Delft University of Technology

Orchestrating Game Generation

Liapis, Antonios; Yannakakis, Georgio N.; Nelson, Mark J.; Preuss, Mike; Bidarra, Rafael

DOI
10.1109/TG.2018.2870876

Publication date
2019

Document Version
Final published version

Published in
IEEE Transactions on Games

Citation (APA)
Liapis, A., Yannakakis, G. N., Nelson, M. J., Preuss, M., & Bidarra, R. (2019). Orchestrating Game Generation. *IEEE Transactions on Games*, *11*(1), 48-68. Article 8466898. https://doi.org/10.1109/TG.2018.2870876

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' – Taverne project

https://www.openaccess.nl/en/you-share-we-take-care

Orchestrating Game Generation

Antonios Liapis, Georgios N. Yannakakis, *Senior Member, IEEE*, Mark J. Nelson, Mike Preuss, and Rafael Bidarra

***Abstract*—The design process is often characterized by and realized through the iterative steps of evaluation and refinement. When the process is based on a single creative domain such as visual art or audio production, designers primarily take inspiration from work within their domain and refine it based on their own intuitions or feedback from an audience of experts from within the same domain. What happens, however, when the creative process involves more than one creative domain such as in a digital game? How should the different domains influence each other so that the final outcome achieves a harmonized and fruitful communication across domains? How can a computational process *orchestrate* the various computational creators of the corresponding domains so that the final game has the desired functional and aesthetic characteristics? To address these questions, this paper identifies game facet orchestration as the central challenge for artificial-intelligence-based game generation, discusses its dimensions, and reviews research in automated game generation that has aimed to tackle it. In particular, we identify the different creative facets of games, propose how orchestration can be facilitated in a top-down or bottom-up fashion, review indicative preliminary examples of orchestration, and conclude by discussing the open questions and challenges ahead.**

***Index Terms*—Artificial-intelligence (AI)-based game generation, computational creativity, orchestration, procedural content generation (PCG).**

I. Introduction

GAME design lies at the intersection of a multitude of creative domains, from art and music to rule systems and architecture. These domains influence each other, with flashy visuals reinforcing a fantasy narrative and creepy background sounds adding to the player's tension during gameplay. While the multifaceted nature of games is a great blessing for their aesthetic expressiveness and functional capacity, it is arguably a curse for algorithmic processes that attempt to automate the generation of games. It is one thing to be able to generate a good level, and another thing to be able to generate a level with appropriate sound effects, narrative, and game rules; the latter is several magnitudes more challenging than the former.

Manuscript received March 20, 2017; revised December 4, 2017 and June 18, 2018; accepted August 17, 2018. Date of publication September 17, 2018; date of current version March 15, 2019. *(Corresponding author: Antonios Liapis.)*

A. Liapis and G. N. Yannakakis are with the Institute of Digital Games, University of Malta, 2080 Msida, Malta (e-mail: antonios.liapis@um.edu.mt; yannakakis@itu.dk).

M. J. Nelson is with the MetaMakers Institute, Falmouth University, Falmouth TR11 4RH, U.K. (e-mail: mjn@anadrome.org).

M. Preuss is with the University of Münster, 48149 Münster, Germany (e-mail: mike.preuss@cs.tu-dortmund.de).

R. Bidarra is with the Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: R.Bidarra@tudelft.nl).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TG.2018.2870876

By decomposing games into creative facets (visuals, audio, narrative, rules, levels, and gameplay), we can simplify the problem of game generation and allow algorithms to treat each facet independently. Each game facet, such as a level or a ruleset, offers a controlled area for our exploration. Due to the evident benefits of simplicity and controllability, the focus of commercial games and academia on procedural content generation (PCG) has so far been on generating a *single* creative facet of a game (e.g., a level generator) [1]. In this paper, we argue that the dominant practice of designing a generator for a specific type of content has been detrimental to the grand vision of PCG: the generation of complete games. When designing a map generator for the strategy game *StarCraft* (Blizzard 1997), for instance, it is easy to completely ignore the remaining components of the game that contribute to the level generation per se. Components such as the cost and speed of units, the progression of both difficulty and storyline from one map to the next, or even the color palette of the map's tiles are overlooked. Even in generators of a broader scope and less specificity [2], certain gameplay patterns such as the need of balance in a multiplayer shooter game are presumed to come with the genre. When generating complete games, however, the computer should be able not only to generate all of those components, but also to reason whether generated content of one type (such as visuals) aligns well with generated content of a different type (such as game rules).

In this paper, we put a particular emphasis on the process we name *orchestration*, which refers to the *harmonization of the game generation process*. Evidently, orchestration is a necessary process when we consider the output of two or more content-type generators—such as visuals and audio—up to the generation of a complete game. To support our definition and argue for the importance of orchestration for computational game design, we use the music domain as our core metaphor throughout this paper. The orchestration process in music can take various forms that are inspiring for computational game design. Music orchestration often takes the form of a *composer* (i.e., an overseer of each instrument's output), who makes sure that musical instruments follow a designated pattern of rhythm, tempo, and melody as represented through notes and symbols in the composer's pentagram. The composer is ultimately responsible for the final outcome. On the other end of the spectrum, orchestration can take the form of *improvisation* or *jamming*, as in freeform jazz or the urban blues. While jamming, musicians try to adapt to the rhythmic and melodic patterns followed by the rest of the band; as a result, orchestration is a property that *emerges* from the interactions among musicians and the outputs of their instruments. The first orchestration paradigm can be defined as a *top-down*, composer-driven, process, whereas

2475-1502 © 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

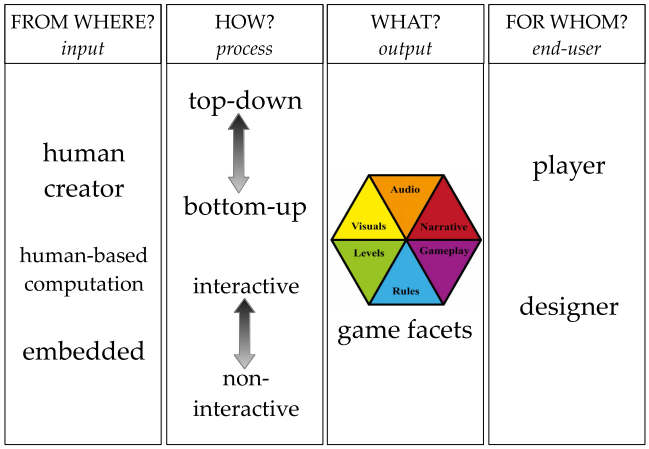


Fig. 1. Key questions of facet orchestration.

the latter paradigm can be viewed as a *bottom-up*, freeform, process. Various hybrids between the two paradigms are possible: for instance, a well-rehearsed and structured song with an improvisational guitar solo part. Fig. 1 depicts this granular relationship between top-down and bottom-up via a gradient-colored arrow. Top-down and bottom-up processes for artificial intelligence (AI) coordination have been researched extensively outside of games and will be used as grounding for our proposed orchestration frameworks.

Music orchestration appears to be an ideal metaphor for describing the orchestration process of game design. During game design, the various types of game content can be coordinated, on one end, by a game designer or game director: this is a typical top-down process followed by, e.g., AAA studios. On the other extreme, game design can be coordinated internally by the different game content designers, e.g., during a game jam. In this paper, we argue that AI-based game generation can benefit from following the orchestration patterns met in human game design (and in music creation). In a similar fashion, computational creators could be orchestrated by an agent that acts as the game director or, instead, self-organize and internally harmonize their creative processes. Hybrids between the two extremes (top-down versus bottom-up) are also relevant for AI-based game generation.

By now, it should be obvious that orchestration for game generation is both a *hard* and an *interesting* challenge [3], [4]. It is a hard problem as it must not only coordinate and initialize multiple dissimilar generators, but also ensure that the result is coherent among all game facets. Coherence evaluation is a major challenge—even in single creative domains such as art—and to solve it fully would require human-level intelligence (making it an AI-complete problem). However, this paper proposes several shortcuts based on the typology of Fig. 1 for ensuring coherence through a hierarchical generation with human-authored associations in a top-down process, through semantic labels, which can be linked together through human-based computation (e.g., via Glunet [5]), through machine-learned patterns across facets from online content such as gameplay videos and reviews, or through human intervention during orchestration. It is an interesting problem as it potentially realizes the grand vision of generative systems in games: the generation of complete games [3], [4]. Given the recent progress of machine and deep learning as a generative process, the ever-expanding repository of semantically annotated open data, and the growing number of academic embryos in this direction, we argue that this is the ideal time to actively discuss orchestration and its challenges in greater depth.

This paper places the first stepping stone toward game orchestration by questioning *what* can be orchestrated and *how* it can be done, *where* the process is inspired or initiated from, and, finally, *for whom* the final outcome of orchestration is. Answering these questions yields a computationally creative process that has an input and an output and targets a class of end users. Fig. 1 depicts what we consider the core dimensions for defining game orchestration: from its input to the end user. Accordingly, this paper is structured along these core dimensions. In particular, the six facets of game content to be orchestrated (i.e., *what?*) are elaborated in Section II. Then, the three types of input (*from where?*) are discussed in Section III, including direct input from a *human creator*, input from online sources (as human-based computation), and input that is already *embedded* in the generative process. *How* orchestration can take place—along with related work on AI coordination—is discussed in Section IV, while how humans can intervene in this process is discussed in Section V. More specifically, Section IV explores the spectrum between *top-down* and *bottom-up* approaches; Section V explores the degree of human intervention in the process (from *noninteractive* all the way to continuously *interactive*). The types of intended end users (*for whom?*) are briefly presented in Section VI. In the context of facet orchestration, Section VII describes some influential work, which combines generation across facets, with a comparative analysis in Section VIII. Open questions and challenges are sketched out in Section IX, and this paper concludes with Section X.

II. Creative Facets of Games

Games are traditionally realized by a team of creative individuals with different skillsets and team roles. Based on the taxonomy of [4], the following subsections detail the six creative facets of games from the perspective of both human creativity and AI-based generation.

A. Visuals

As most of the digital games are rendered on a display, their visual representation is an important component and often a selling point. Game visuals range from photorealistic to abstract, and everything in-between [6]. Many games rely on the photorealistic depiction of real people, e.g., in *FIFA* (Electronic Arts, 2017) or imaginary people and locations, e.g., in *Skyrim* (Bethesda, 2011). On the other hand, games often rely on the exaggeration of caricatures, either to offer visual cues even in low resolution, e.g., in the extraordinarily large weapons in *Final Fantasy VII* (Square Enix, 1997) or to elicit a specific emotion as in dark grayscale representations of the unfriendly world of *Limbo* (Playdead, 2010).

