods for procedurally generated game content appear to be more and more popular due to the accelerating growth of the game market and the restrictions of manual content creation. Instead of relying on the efforts of an army of human designers, PCG follows well-defined procedures for automatic creation of game content [11, 13]. While PCG started with generating non-interactable content like stars, trees, lakes, roads, mazes, etc., modern approaches tend to drive innovative game design by automatic creation of game content possessing a level of interactivity [23]. Interactive content may include game rules and dynamics, weapons and shooting targets, game levels, learning courseware, and other types of game objects. For a high interactivity of generated game content, an advanced and versatile parameterization of the generation process is necessary. Next to content interactivity, personalization and adaptation based on advanced player modelling, use of game descriptive languages like VGDL, PuzzleScript, and PDDL and, as well, semantic world representations appear to be important research directions of modern PCG, as explained in sections 3 and 4. These important features of modern PCG will fuel the creation of novel and attractive games and will foster player immersion, engagement and overall game playability [31].

The future of the procedural generation of games is determined by the ever-growing gap between the demand for fresh game content and player-centric game customization and, from the other side, both the rate and the price of the current practices of manual content production by game designers [11]. In order to provide new and customized game content at an affordable price and in limited time, novel methods for PCG have to be applied in both entertainment and serious games. All of them should allow game designers and artists to control the (semi-)automatic design process, by providing mechanisms for fine-tuning and adjusting the generation process parameters.

Finally, the modern PCG methods are very promising with respect to resolving major obstacles in the large-scale adoption of serious games. With regard to modern trends in adoption of game-based learning, teachers and instructors cannot continue relying only on single custom educational video games embedding learning content from a specific domain. PCG should be applied for creation of simple, cheap, personalisable and extensible software platforms for rapid and easy construction of didactic games on top of semantically structured course content in any learning domain, in order to facilitate massive penetration of educational video games.

REFERENCES

[1] Lee R. S. Home videogame platforms. The Oxford Handbook of the Digital Economy, 2012, 83–107.

[2] ESA. Essential facts about the computer and video game industry. Electronic Software Association, 2015.

[3] Granic I., A. Lobel, R. C. Engels. The benefits of playing video games. *American Psychologist*, **69** (2014), No 1, 66.

[4] Aldrich C. The complete guide to simulations and serious games: How the most valuable content will be created in the age beyond Gutenberg to Google. John Wiley & Sons, 2009, 576.

[5] Smelik R. M. A Declarative Approach to Procedural Generation of Virtual Worlds. PhD Thesis, Technische Universiteit Delft, Netherlands, 2011.

[6] Smelik R. M., T. Tutenel, R. Bidarra, B. Benes. A survey on procedural modelling for virtual worlds. *Computer Graphics Forum*, **33** (2014), No 6, 31–50.

[7] Togelius J., G. Yannakakis, K. Stanley, C. Browne. Search-based procedural content generation. Applications of Evolutionary Computation, 2010, 141–150.

[8] Robertson J., R. M. Young. Automated gameplay generation from declarative world representations. In: Proceedings of 11th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2015, 72–78.

[9] Smith, A. M., E. Andersen, M. Mateas, Z. Popovic. A Case Study of Expressively Constrainable Level Design Automation Tools for a Puzzle Game. In: Proceedings of the International Conference on the Foundations of Digital Games (FDG'12), 2012.

[10] Bontchev B. A Framework for Educational Word Games. In: Proceedings of the International Conference on Intelligent Computational Systems (ICICS'2012), 978–981.

[11] Hendrikx M., S. Meijer, J. Van Der Velden, A. Iosup. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, **9** (2013), No 1.

[12] Riedl M., V. Bulitko. Interactive Narrative: An Intelligent Systems Approach. *AI Magazine*, **34** (2013), No 1, 67–77.

[13] Yannakakis G. N., J. Togelius. Experience-driven procedural content generation. *IEEE Transactions on Affective Computing*, **2** (2011), No 3, 147–161.

[14] Ponsen M., H. Munoz-Avila, P. Spronck, D. W. Aha. Automatically generating game tactics through evolutionary learning. *AI Magazine*, **27** (2006), No 3, 75–84.

[15] Grey J., J. J. Bryson. Procedural quests: A focus for agent interaction in role-playing-games. In: Proceedings of the AISB 2011 Symposium: AI & Games, UK, 2011, 3–10.

[16] Hewlett W. R. Creating a Cognitive Agent in a Virtual World: Planning, Navigation, and Natural Language Generation. PhD thesis, University of California, Los Angeles, USA, 2013.

[17] Brathwaite B., I. Schreiber. Challenges for Game Designers. Charles River Media Inc., Rockland, MA, 2008.

[18] Hunicke R., M. LeBlanc, R. Zubek. MDA: A formal approach to game design and game research. In: Proceedings of AAAI Workshop on Challenges in Game Artificial Intelligence, 2004, 1–5.

[19] McDermott D., M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, D. Wilkins. PDDL—The Planning Domain Definition Language, Tech. Rep. CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, New Haven, 1998.

[20] Smith K. B. Typologies, taxonomies, and the benefits of policy classification. *Policy Studies Journal*, **30** (2002), No 3, 379–395.