Algorithmically generated visuals originated around mathematical models of noise [7], which were instrumental for the procedural generation of many visual features, from textures [8] to terrain [9]. According to the type of features needed in a game world, a large variety of procedural modeling methods and techniques have been proposed to generate them [10]. Most of the methods for generating visuals are based on computer graphics techniques; however, there have been a few attempts at AI-based generation of visuals, such as the evolution of graphic shaders toward a designer-specified color palette [11], procedural filters based on semantics for slight visual changes to modeled scenes [12], and evolution of arcade-style spaceships toward visual properties such as symmetry, simplicity, and other patterns [13].

B. Audio

While often underestimated, a game's audio plays a significant role in the feel and mood of a game [14]. Background audio can set players at ease with rhythmic repetitive sequences, as in the case of *Moon Hunters* (Kitfox Games, 2016) or increase their tension with staccato cacophonous string instruments as in *Amnesia: The Dark Descent* (Frictional Games, 2010). Moreover, the fast heavy metal tracks of *Doom* (Bethesda, 2016) indicate moments and locations of high challenge, but also energize the player to perform the frantic run-and-gun gameplay necessary to overcome this challenge.

In terms of procedural sound, *Proteus* (Key and Kanaga 2013) uses the player's current location and viewpoint to turn ON and OFF certain prewritten and carefully designed sound channels, thus changing the soundscape. Other work, such as *Sonancia* [15], chooses from a range of prewritten sound tracks to play at specific events or areas of a game. Finally, Scirea *et al.* [16] use music generated in real time to foreshadow game events according to a prewritten narrative arc. Earlier examples of procedural music in games are surveyed in [17].

C. Narrative

While not all games require an elaborate narrative, a plethora of large-scale games feature an extensive storyline to connect the different locales and quest lines. The motivation to play and complete the game is often built around an in-world narrative. Role-playing games such as *Planescape: Torment* (Black Isle, 1999) are especially grounded in the lore of an elaborate world and introduce nonplayer characters (NPCs) with a rich backstory and personal growth throughout the game.

In terms of algorithmic narrative generation or mediation, there is extensive work in interactive narrative and drama management [18]; games such as *Façade* [19] and *Prom Week* [20] model the game state in a way that allows the manager to choose which NPCs utter which lines of preauthored dialogue. A recent survey [21] has pointed out that a stronger orchestration between plot and level generation techniques has a huge potential and impact on the authoring process of computational narratives for games.

D. Levels

Just as most of the digital games are displayed visually, their gameplay takes place in a virtual space. This virtual space is identified as a *game level* and can range from the extremely simple in *Pong* (Atari, 1972) to the labyrinthine in *Dishonored* (Arkane, 2012). A game may consist of numerous short levels, e.g., in *Super Mario Bros.* (Nintendo, 1985), or take place in one level spanning galaxies, e.g., in *Stellaris* (Paradox, 2016). Game levels need to combine form and function: the former should aid navigation via memorable visible landmarks, while the latter should constrain the paths of players (e.g., forming chokepoints in strategy games). Exceptions to these level design patterns abound: the horror feel of *Amnesia: The Dark Descent* is enhanced by low lighting and winding corridors, which limit visibility and increase the chance of "jump scares."

Level generation is by far the most popular domain of PCG in games, both in academia and in commercial titles of the last 35 years, from *Rogue* (Toy and Wichman, 1980) to *Civilization VI* (Firaxis, 2016). Level generation can be performed in a constructive manner [22] and via many other methods such as generative grammars [23], artificial evolution [24], declarative modeling [25], and constraint solving [26].

E. Rules

Regardless of the level they are in, players are bound by the game's *rules* and have access to its *mechanics*. Mechanics allow the player to interact with the world [27] and are usually described as verbs [28] such as "jump" in *Super Mario Bros.* or "take cover" in *Gears of War* (Epic Games, 2006). On the other hand, game rules determine the transition between game states, e.g., after a player uses a mechanic. Some rules may lead to winning or losing, e.g., if Pac-man eats (mechanic) the last pellet in *Pac-Man* (Namco, 1980), then the level is won (rule), or if Mario fails to jump (mechanic) over a gap in *Super Mario Bros.*, then they lose a life (rule). While rules are different from mechanics, as "rules are modeled after [player] agency, while mechanics are modeled for agency" [27], for brevity, we use the term "rules" for this facet in the typology to include mechanics, rules, winning and losing conditions, etc.

AI-based generation of rules and mechanics is one of the most challenging aspects of game generation [29], not only because they greatly affect the playability of the game, but also because arguably their quality can only be assessed via playtesting. *Ludi* evolves interaction rules in board games [30], [31], and is analyzed in Section VII. In digital games, several early attempts at automated game design have focused on abstract arcade games, generating movement schemes and collision rules based on designers' constraints [32] or based on the ability of an AI controller to learn the game [33].

F. Gameplay

While the other facets focus on how a team of human or computational developers create a game, the experience of the end user playing through the game cannot be ignored. Sicart

specifies that "gameplay, or the experience of a game, is the phenomenological process of an epistemic agent interacting with a formal system" [34, p. 104]. Each player interprets the visuals, level structures, narrative, and game rules in their own way, based as much on cultural and ethical preconceptions as on their in-game decisions (e.g., the order in which they visit locales in an open-world game such as *Skyrim*). The game's mechanics are prescripted when the player launches a game for the first time; however, when the mechanics are combined together and used to advance players' ad-hoc goals or approaches, they can lead to emergent *dynamics* [35]. Such dynamics can be influenced by social and competitive concerns on the part of the gamer community, which can lead to an ever-changing *metagame* [36] of strategies and counterstrategies. In extreme examples, players exploit unforeseen ramifications of the game rules in order to bypass the intended challenge. An exemplar player exploit (turned emergent game mechanic) is "rocket-jumping," where a player shoots a rocket at the ground, receiving damage and propelling through the air by the blast. Rocket jumping allows players to travel faster (breaking movement speed restrictions in the game rules) or access unreachable areas (breaking the intended level design limits). Beyond the primarily functional concerns of dynamics, however, the interaction among all facets (and especially visuals and audio) can evoke strong emotional responses by the player. These responses range from basic emotions such as fear and joy [37] or a broader range of *aesthetics* such as sensation and discovery [35]. While the intended emotions and aesthetics of players can be designed *a priori*, they can only be elicited during gameplay and may vary immensely from player to player and from those imagined by the designer.

Simulating human play via computational processes is the primary goal of AI agent control, which is the oldest and most popular field in game AI research [38]. Most important to our perspective of generator orchestration is the challenge of automated playtesting, where AI agents can learn to play any type of generated game and evaluate its quality (in terms of, e.g., playability, fairness, memorability, uniqueness, and more). Gameplay logs produced by AI agents are often used to derive the quality of generated content, e.g., in simulation-based fitness functions [24] for evolving game levels. In such simulations, the AI playtester often attempts to follow the optimal strategy to achieve the designated goal (such as gaining maximum score), e.g., when creating levels for an AI competition [39]. Such AI playtesters simulate an *achiever* type of player [40] or presume an aesthetic of *challenge* [35]. Assessing challenge does not require objectively optimal agent behavior, however. Artificial drivers [41] have attempted to maximize "objective" efficiency (i.e., distance covered in a preset time), while minimizing deviations from captured player data in terms of steering and acceleration; this project attempts to more closely match how human players approach this challenge. In other work on AI playtesting, a broader set of agents attempt to solve the *MiniDungeons* puzzle game [42] targeting different objectives, such as collecting the largest treasure or taking the least steps. In this way, the notion of performance is personalized based on the priorities of players; the artificial playtraces of such agents can then be compared with human traces in order to classify players into "archetypical personas" akin to [40].

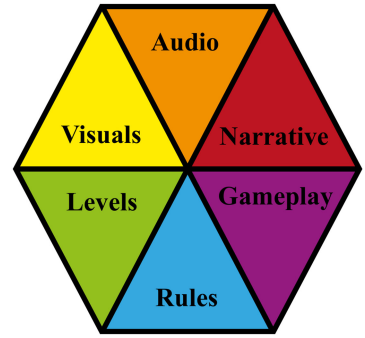


Fig. 2. Creative facets of games.

G. Meta-Facet Issues

So far, we have attempted to categorize elements of games into the six general facets of Fig. 2. However, we acknowledge that not all elements of games can be described in this way, and some elements seem not to fit under just one facet. We now attempt to address these more contentious issues.

1) Question of Blurred Borders: It is expected that any game, as a multifaceted experience, would have unclear distinctions between the different elements of visuals, audio, etc. Since level design combines form and function, the borders between level and visuals become somewhat blurred. For example, a level in *Amnesia: The Dark Descent* has carefully placed lights: the rest of the level consists of dark shadows. The placement of lights is strategic, as it forces the player along specific pathways. However, lighting and ambient brightness are directly tied to the game's visuals, renderer settings, etc. In this case, the placement of lights is as much a part of the level design as it is of the game's visuals.

A more ubiquitous instance of blurred borders is gameplay. While a designer defines rules and mechanics, how these mechanics will be used—or the rules be exploited—by the player is a part of gameplay. Following the mechanics-dynamics-aesthetics framework [35], the borders between rules and gameplay become even more blurred: while defining the mechanics is firmly part of the rules facet, the designer should anticipate how mechanics will be used or combined (into dynamics) in order to balance different strategies. However, gameplay dynamics are saturated by context and trends in a player community, walkthroughs on third-party websites, viral videos, etc. Finally, only through gameplay can a player experience emotions or the aesthetics of play, and thus, even a game designer must resort to gameplay testing (i.e., become a player) to verify these aesthetic outcomes.

2) Question of NPC Behavior: NPCs are by themselves multifaceted elements of games, relying on a memorable appearance, voice-acting, character growth, etc. On the other hand, considering NPCs as intelligent agents begs the question if such intelligence requires an additional facet. We consider NPC behavior traditionally exhibited in games (and in AI research)

to belong primarily to the gameplay facet. Specifically, NPCs playing the game in a similar way as the player count as automated gameplay. This is especially true for agents that play the game using the same rules as the player (e.g., in adversarial symmetrical games such as chess) or with minor changes (e.g., asymmetrical gameplay, where the AI has more knowledge of the game state as in the *Civilization* series). For all intents and purposes, the AI is attempting to emulate a player's decision-making processes, and thus, it generates functional aspects of the gameplay facet.

Beyond playing the game strategically as a player would, research in believable agent behavior where NPCs attempt to emulate how humans would react to a situation similarly targets automated gameplay. For example, *Prom Week* [20] uses a complex internal model for each NPC regarding relationships, traits, and cultural knowledge. These NPCs do not attempt to "win" the game as a player would; they attempt to be human-like in their reactions and thus emulate a human experience when seeking a date to the prom. While not a player per se in this case, emulating human-like dynamics and aesthetics in a multi-agent system is akin to prompting the aesthetic of fellowship in multiplayer games and falls under gameplay.

That said, simple NPC behaviors such as the rule-based system of monsters in *Spelunky* (Yu, 2008) or *Super Mario Bros.*, which patrol to the edge of a platform and then change direction, would be a stretch to ascribe to gameplay. In such games, NPCs (if they can even be named so) act as dynamic obstacles which move in a simple deterministic pattern; their behavior rules are merely part of the rules facet, while their placement in a level is a part of the level design facet, but their resulting behavior does not quite match the gameplay facet.

3) Question of Genre: The notion of game *genre* has not been highlighted in the above distinction between facets, as most of the genres affect all facets—although usually not in equal degrees. It could be argued that genre conventions can become the connecting thread between different facets. Based on the long history of games and their categorization, specifying a genre such as turn-based strategy comes with many assumptions that narrow down the possibility space of most of the facets (e.g., top-down visuals, rock-paper-scissors game rules, war narrative, etc.). While most of the game genres primarily define interactions with the game system (i.e., rules, mechanics, goals, and winning conditions), merging genre with the rules facet would be a risky generalization. A game genre often describes and constrains more than one facet: the *music game* genre constrains the rules (i.e., synchronize player input to the sounds), the level design (i.e., a level structure that allows a player to foresee the next few keys that must be pressed), and of course the audio. As genre imposes constraints on several facets, it can provide the necessary context and anchor for orchestration if it is identified before any generation takes place. Demonstrably, all case studies in Section VII were constructed with a specific game genre in mind (e.g., arcade).

4) Question of Facet Hierarchy: When attempting to generate content for many different facets, it is perhaps inevitable to ask "where do we start from?" as some facets may hinge on the pre-existence of others. An obvious example is gameplay, which requires game rules and a level (at the minimum) in order to occur. On the other hand, a computational or human designer can draw inspiration for a game from a song or a book and can pitch a game to the studio based on its visual style or historical setting. When it comes to actual game production, however, we argue that the rules facet comes first. In a generative pipeline, the rules of the game and its goals would need to be generated ahead of the level, visuals, or a concrete narrative. In many examples of game generation, including those in Section VII, the game rules and ending conditions are implied (e.g., reaching 0 hit points loses the game) based on genre conventions.

This does not mean that a ruleset should be emphasized in the final game, i.e., the argument is not for "mechanics-heavy" design. Different games may foreground different factors of player experience or different design elements in the final product (e.g., adventure games often foreground the story and dialog). However, the main gameplay loop (itself in the gameplay facet) and the aesthetics of the player experience [35] are ultimately shaped by the game's mechanics, goals, losing conditions, and their interrelation.

III. Inputs to Orchestration

In this paper, we assume that AI is the main driver not only of the orchestration process (discussed in Section IV), but also of every generative task. However, the inspiration or guidelines for these tasks may originate from sources outside the confines of the algorithmic codebase. We shortly survey possible inputs to the generative processes being orchestrated.

A. Input From a Human Creator

A human creator or player can often customize a generative process based on their preferences. The degree and impact of human input varies. Many games such as *Stellaris* allow the player to customize a few intuitive parameters such as the number of players (enemies), which directly affects the game's difficulty and dynamics. Parameterization of the generator on a case-by-case basis by the user is also available, where orchestration is concerned: in Game Forge [43], a user can express spatial preferences for the final level (e.g., "branchiness"), but the level generator must still obey the narrative structure of the underlying storyline. While in *Stellaris* the user customizes parameters of a scripted generator, in Game Forge, player preferences directly affect the objective of an evolutionary algorithm. Similar objectives can be tailored through a graphical user interface as a target hue selected by the user on a color wheel [11], less directly as an intended tension curve [15], or inferred based on player interactions with generated results [44]. Human input can also take the form of English text: A Rogue Dream [45] and WikiMystery [46] require a single word or a person's name as input, respectively, to draw inspiration from. Extensive human authoring may also be required: Game-O-Matic [47] requires a user-created graph with customized edges and nodes, while mission graphs in *Dwarf Quest* must similarly be hand-authored along with their node types [48].

B. Input From Human-Based Computation

There is an ever-expanding volume of data available online, and human users constantly engage with each other and with web content. A multitude of software programs perform *human-based computation* by outsourcing several steps of their processes to a crowd of human users. It is not common for generators to rely on human-based computation, although there are several noteworthy examples beyond games. Google N-grams have been used to find associations and similes between words [49] and transform one character role into another [50]. In nongame orchestration, online newspaper articles have been used as a seed to create collages, the constituent images of which were collected based on Flickr searches [51].

For game generation, human computation has been used to learn patterns from stored game content available online, such as video playthroughs [52] and human-authored game levels [53] to inspire level generation for arcade games. On the other hand, data games [54] transform open data into playable content. Data Adventures [55] and Angelina [56] use online sources, which are constantly updated such as Wikipedia or The Guardian newspaper (respectively); thus, a generated game may differ from one day to the next based on recent changes. Finally, games such as A Rogue Dream [45] and Game-O-Matic [47] use online repositories such as Google images to find appropriate artwork to integrate in their games; this bypasses the serious challenge of generating appropriate and visually relatable visuals for many of the complex or contemporary content semantics that these games produce.

C. Embedded Input

For automated generation, there tends to be an assumption that the entirety of the world knowledge is encoded (or generated) within the codebase, which creates content. This decision is due to practical concerns (e.g., not all human-editable parameters are intuitive, and online queries can be mercurial or finite) as much as it is due to the desire for fully autonomous generation—as computational creativity often aspires to [57]. The simplest type of embedded input is a random seed: how this seed is transformed into content depends on more complex embedded structures such as rulesets (e.g., in cellular automata) or lookup tables (e.g., random encounter tables). For many generators of narrative, a full world model must be embedded [58]. Similarly, most of the level generators encode all possible tiles and their relationships *a priori*; if the evaluation of generated levels requires a simulated playthrough, the gameplaying agent is also hard-coded *a priori* into the system, e.g., in [59]. Orchestration is arguably easier when the entire system is contained within the codebase, especially if the orchestrating software has knowledge of each generator's world model. For instance, Ludi [30] orchestrates level and rule generation by integrating both in the same genotype; a game player encoded in the same software can directly return a fitness score to the search-based rule/board generator. On the other hand, orchestration may not rely on external inputs except on specific generative steps: for instance, while Sonancia can generate the intended tension progression and level without any human input, the last step where sounds are added to the level requires external input from crowd-sourced tension models [60].

In theory, a fully internalized orchestration module seems ideal for fast and efficient coordination; however, the main challenge is the onerous and sometimes infeasible task of encoding a world model into the generator. While narrative generators include a thorough knowledge model, embedding it is a very tedious task, which requires extensive textual input even for minimal story domains. When more game facets are considered, such as visuals that represent real people, then the complexity of such an internal model is too large and external input (e.g., online sources) is the only viable solution.

IV. Orchestration Processes

As noted in Section I, we borrow the metaphor of *orchestration* to contextualize the collaboration of multiple computational designers, each focusing on the creation of content primarily for one facet: examples include level generators, ruleset generators, artificial playtesters, etc. In that context, we identify two ends of the spectrum, which have been heavily researched both for game development and for general production (algorithmic or not). On one end, the *top-down process* features a composer, which provides as much detail as possible to musicians (individual generators), leaving little room for creativity to each musician. On the other end, the *bottom-up process* features a group of musicians "jamming" until they find a common frame of reference and adjust their performances toward a shared emergent rhythm and complementing melodies (see Fig. 1).

The sections below attempt to unpack the notions of top-down and bottom-up processes for orchestration, proposing possible implementations for each of them, as well as framing them in the context of other possible metaphors and related work in broader AI coordination. Moreover, the fertile middle ground between these two ends of the generation spectrum is identified with some examples.

A. Top-Down Process: The Composer

The simplest way to achieve a consistent design seems to be to impose it *a priori* to all constituent members of a production team. In our music metaphor, this would be a composition written by a musical luminary, such as a concerto written by Vivaldi. Distributed as sheet music to each instrument player, the constituent musical pieces are played by the respective instruments. Taken at the face value, this resembles a production line at a manufacturing plant where machines (or humans, for that matter) are given a firm set of instructions, which they must execute with precision. In game production, this could be likened to the *waterfall model* [61], where a thorough game design document, created before production begins, informs all creative decisions down the line. In the waterfall model (see Fig. 3), the game is first designed on paper; then, implementation (be it graphical, functional, or other) takes place following the design document closely, followed by postprocessing and testing. The core principle of the waterfall model is that each step can start only after the steps before it are completed. As with a concerto, some common understanding on how to interpret the design

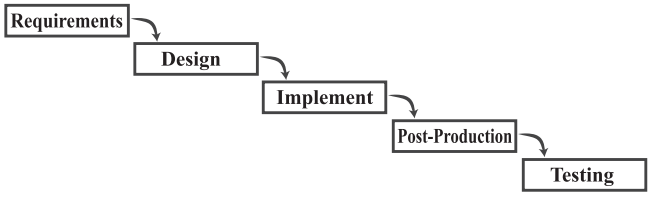


Fig. 3. General view of the waterfall game design process.

document is necessary—mediated by a maestro in orchestras and by art directors or tech leads in game development. This common understanding is further strengthened and finalized during postproduction, where all components come together and additions or fixes are made to better adhere to the design document. Admittedly, we oversimplify the role of maestros, art directors, or postproduction in this example to present the extreme end of the orchestration spectrum as a production line; often the creativity and influence of such individuals is imperative, and we will discuss this in Section IV-C.

1) Related Background Work: In generative systems, a waterfall model is best represented as the generative pipeline, also called a *feedforward* or *linear* approach. A general input is fed to one process marked as first in the pipeline. The first process produces some kind of output, which is handed off as input to the next process in the pipeline. The second process elaborates on or transforms its input as necessary and hands off its output to the third, and so on down the line until the final product is output. This is perhaps the most straightforward way of combining multiple generative systems and has, therefore, been used frequently in many domains, including in most of the multifacet game generation systems discussed in Section VII. It is important to highlight the role of *input* in this approach: this is most often a user specification (i.e., preauthored by a human designer) although the level of detail of this specification affects the creative leeway afforded by the generators within the pipeline.