[21] Shaker N., J. Togelius, M. J. Nelson. Procedural Content Generation in Games: A Textbook and an Overview of Current Research. Springer, 2015.

[22] Bontchev B. Video games for teaching entrepreneurship. *Avtomatika i Informatika*, **3** (2015), 23–28.

[23] Smith G. The future of procedural content generation in games. In: Proceedings of the Experimental AI in Games Workshop, 2014, 53–57.

[24] Gilleade K., A. Dix, J. Allanson. Affective videogames and modes of affective gaming: assist me, challenge me, emote me. In: DiGRA 2005: Changing Views: Worlds in Play, 2005.

[25] Terdiman D. What's wrong with serious games? 2006. https://www.cnet.com/news/whats-wrong-with-serious-games/, 25 May 2017.

[26] Adams E. Fundamentals of Game Design. Pearson Education Inc., Third Edition, 2014.

[27] Amato A. Procedural Content Generation in the Game Industry. In: Game Dynamics. Springer, 2017, 15–25.

[28] Pedersen C., J. Togelius, G. N. Yannakakis. Modeling Player Experience for Content Creation. *IEEE Transactions on Computational Intelligence and AI in Games*, **2** (2010), No 1, 54–67.

[29] Roberts J., K. Chen. Learning-based procedural content generation. *IEEE Transactions on Computational Intelligence and AI in Games*, **7** (2015), No 1, 88–101.

[30] Vassileva D. Adaptive e-learning content design and delivery based on learning styles and knowledge level. *Serdica Journal of Computing*, **6** (2012), No 2, 207–252.

[31] ADAPTIMES. Deliverable D9: Demonstration of the final field trial, Version 1.0. 2016. http://adaptimes.eu/deliverables.html, 25 May 2017.

[32] Love N., T. Hinrichs, D. Haley, E. Schkufza, M. Genesereth. General game playing: Game description language specification, Technical Report LG-2006-01, Stanford Logic Group, 2006.

[33] Browne C., F. Maire. Evolutionary game design. *IEEE Transactions on Computational Intelligence and AI in Games*, **2** (2010), No 1, 1–16.

[34] Lavelle S. PuzzleScript. 2013. http://www.puzzlescript.net/, 25 May 2017.

[35] Levine J., C. B. Congdon, M. Ebner, G. Kendall, S. M. Lucas, R. Miikkulainen, T. Schaul, T. Thompson. General video game playing. *Artificial and Computational Intelligence in Games*, **6** (2013), 77–83.

[36] Ebner M., J. Levine, S. M. Lucas, T. Schaul, T. Thompson, J. Togelius. Towards a video game description language. *Artificial and Computational Intelligence in Games*, **6** (2013), 85–100.

[37] Schaul T. A video game description language for model-based or interactive learning. In: Proc. IEEE Conference on Computational Intelligence in Games (CIG), 2013, 1–8.

[38] Perez-Liebana D., S. Samothrakis, J. Togelius, T. Schaul, S. M. Lucas, A. Couëtoux, T. Thompson. The 2014 general video game playing competition. *IEEE Transactions on Computational Intelligence and AI in Games*, **8** (2016), No 3, 229–243.

[39] Khalifa A., D. Perez-Liebana, S. M. Lucas, J. Togelius. General video game level generation. In: Proc. Conf. Genetic and Evolutionary Computation, ACM, 2016, 253–259.

[40] Lim C.-U, D. F. Harrell. An approach to general videogame evaluation and automatic generation using a description language. In: Proc. IEEE Conference on Computational Intelligence and Games (CIG), 2014, 286–293.

[41] Zook A., M. O. Riedl. Automatic Game Design via Mechanic Generation. In: Proc. Association for the Advancement of Artificial Intelligence, 2014, 530–537.

[42] Tutenel T., R. Bidarra, R. M. Smelik, K. J. de Kraker. The role of semantics in games and simulations. *Computers in Entertainment*, **6** (2008), No 4, 57:1–57:35.

[43] Tutenel T., R. M. Smelik, R. Lopes, K. L. de Kraker, R. Bidarra. Generating Consistent Buildings: a Semantic Approach for Integrating Procedural Techniques. *IEEE Transactions on Computational Intelligence and AI in Games*, **3** (2011), 274–288.

[44] Lopes R., T. Tutenel, R. Bidarra. Using gameplay semantics to procedurally generate player-matching game worlds. In: Proceedings of the 3rd workshop on Procedural Content Generation in Games, ACM, 2012, 3–10.

[45] Oliveira M. Major barriers in the large-scale adoption of serious games. Private discussion, March, 2014.

[46] Andersen E., E. O'Rourke, Y. E. Liu, R. Snider, J. Lowdermilk, D. Truong, S. Cooper, Z. Popovic. The impact of tutorials on games of varying complexity. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 2012, 59–68.

Boyan Bontchev
Faculty of Mathematics and Informatics
Sofia University "St. Kliment Ohridski"
1113 Sofia, Bulgaria
e-mail: bbontchev@fmi.uni-sofia.bg

Received June 13, 2017
Final Accepted June 26, 2017