Linear generative pipelines are also common as an architectural foil held up by systems that wish to move beyond them, perhaps because they are easy to make work but are unsatisfying as a model of the creative process. Critics correctly point out that humans do not produce creative artifacts in this purely linear manner, and instead, different facets of a creative domain may impact others in a multidirectional manner that includes interaction between facets and multiple revisions. For example, working in AI jazz improvisation, Horowitz [62] proposes several different interacting areas, such as melodic concepts, melodic lines, goals and emotions, context (meter and harmony), solo lines, low-level features (pitch and rhythm), etc. Each of these areas mutually impacts the others and, furthermore, may itself have multiple subfacets. In this system, a spreading-activation network is used for multiway interaction between those facets.

2) Envisioned Framework: Many of the case studies in Section VII use a generative pipeline of some sort. As an example, Sonancia [63] first generates the desired progression of tension (following film tropes), then uses it to generate a level that matches it as closely as possible, and then uses the actual tension in the generated level to choose sounds for each room.

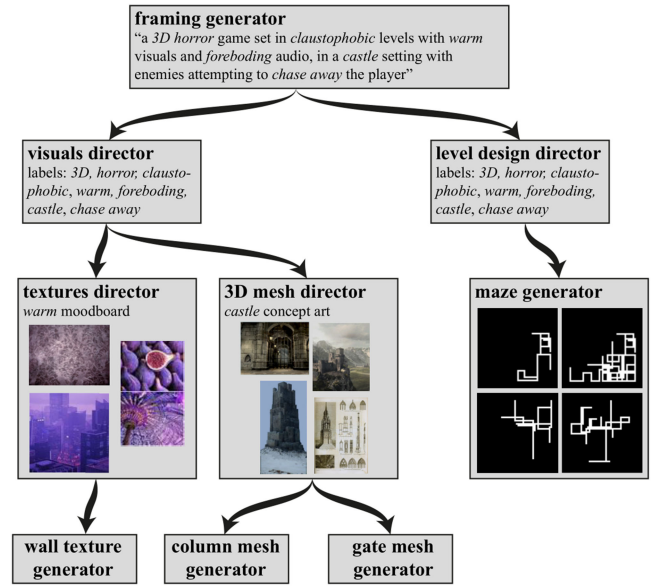


Fig. 4. Example top-down generative process based on frames that are refined by subsequent generators (as *directors*).

In order to give a broader and more inspiring framework for top-down generation, we envision a full game generation pipeline illustrated in Fig. 4. This system starts from a general game description and uses a pipeline to introduce more details until a complete game can be composed out of the outputs of specialized small-scale generators. This general game description acts as a frame [64], which can be generated or provided by a human designer, and it must identify the core principles, technical and semantic, in each creative facet. An example frame can include one or more semantically rich labels for each facet (e.g., "a *3-D horror* game set in *claustrophobic* levels with *warm* visuals and *foreboding* audio, in a *castle* setting with enemies attempting to *chase away* the player"). Ideally, the frame should be described in natural human language [65], although it suffices if the frame is only machine interpretable (described as fitness functions, grammars, ontologies, text files, or other parameters). The frame is the blueprint for the generated game and acts as *producer* in a game studio [66].

The high-level frame can be disseminated among the different single-facet generators, which produce content appropriate to the directives and constraints of this frame. However, the frame will need to be further refined in terms of each facet before content can be generated. Refinement can be done by lower level directors, focusing exclusively on each facet: an analogy can be made with, e.g., *art directors* in game studios, who coordinate artists via moodboards or concept art. The example frame above can be refined by a generator of art frames into art-specific guidelines such as "the game needs stone castle textures, 3-D meshes of narrow corridors, 3-D meshes of creepy statues, and animated meshes of undead monsters." These frames can be further refined (e.g., to define the exact dimensions and components of the wall meshes or the monsters), leading to a series of generated "primitive" components, which are recombined by the directors and provided back to the producer/frame to form

the generated game. This generative model allows for a more directed flow of information and a clear task allocation to the generators of "primitive" components. These generators do not need to be particularly "intelligent" or "creative" in that regard: constructive PCG methods likely suffice for the speedy creation of content. Assurance of quality and consistency is done by directors who narrow down the generative parameters sufficiently to ensure that any content generated will be appropriate. This top-down generative model follows largely the waterfall model of software development, where each phase leads to a subsequent phase more grandiose (and expensive) than the next. This holds true in the generative "waterfall" orchestration, as generators of high-level frames are computationally and conceptually simpler than generators of "primitive" components.

3) Challenges: As discussed earlier, the main argument against a generative pipeline is, perhaps counterintuitively, the simplicity in which it can be implemented. The pipeline's simplicity hides another danger that the constituent generators in the pipeline merely obey rules similar to a machine on the production line. A generative pipeline itself, under that lens, does not contain any AI or creativity. The creative challenge in this case is twofold: 1) how the original "frame" is generated; and 2) how the high-level frame is iteratively interpreted into progressively more precise descriptions and actionable generative commands. On the other hand, the top-down process handles the challenge of harmonization fairly efficiently by breaking it into smaller parts (sub-frames) and by ensuring that generators only produce content under very specific constraints that match that frame.

B. Bottom-Up Process: The Jam

As mentioned in Section I, music orchestration does not necessarily need a composer or maestro but can instead be done through jamming, e.g., freeform jazz musicians creating music by feeding off each other's riffs. A similar brainstorming and iterative development method is followed in more freeform game development settings, such as in game jams. Based on the survey of [67], when developing a game, the (human) participants of the Global Game Jam start from many ideas and iteratively reduce scope, or start from vague ideas and add mechanics and features during implementation, or start from a core idea and build it up based on testing and feedback. Of these processes, the first two hinge on the iterative refinement of one or more vague ideas, which are formalized both through testing and through conversation among team members.

Can generators "jam" like freeform jazz musicians or Global Game Jam participants? A possible bottom-up approach requires multiple generators, each contributing content pertaining to one facet, producing initially random content and then observing how their output matches other generators' output. Initially, random content should be produced at a more abstract representation than the intended end results, e.g., as map sketches [2] rather than 3-D levels. Like freeform jazz musicians, the generators would then need to adapt and refine their generative process in order to better match the output of the other generators, aiming to reach a consensus. To achieve this, the generators would need a way to evaluate both their outputs and the outputs of other generators in order to assess how well the primitive components that each generator produces match. This can be done in several ways, e.g., based on labels as discussed in Section IV-A2: a mesh generator creating pieces of a *castle* (label) would not match a texture generator creating *sci-fi* tiles or an NPC name generator for *cyberpunk* settings. Several functional flaws of nonmatching components could be recognized during playthroughs by a generic AI player, e.g., when combining narrow maze-like levels with a control scheme of racing games. Finally, consistency can be evaluated in a completely data-driven fashion, which uses a vast database of existing games to learn patterns (e.g., via deep learning) and assess whether typical [68] associations between facets are present in the output of the different generators.

It is clear, therefore, that the proposed bottom-up approach to facet orchestration likely needs fewer generators than the top-down approach, as generators of framing information become unnecessary. On the other hand, the generators must be: 1) highly expressive, i.e., able to create a broad range of content; 2) able to adapt their process to match the output of other generators; and 3) able to assess how well their output matches that of other generators. For the latter point, evaluating asset consistency can be either included in every generator or take the form of an external AI playtester or an external AI data processing unit. Regardless, it would seem that achieving a fully automated bottom-up generative approach requires human-level aesthetic evaluation and adaptation capabilities.

1) Related Background Work: A set of independent generators collaborating in a shared design space is reminiscent of the *blackboard system* [69], [70], which is found in a large amount of classical AI work [71], [72]. These systems are based around a central data structure called the *blackboard*, which multiple independent processes can read from and write to. In this type of architecture, processes generally do not directly communicate; instead, they communicate implicitly through the blackboard by recognizing content on the blackboard that is relevant to their own operation. The intention is to couple processes only loosely, avoiding both the n^2 process-to-process communication explosion and the need to specify a fixed control flow and data model at the outset. To facilitate this, processes in blackboard systems must be able to ignore the blackboard content that they do not recognize. In fact, there is typically no explicit control flow at all; instead, processes asynchronously read the blackboard and take action as they see fit, thereby decentralizing the decision-making logic. However, a central scheduler may exert some high-level control by modifying, when each process is scheduled, its resources.

One early use of a blackboard approach to generate content in a creative design domain is in a series of systems from Hofstadter's group at Indiana University, which have been applied to generate creative analogies [73] and typefaces [74]. In these systems, the blackboard (which they call the Workspace) contains the current artifact in progress, along with a number of annotations that generative processes (which they call Codelets) can add to items on the Workspace, in addition to changing the items themselves. Codelets come in several flavors: some codelets notice low-level patterns and add an annotation proposing that

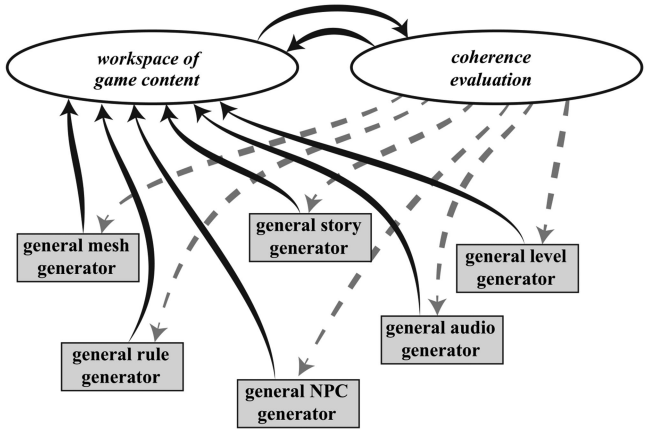


Fig. 5. Example bottom-up generative approach based on an external or internal system that evaluates the components' consistency and playability as a whole and may adapt the individual generators' parameters (shown as dashed arrows) based on these evaluations.

these patterns may be useful, other codelets try to enact a high-level goal (such as a particular analogical schema), and yet other codelets notice and/or attempt to fix specific types of problems that the distributed operation of the other codelets has produced.

2) Envisioned Framework: Inspired by blackboard systems, we envision a bottom-up process similar to Fig. 5, where a number of "general" generators for each creative facet produce artifacts that are then placed on the workspace. The generators are general in the sense that a generator can produce a broad range and style of artifacts, such as medieval textures (e.g., for stone castles or wooden carts) as well as sci-fi or modern textures. However, it is likely that these generators could be composed of more specific generators: a general audio generator could consist of generators for background music and generators for short sound effects. The workspace is a combination of all types of game content, which must be checked for coherency by some internal system (e.g., a data processing unit) or some external system (e.g., a general AI playtester). This coherence evaluation would, at the very least, reject a combination of game content and clear the workspace; it could also remove the most incoherent content, leaving more coherent content as a seed for future combinations. Ideally, the coherence evaluation system should adjust parameters of the generators so that they would produce content more appropriate for the workspace. Alternatively, the generators themselves could inspect the workspace and adjust their own parameters to harmonize with the patterns that emerged already. This is similar to how jazz musicians attempt to change their pace or notes to match the emerging melody created by the band.

3) Challenges: The largest challenge for generators able to brainstorm and eventually coalesce to a common goal and final artifact is the requirement for a very sophisticated appreciation model to assess not only their own but also other generators' creations. It should not be understated that jazz musicians jamming together are accomplished musicians individually, and that game jam participants have a common real-world model of the popular games' features in any facet. Thus, in a fully bottom-up approach, each generator should have a knowledge model, e.g., learned from existing games, which it can use to identify patterns in the generated data of other generators and adjust its own creative process to match those patterns more closely. The ability to appreciate other generators' output or the ability to adapt its own output can quickly escalate to an AI-complete problem. Similarly, a general AI playtester, which can account for (and take advantage of) nonfunctional elements, such as visuals or narrative cues during play, is similarly beyond the scope of the next decade of game AI research. However, several shortcuts can alleviate this challenge: for instance, using common labels between generators (which can create sci-fi themed visuals and sci-fi themed rules, for instance) would allow for a fairly simple coherency evaluation. This shortcut comes at the cost of expressiveness, since only certain labels or game themes can be accommodated in this case, but it is a stepping stone toward realizing more ambitious bottom-up generation.

C. Intermediate Approaches

Sections IV-A and IV-B elaborated on two edge cases, where generation follows a top-down or a bottom-up flow. To better illustrate each process, analogies with musical composition and commercial game production processes were used. However, many of the assumptions for how musicians or game developers are creative were oversimplified to offer the extreme ends of a spectrum. Orchestra musicians are hardly production line robots, and jazz musicians come to a jamming session with some assumptions (e.g., that they will be playing jazz). Similarly, even the most complete game design document does not contain the coordinates of all vertices in a 3-D model, and even in the most rigorous waterfall model, most of the discrepancies—often identified via internal playtests—are fixed during the postprocessing step. In game jams, jammers share abstract ideas first via sketches, agreeing on a basic premise before starting to create content. There is, therefore, a fertile middle ground between a strictly top-down and a purely organic bottom-up process when it comes to automating game generation. Using similar musical and game production analogies, we briefly highlight some promising directions for orchestration, which bridge the two extremes.

1) Creative Maestro: The top-down process of Section IV-A assumes that a composer is the solitary genius who disseminates more-or-less explicit orders to the musicians or, in our case, to generators of specific artifacts. The role of a director in Fig. 4 is to interpret the specifications of the high-level frame into actionable commands for simple constructive generators beneath it. However, these directors could interpret the provided frame much more loosely and creatively: for instance, the visual director of Fig. 4 could identify a castle as the expected medieval castle made of stone walls or—with a creative interpretation—as a flying fortress with sci-fi or mechanical walls. Such creative interpretations could lead to dissonance between, e.g., the visual output of a flying fortress and audio output for medieval throne rooms. To overcome this and achieve better orchestration, a dissonant direction (from the different directors) could suggest a change in the common frame, and that change would then have to be interpreted and propagated to all directors (and

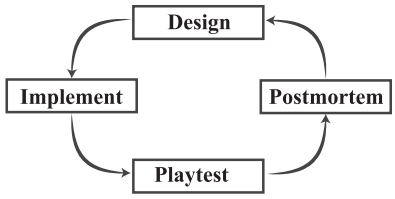


Fig. 6. General view of the iterative game design process.

thus all generators) for every facet. Having a director suggest a change of frame is contrary to the top-down process, where past decisions (or generative steps) are frozen. The benefit of such a change, on the other hand, could be a more creative process than a strict top-down process; at the same time, since only directors (using high-level labels) can suggest changes to the frame means that the process is more controlled than in pure bottom-up approaches, where the shared workspace is equally likely to contain low-level content such as textures or dialogue lines.

2) Jamming With Fake Sheets: The ambitious nature of bottom-up approaches has been highlighted in Section IV-B3. To take a plausible intermediate step toward fully bottom-up approaches, the notion of a "frame" similar to that in Fig. 4 (and discussed above) can be introduced. In the case of bottom-up approaches, a frame could act more as a "fake sheet" (or lead sheet) in music: fake sheets specify only the essential elements of a song (melody, harmony, and lyrics) and is often the only form of written music in jazz ensembles. A musician plays this melody, while other musicians improvise accompaniments or solos on top of this basic chord progression. In a bottom-up generative process, a fake sheet would then be a frame which all generators agree to build on and expand, following its essential structure without being subdued to it. The frame can originate from a human or a computational designer (similar to the frame generator in Fig. 4). The frame can be disseminated to the individual generators for adjusting their parameters and ensuring that their content more or less fits the frame; alternatively, it can be placed on the workspace just like any other artifact and inform the coherency evaluation system, which must not only ensure that generated content is coherent with each other, but also with that specific frame.

3) Vertical Slices: The grandiose vision of the previous sections has been the generation of full games as a result of combined efforts of orchestrated generators. The assumption has been that everything needed for this game is generated at the same time—no matter how long that takes. On the other hand, game development companies often follow an iterative approach (see Fig. 6), where each small game mechanic, level element, or idea is developed in isolation (so as not to endanger the general development schedule and pipeline) and tested, refined, and redesigned until it is ready to be integrated into the existing game. Working on such *vertical slices*, development becomes flexible and open to innovations, since any feature can be designed, tested, and refined or discarded without hurting the game. This can be integrated as a generative process: arguably, any level design task is a large vertical slice as the level can be tested (with existing game rules), refined, and included in the level collection or discarded. However, vertical slices can be useful for other facets as well as for a combination of facets, provided that game generation proceeds in an iterative fashion. As an example, an iteration can introduce a new game mechanic, which allows climbing vertical surfaces; this affects the level design, which now can feature sheer walls, and possibly other facets such as sound effect or animation for climbing. If any of these generated components underperforms (e.g., no appropriate animation can be devised for climbing, or levels become too easy regardless of adjustments), then the iteration is ignored, and the game content already generated remains as it was at the start of the iteration. Working on vertical slices allows for a more manageable breadth of possible content, as modifications of already generated content, which is suitable both for a top-down process (e.g., with the high level directive "Levels must be easier to speed run through") or for a bottom-up process (e.g., with a rule generator pitching a new mechanic). The smaller scale of these vertical slices will likely speed up generation of the slice itself but may also slow down the final development of the game if a large portion of the vertical slices are discarded because they produce worse results.

4) Postproduction (Repair): In the general top-down game development process of Fig. 3, the combined content after the implementation step is not immediately sent for testing but instead goes through a *postproduction* step. This step is largely overlooked in the described top-down orchestration process of Section IV-A2, which assumes that all content generated by the primitive generators can be combined together without issue. However, postproduction can identify flaws in the integration of different content (or codebases, in the case of both commercially developed and computationally generated settings). It can also take steps to smooth out any errors introduced during integration or, more ambitiously, any dissonance or incoherence among the different components being combined. Postproduction can be simulated programmatically and is often identified as a "repair" function on generated content. In PCG research, repair functions have been applied based on a constructive algorithm, which, for instance, removes inaccessible tiles in a level [75], and also modifies levels by applying filters learned from the high-quality content [76]. It is likely that deep learning can be used to produce repair filters similar to those in [76], but for a broader variety of content and facets—provided a sufficient corpus of such data to train from. Regardless of the method, postproduction is a necessary step for top-down processes in commercial settings and should be considered for top-down orchestration as well. However, bottom-up processes that place a diverse set of content on a workspace would likely also require a postproduction step in order to smoothly integrate such content into a playable game. Iterative generation through vertical slices would also require a postproduction step when the iteration is deemed successful and its changes need to be integrated with the game created so far. In short, a high-quality postproduction step that can identify and smoothen dissonance between generated content is a core element of orchestration and, in certain cases, could be considered the main method for orchestration itself.

V. Human Interaction in Orchestration Processes

This paper assumes that orchestration is largely automated by an AI; however, human intervention during the orchestration process should also be considered. Unlike human input to each generator discussed in Section III, in this context, intervention occurs *during* (rather than before) orchestration. As an example, interactive evolution [77] can be applied on the orchestration level to select the most appropriate or harmonious facet combinations. Interactive evolution allows users to directly select which content they find most appropriate in cases of single-facet content such as tracks [78], flowers [79], or spaceships [44]. Beyond mere selection of favorite content by visual inspection, humans can create data on the gameplay facet by playing the generated content; based on the gameplay facet, content of other facets such as levels [80] can be orchestrated. Interactive evolution has also allowed human intervention on orchestration between multiple game facets, such as space shooter weapons' visuals and trajectories (game rules) in *Galactic Arms Race* [81]. *AudioInSpace* [82] requires the player to test the weapons and select which weapons' visuals and rules are best orchestrated with the game's generated soundscape; the player can select which weapon they prefer but also which audio they prefer, allowing for more direct control over which facet should be adjusted to become more harmonious.

As with bottom-up and top-down approaches, there are different degrees of human interaction, which can be included in the orchestration process. Human interaction can use direct selection as in [78], [79], and [82] to replace the coherence evaluation of Fig. 5, indirectly learned designer preference models, which can automate evaluation until new data are offered by the player [44], [83], [84], or human-provided gameplay data to replace automated playtesting for the gameplay facet [80], [81]. Expanding on the latter case, gameplay data have so far been used to help generate new game rules, visuals, or levels based on a preauthored mapping between gameplay and the other content being generated: better levels feature more play time spent in combat [80]; better weapons are fired more often than others [81]. On the other hand, this mapping between different types of content is of paramount importance to orchestration (amounting to the coherence evaluation of Fig. 5) and could be further refined based on human associations, e.g., by players' self-reports that can be modeled via machine learning into a predictive model of coherence similar to [85].

VI. End Users of the Orchestration Process

Ultimately, AI-based game generation aims to create complete games that are immediately playable. For most of the generators (in one or more facets), the output is playable by a human or AI player [86]. However, other generators produce intermediate or unfinished content, which must be verified or edited by a designer before becoming playable. *WorldMachine* (Schmitt, 2005) produced masks and heightmaps [87] that are then edited by level designers to add 3-D meshes and game-specific objects for *Battlefield 3* (EA, 2011). Tanagra [26] generated platformer levels which the designer could then adjust further in the same interface before making them available for play. Sentient Sketchbook [88] generated map sketches in response to a designer's actions; these map sketches were used for concept development rather than as final playable output. Generators of Role-Playing Game content[1] provide a springboard for game masters, who can adapt (or ignore) the generated results to fit their current needs or their campaign's backstory.

When it comes to the output of orchestration, the question of intended end user becomes even more pertinent. While ideally the outcome of orchestration should be a game ready to be consumed by a player, this requires a perfectly harmonized and balanced set of artifacts. As noted already, the problem of coherence evaluation can be AI-complete, while designing generators for every type of content (of every genre, and every facet) could be infeasible. Until these problems are solved—or instead of attempting to solve these problems—the output of a creative set of generators can be provided to designers as a springboard for authoring a game themselves. When the end user is a designer, the granularity of the generated output is far more flexible. AI-based orchestration can provide a broad direction such as a game pitch described textually or visually (e.g., a collage or logo), or it can provide a full game (e.g., structured as Fig. 4) but with the necessary options for a designer to adjust or reconfigure any level of the generation (e.g., on the broad art direction level or on the color level).

VII. Case Studies of Orchestration

Several research projects have targeted, in one way or another, the cocreation of multiple game facets. While these projects do not fully realize the goals set out in this paper, they are worth studying as their principles can be extended for a better orchestration between facets. Section VIII summarizes and compares these projects along the dimensions of Fig. 1.

A. Angelina

Angelina is a creative software developed from 2011 to 2016, which makes tracking its different versions difficult. For the purposes of this paper, we focus on a version described in [56], which scrapes information from online sources (e.g., stories from The Guardian news site) to create simple platformer games. Angelina evaluates the mood of the article based on natural language processing and chooses appropriate image backgrounds and sound-bytes based on the text contents (e.g., an image of a sad British Prime Minister if the article has a negative piece on U.K. politics). While the generated platformer level is not affected by the article's content or mood, the game's visuals and soundscape are orchestrated by the (high-level) narrative of the news piece (see Fig. 7).

B. Game-O-Matic

Game-O-Matic is an AI-based game generator, which transforms human-authored microrhetorics [90] into playable arcade games. Game-O-Matic is intended for journalists to quickly create newsgames, i.e., a type of game where "simulation meets

[1] Many RPG generators can be found online at `chaoticshiny.com`

Fig. 7. Facet orchestration in Angelina, where different online sources are used to combine visuals and audio based on the mood and keywords of a Guardian article acting as (external) narrative. The level generator, however, was not connected to the remaining facets. The in-game screenshot is from [56].

political cartoons,"[2] by constructing relationships between entities. These relationships take the form of a *concept map*, a directed graph connecting entities through verbs: e.g., "cows make burgers," "man eats burgers" [90]. While the author can create any sort of entity, the verb in each relationship must be chosen from a predetermined yet extensive list. These verbs are transformed algorithmically into game mechanics via predetermined rules. Thus, "man eats burgers" may be transformed into a game, where the player controls a "burger" chased by "man" avatars, and the game is lost if it collides with a "man" avatar, or the player controls the single "man" avatar who wins by colliding with all on-screen "burgers." When combined together, the different verb-entity triplets may create infeasible game rules [47] or games which cannot be completed: the partial game description is then modified by one of many possible *recipes*, which best fits the partial game description. Sprites for entities (e.g., "burger") are based on Google image search results for that entity's name.

Game-O-Matic, therefore, primarily interprets human-authored concept maps (microrhetorics) into a complete ruleset (i.e., with custom game mechanics, goals, and instruction sets). Additionally, the visuals of the game are fully dependent on the entities chosen in the microrhetoric. While one could argue that the visual generation in this case is superficial, it cannot be denied that different visuals (and underlying entities) result in a wholly different message. Treanor [89, g. 27] demonstrates how the same mechanics can have very different political and religious messages by merely changing the visuals of the game objects. Finally, since Game-O-Matic determines how entities of each type will be instantiated, the system superficially configures the level setup (see Fig. 8).

C. Rogue Dream

A Rogue Dream [45] is a roguelite game prototype, which uses online sources to discover associations between game objects in order to instantiate preauthored rules templates such as "⟨enemy⟩ damages ⟨avatar⟩." Unlike Angelina, the human input for the narrative is a single word: the name of the player's avatar. Users provide this name as a form of proto-narrative, which is strengthened algorithmically with names for enemies, edible items, and game goal. These are discovered through

[2]The slogan of `newsgaming.com` by Gonzalo Frasca *et al.*

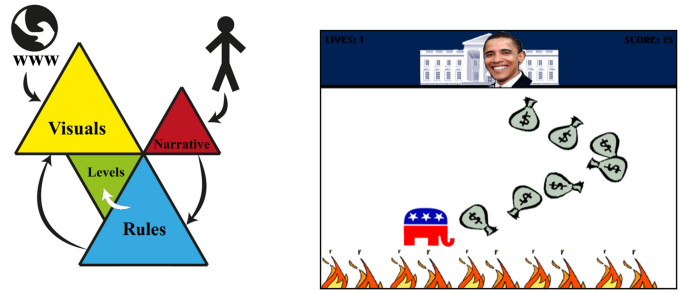


Fig. 8. Facet orchestration in Game-O-Matic, where a human-authored microrhetoric (small-scale narrative) informs which game objects and rules exist in the gameworld; game objects get their visuals from online sources through a search query based on the microrhetoric. In-game screenshot is from [89].

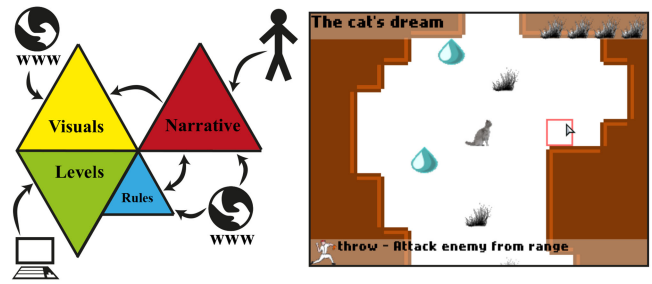


Fig. 9. Facet orchestration in A Rogue Dream, where a user-provided name for the avatar becomes the seed for discovering the names (and from them, the visuals) of enemies, goals, and healing items. A special ability name and mechanic is based on the avatar name (and a pre-authored list of possible game mechanics). The level generator, however, was not connected to the remaining facets. In-game screenshot is from [45].

Google's autocomplete results using templates such as "why do ⟨avatar⟩ hate..." for discovering enemies (as the next word of Google's autocomplete results). The game mechanics are prescripted (e.g., the player will have to reach the goal and edible items heal a prespecified amount of damage up to a maximum prespecified hit point limit). The only generated component for the game rules is the avatar's ability, which is also discovered through Google autocomplete result for the query "why do ⟨avatar⟩...." The verbs found as results of this query are matched to a prescripted list of possible abilities such as ranged attacks; if no match is found, a random ability is linked to the verb (i.e., there is a fallback that decouples narrative and rules, if orchestration is impossible). Similar to Angelina, A Rogue Dream generates a simple grid-based level with enemies, healing items, and a goal dispersed in it, disregarding their instantiated names. Similar to Game-O-Matic, the names of avatar and discovered associations (including abilities) are used as search queries to find the visuals used in the level and user interface (see Fig. 9).

D. Data Adventures

While most of the instances of multifaceted generators are based on content generated from scratch, this does not have to be the case. The Data Adventures series [46], [55], [91] create simple adventure games based on components already existing and freely available as *open access data*. Using primary sources of open content such as Wikpedia for data, Wikimedia Commons for images, and OpenStreetMap for levels, Data

Fig. 10. Facet orchestration in Data Adventures, which links Wikipedia articles together into a generated plot and searches for visuals based on article titles. Level design is based on the real location of cities on the globe, while city layouts are based on OpenStreetMap. In-game screenshot is from [91].

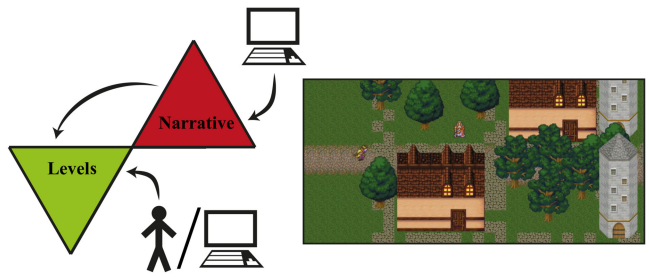


Fig. 11. Facet orchestration in GAME FORGE where a generated narrative informs level generation. Level properties (e.g., side paths) that do not break narrative constraints can be customized by the player. In-game screenshot is from [43].

Adventures recombine that information in novel (and often absurd) ways to create adventures for finding an NPC specified at the start of the game [55] or for discovering the culprit of a murder [46], [91]. The generative system finds links between people, places, and topics in Wikipedia and crafts a series of encounters with different "real" people (i.e., existing in Wikipedia) in locations around the globe, the maps of which are collected from OpenStreetMap; finally, the images of places and people are collected from Wikimedia Commons. From an orchestration perspective, the Data Adventures system acts as the "maestro" that collects semantic associations and generates proto-narratives from them ("the culprit may be X because it has the same profession as Y"), geolocations of different cities to generate the level (world map), and appropriate visuals for these people and locations. While Data Adventures operate on a different level than traditional PCG approaches, and not all their constituent generators are as sophisticated, the outcomes are some of the most elaborate instances of game generation in the form of an extensive highly branching adventure game.

E. Game Forge

An earlier instance where generated content of one facet is used as input for generating another facet is GAME FORGE [43]. This system begins from a generated narrative, where the story is represented as a sequence of hero and NPC actions at specific plot points, and generates a level layout so that the positions specified in the plot are visited in order. As an example, if a paladin must kill Baba Yaga to earn the king's trust in order to receive information about a secret treasure cave, then the generated level places the lair of Baba Yaga close to the king's palace, but the treasure cave is accessed from the palace only by crossing the lair. Moreover, the generator adds variety with locales unrelated to the plot to increase unpredictability. The level's target features (e.g., world size, number, and length of side paths) are specified by the player and form an objective for evolving the level toward the player-specified features and the narrative constraints. GAME FORGE uses the designer-provided or computer-generated narrative (quest-line) to guide level generation, but also accounts for player preferences (see Fig. 11).

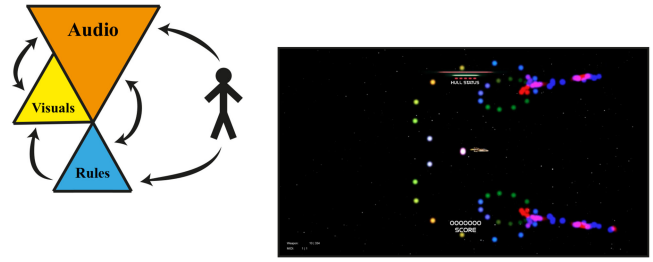


Fig. 12. Facet orchestration in AudioInSpace, where players select which weapon or music mappings to evolve. Weapon particles can have different colors. Audio affects the weapon's behavior; a fired particle, its speed, and color can affect the audio pitch or volume. In-game screenshot is from [82].

F. AudioInSpace

AudioInSpace [82] is a space shooter similar to *R-type* (Irem 1987), where the game's soundtrack affects the behavior of the player's weapons and vice versa. The weapon's bullets are represented as particles, the position and color of which are controlled by a compositional pattern producing network (CPPN) [92] that uses the game audio's current pitch information and the position of the bullet (relative to where it was fired from) as its input. This allows the audio to indirectly control the trajectory, speed, and color of the players' bullets; the player can control the behavior of their weapons via interactive evolution, choosing their favorite weapon among 12 options.

On the other end, the player's bullets (part of the rules facet) and the player's firing actions (part of the gameplay facet) affect the audio being played. New notes occur when the bullet hits an enemy or otherwise at the end of the current note. The audio, represented as the note's pitch and its duration, is controlled by a second CPPN, which uses as input the position from where the last bullet was fired, the time since it was fired, whether it hit an enemy, and its color. Thus, the player's firing behavior (i.e., how often they fire, and how accurately) and the weapons' visuals can affect the notes played. This creates an interesting loop, where one CPPN uses the audio to influence the weapons, while another CPPN makes the weapons' and player's behavior affect the music played. Both CPPNs can be evolved by the player, which makes interfacing with the system difficult as it is unclear which facet is currently influencing the other. To our knowledge, this is the first attempt at creating a closed loop, where rules, gameplay, visuals (as the particles' color), and audio are orchestrated with the player acting as a "maestro" controlling how each facet affects the others (see Fig. 12).

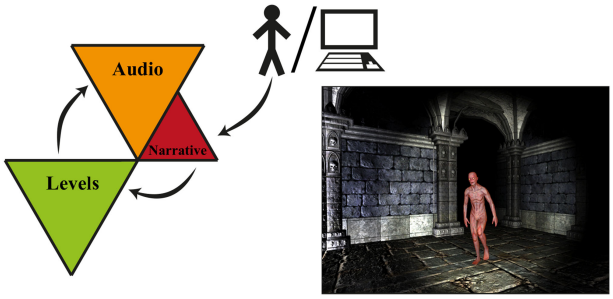


Fig. 13. Facet orchestration in Sonancia, which generates levels based on a generated or authored tension progression; the level's progression is then used to sonify each room. In-game screenshot is from [93].

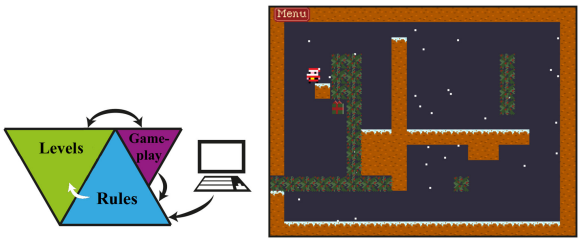


Fig. 14. Facet orchestration in Mechanic Miner, where existing game code is adjusted to allow new gameplay, and then, levels are evolved to take advantage of the new mechanics, assessed on random agents' gameplay. In-game screenshot is from [94].

G. Sonancia

Sonancia [63] is a generative system, which evolves levels for horror games based on a desired progression of tension and then uses the levels themselves to produce a soundscape for the experience. The model of tension defines how tension should increase or decrease as the players get closer to the end of the level; this tension model is generated first—or alternatively provided by a game designer [15]—and acts as the blueprint which the level generator tries to adhere to. Level generation is performed via an evolutionary algorithm, which tries to match the provided model of tension to the generated level's progression of tension, which is affected by the presence or absence of monsters in the rooms of the level.

Once the level is generated, its own tension model is used to allocate preauthored background sounds to it, in a fashion that corresponds to the tension progression. Each sound has a tension value, which can be defined by an expert designer [15] or derived from crowd-sourcing [60]. Each room is assigned a background sound, which loops while a player is inside it. By using the generated room's tension value (which depends on whether the room has a monster and whether there are monsters before this room), the evolutionary level design facet affects the constructive audio facet. Both facets are also guided more or less directly by the framing information of tension, which can include narrative terms such as "rising tension" or "cliffhanger" [63], although such framing information could be considered a narrative structure only at a very high level (see Fig. 13).

H. Mechanic Miner

Mechanic Miner [95] generates game rules by adapting the source code of a platformer game (e.g., generating a player action that sets gravity to a negative value) and then generates levels, which can be completed (i.e., the exit can be reached) with these new rules. Playability of generated levels is ensured by an agent performing random actions. A more intelligent agent that learns to use the mechanics could lead to a stronger orchestration of AI-based gameplay taking advantage of AI-based rule generation to adapt AI-based level design.

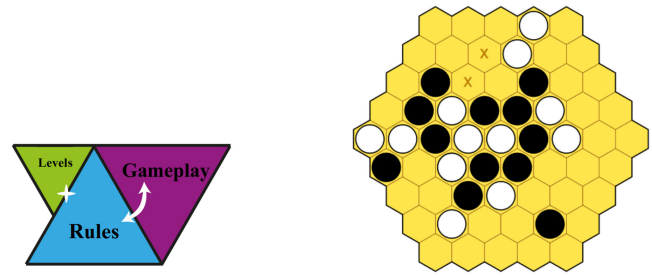


Fig. 15. Facet orchestration in Ludi, where the rules for black, white and neutral pieces are evolved alongside a board layout (i.e., a game level). Evaluation of the board and game piece rules is based on simulated playthroughs of two artificial agents using the same evolved policy based controllers. Screenshot of generated game Yavalath is from [30].

I. Ludi

While this paper has focused on AI-based generation of digital games, work on orchestration of board game facets shows great promise in the Ludi project [30], [31]. Ludi creates two-player adversarial games, abstract in nature and similar to checkers or tic-tac-toe. The game's winning and losing conditions, rules for piece movement, and board layout are described in a custom general description language with *ludemes* as "units of game information" [30] (e.g., a board ludeme or start ludeme). The game description combines game rules and the design of the level (board), although the possible board layouts are fairly limited compared to levels in digital games. The game description is then tested through simulated gameplay by two adversarial agents that use a policy evolved specifically for this game from a set of pre-authored policy advisors. The produced gameplay logs from a set of such playthroughs are parsed to assess objective properties (e.g., completion rate and game duration), but also aesthetic properties (e.g., drama and uncertainty). Ludi's generated gameplay simulates players' learning (initial moves are random, for instance). Gameplay logs are processed extensively to find 57 gameplay features, which are then combined into an aggregated fitness score to bias the genetic selection of game descriptions. Orchestration in Ludi largely follows a bottom-up approach, adapting game rules and board layouts based on feedback from the artificial players, which, in turn, adapt to the specific generated game and take advantage of its board layout (see Fig. 15).

J. Other Examples

The limited survey above describes a few exemplars of AI-based game generation that incorporate multiple game facets. The survey is far from exhaustive, but it aims to highlight a broad range of approaches, different facets, and different inputs that seed generation. However, several other projects on AI-based

game generation have attempted to orchestrate content and are worth noting.

Arguably, any generate-and-test PCG method or simulation-based evaluation in search-based PCG [24] generates the gameplay facet in one way or another. Gameplay is simulated in level generators via pathfinding between the level's start and finish, e.g., in [96] or via rules on when players should use a mechanic, e.g., in [97], or via solvers for optimal puzzle solution, e.g., in [26] and [98]. Such simulation-based evaluations usually trivialize the player's expected experience or aesthetics. On the other hand, Cook[99] uses the same trivial A* pathfinding playtraces but uses the computer agent's viewport to assess whether certain markers are visible or not visible. This artificial playtrace evaluates generated levels based on the visual stimuli rather than purely functional aspects of player experience (i.e., completing the level). On the other hand, rule-based systems in [97] simulate player precision by introducing some randomness to the timing of an artificial player's use of a mechanic. This better captures player experience and can be used to assess how accessible or difficult a game is to novice players. Focusing instead on human players' different priorities when playing a game, levels for MiniDungeons [59] were evolved for different procedural personas, i.e., artificial agents with archetypical player goals such as treasure collection, challenge (monster killing), or survival [42]. Finally, racing track generation in [100] was informed by gameplay traces of computational agents that simulated specific players' skills captured via machine learning.

Beyond simulation-based fitness functions for level generation, gameplay and rulesets are orchestrated in a similar way. For instance, constraint programming can produce "optimal" gameplay traces for a broad set of generated collision-based arcade games in [32]. Game-O-Matic in Section VII-B performs a similar playability check; however, such gameplay generation is deemed too trivial to include as a generated facet.[3] In contrast, Ludi creates gameplay logs by agents that use customized policies to that particular game and simulate learning (or initial lack of understanding). Similarly, Togelius and Schmidhuber [33] evaluate generated collision and scoring rules for simple arcade games based on controllers evolved explicitly for this game. Unlike Ludi, the evaluation is based on the average fitness of these controllers throughout evolution, simulating how difficult it would be for a player to learn (optimize) their gameplay towards maximizing the score.

Nelson and Mateas [101], [102] proposed a four-facet model that partly overlaps with that of Fig. 2 and implemented a generator of *WarioWare* (Nintendo, 2003) style games orchestrating a subset of those facets. Those four facets were: abstract mechanics (corresponding to our rules facet), concrete game representation (a mixture of our visual and audio facets), thematic mapping (our narrative facet, plus the aspects of visuals that establish setting and meaning), and control mappings (subsumed in our rules facet). Their generator takes a high-level micronarrative provided by the user (such as a game about *chasing*) and finds a combination of game mechanics and sprites from a preauthored set to produce the narrative. It thus follows a top-down process that starts from the narrative facet and then jointly searches the rules and visuals facets for a suitable content pair.

[3]Similarly, Sonancia uses A* pathfinding to check that the objective in a level can be reached, but that is not considered generated gameplay.

TABLE I
PROPERTIES OF THE CASE STUDIES OF SECTION VII

Case	Order	Facets	Input
Ang	Concurrent	4	Human computation
GoM	Concurrent	4	Human creator, human computation
aRD	Sequential	4	Human creator, human computation
DA	Sequential	3	Human creator, human computation
GF	Sequential	2	Embedded only
AiS	Concurrent	3	Human creator
Son	Sequential	3	Human creator, human computation
MM	Sequential	3	Embedded only
Ludi	Concurrent	3	Embedded only

An unusual example is the extensible graphical game generator [103], an automated programming system that generates playable user interfaces for games that are specified in a description language, where the interfaces respect features of the ruleset such as keeping hidden information hidden. In addition, for two-player games, it generates an AI player specialized to that game. This can be seen as a generative pipeline from rules to visuals and to gameplay.

Another system that couples the level design and audio facets is Audioverdrive [104], a side-scrolling space shooter for iOS that sets up a feedback loop between level design elements and a procedural audio system. For example, the height of the bottom terrain (level design facet) controls the pitch of the bass synth (audio facet), and in turn, treble sound events (audio facet) trigger the placement and timing of enemies (level design facet). Unlike the—similarly themed—AudioInSpace, the mapping between the two facets is controlled by the game's designer/composer, not by the player.

Some work in interactive narrative can also be viewed as performing facet orchestration. *Charbitat* [105], for example, has a pipeline that generates game worlds and then generates quests to fit them. Likewise, the General Mediation Engine [106] creates levels (as sequences of rooms) based on a narrative created via planning (which adapts to player actions while the game is played) and can also take some decisions regarding game rules (such as the presence of an inventory).

VIII. COMPARATIVE ANALYSIS OF THE CASE STUDIES

Table I summarizes how the case studies[4] of Section VII tackle the issue of facet orchestration in different ways. Along the dimensions considered, the order of generation is important as it is indicative of a more top-down approach (in the case of sequential generation using previous steps as scaffolds) or a bottom-up approach (in the case of content generated concurrently). Note, however, that order of generation does not necessarily mean that all content concurrently generated are actually orchestrated or checked for coherence as suggested in Section IV-B. The only truly bottom-up approach is Ludi,

[4]aRD: A Rogue Dream; GoM: Game-o-Matic; Son: Sonancia; GF: Game Forge; Ang: Angelina; MM: Mechanic Miner; DA: Data Adventures; AiS: AudioInSpace.

where the boards and rules are generated simultaneously and independently from each other,[5] with gameplay traces created based on policies customized for that specific board and rules. In contrast, Sonancia uses a clear sequential process as it first generates the frame of tension progression; then, the frame is unchanged, while a level is generated to match it; then, the level is unchanged, while a generator finds appropriate sounds for each room. As noted in Section VII-J, any generator with simulation-based evaluation, e.g., [59], has to generate the gameplay concurrently with another facet (usually level or rules); a sequential approach would finalize the level and the rules and then create controllers for that combination (which will remain unchanged until the end of generation).

In terms of facets being combined in the cases studied, most of the systems combine in a nontrivial way at least three facets. As noted, any simulation-based evaluation generates the gameplay facet, so, in that sense, two facets are relatively easy to orchestrate. GAME FORGE is noteworthy as it does not generate gameplay but instead focuses on the mapping between story and spatial arrangement of story elements. Among the most ambitious projects in terms of facets combined are A Rogue Dream, Angelina, and Game-O-Matic, although the role of facets such as narrative is subdued (a story's mood or a proto-narrative). Moreover, these three systems include fairly simple level generators, which do not consider any of the other facets and can thus operate concurrently without actual orchestration. On the other hand, Ludi only generates content for three facets, but it is the only instance of *complete game* generation, as the simple and abstract board games generated by Ludi do not require or benefit from visual, audio, or narrative generation. In terms of the types of facets often generated, among the nine case studies, the most popular is level design (eight cases) followed by narrative (six cases), although it should be noted again that the latter is used loosely, as it is also interpreted as story's mood (Angelina), entity relationships (Game-O-Matic), or progression of tension (Sonancia).

Finally, many of the surveyed systems include human or crowdsourced input as well as human intervention. Table I lists case studies, which make use of human input, online sources, or are only working on an internal knowledge model. Among projects that rely on user input, the differences are notable: A Rogue Dream requires only one word as a seed, while Game-o-matic requires the user to create a graph and name its nodes and edges. GAME FORGE and Sonancia do not require users to customize generative parameters, but it is likely that computer-provided parameterization may lead to unwanted results. AudioInSpace is the only case studied, which requires human intervention to affect the orchestration process, while the game is played. On the other hand, all projects that use online sources (except Sonancia) do so to retrieve images on specific keyword search queries. In addition to this, A Rogue Dream uses Google search autocomplete for mechanics and game objects, while Data Adventures bases most of its generation on open data repositories (OpenStreetMap, Wikipedia). In terms of systems that only orchestrate content internally generated, of special note is Mechanic Miner, as it discovers new rules by manipulating source code, and thus, the world model it uses is far more granular than in other systems.

[5]The fact that both facets are part of the same genotype is not relevant as genetic operators applied on rules do not affect board layouts and vice versa.

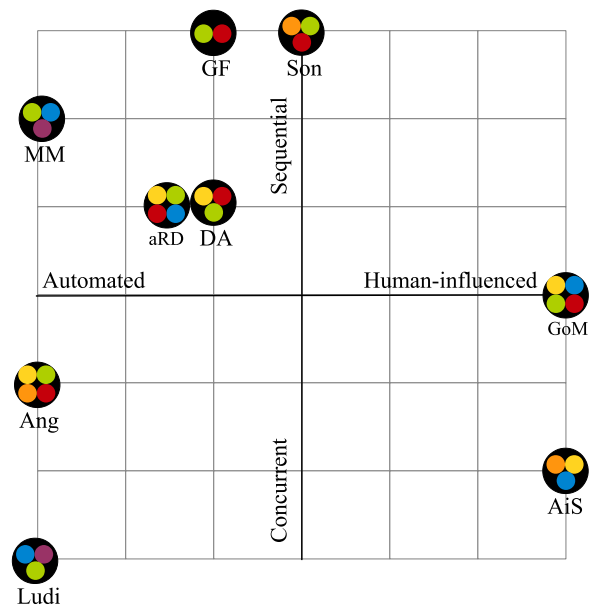


Fig. 16. Spectrum of the case studies on orchestration.

Fig. 16 shows where the case studies are found in terms of sequential (alluding to top-down) or concurrent (alluding to bottom-up) processes, as well as how much human customization goes into each. The latter conflates human input before generation (e.g., Game-O-Matic) and during generation (for AudioInSpace) and is assessed on the effort needed from a designer to initiate or guide the orchestration process. Fig. 16 clearly shows that there is much ground left unexplored on different dimensions and for most facet combinations.

IX. OPEN PROBLEMS

Section I already highlighted that orchestration is an interesting but hard problem. The analysis of how existing work in Section VII handles orchestration, and the dimensions of full orchestration discussed in Sections III, IV, and VI are important stepping stones. However, as in any *hard* problem, many and difficult challenges still need to be surpassed before orchestration can hope to achieve complete game generation.

A. Combinatorial Explosion

We have emphasized that in order to orchestrate several facets into a complete game, the individual generators must be highly controllable. Otherwise, we run into the problem of having lots of superfluous single generators with many interdependencies, which leads to a combinatorial explosion. Lightweight scripted approaches can work for a top-down process, where a number of generators that do one thing well (e.g., castle wall texture generation) could be chosen by the art director among an expansive set of such generators, as in Fig. 4. However, crafting all those generators would be a grand undertaking. Modular and

expressive generators, which, e.g., can customize their output based on top-down or cocreator directives, are much more desirable.

B. Learning the Mapping Between Facets

While Section IV focused on the architecture and generation sequence, at the core of orchestration is the challenge of assessing to which extent different content match. This task can be somehow simplified in top-down approaches, where a human-authored hierarchy of content per game type is feasible. In the example of Fig. 4, for example, the visuals director refines the labels in the frame, based on a human-provided grammar, e.g., warm visuals can be mapped to red or orange or purple, castles can be mapped to a series of walls, floors, doors, columns, etc. The task of mapping every possible game in this way in a database is certainly tedious, but genre conventions could be exploited to streamline the process. Since the top-down generative commands are one-way, it is not vital to verify that the final content match.

A more automated approach to this task could be via existing ontologies beyond games. Capturing the use of associations between entities as, for example, between action and agent/patient (e.g., *VerbNet*) or between a verb and its possible usage contexts (e.g., *FrameNet*) can be instrumental in deriving such mappings between facets. Similarly, semantic knowledge bridging game facets may be captured from large knowledge ontologies such as Wikipedia, from which, e.g., the structure or color palette of castles can be mined, or the fact that castles are tied to medieval times and narratives. Incipient work has been done on integrating existing lexical databases in a computational narrative context [5]; the resulting *GluNet*, however, could also be deployed to bridge knowledge between game facets as, e.g., narrative, level, and visuals.

On the other end of the spectrum, purely bottom-up approaches hinge on a coherence evaluation mechanism; this was already identified in Section IV-B as the main challenge of these approaches. In such cases, recent advances in machine learning could offer a solution. Deep learning has already shown its potential in purely visual tasks, but is also making good progress on finding associations between content from dissimilar domains such as text and images [107] or images and sound [108]. Given sufficient data, we can assume that a deep learning approach can assess whether game content match. For instance, deep learning was used to predict how levels and weapon parameters, combined, affect gameplay balance [109]; in [110], a similar mapping was used to adapt hand-authored or generated levels to be more balanced for specific matchups between character classes. It has been already shown that machine learning can capture level structures from gameplay videos [52], which can drive the generation of new levels. Mapping the actual visual output (in terms of color palette or structures) on top of the discovered level structure could provide a simple coherence evaluation for a specific game. Beyond that, many games could be used from a variety of "Let's play" videos on *YouTube* or e-sports streaming on *Twitch*. This would allow a more general model to be trained to assess consistency across games. Although far from straightforward, there is sufficient data available today and sufficiently sophisticated algorithms for machine learning to make the claim that machine learning-based orchestration is possible [111]. Even with limited data, decision trees were able to find mappings between color and Pokémon types in [112], and statistical models could match a pawn's shape with its in-game importance [113]. However, some facets will be easier for coherence evaluation in this way than others. For example, visuals, sounds, and even levels are straightforward inputs to a deep learning network, whereas narrative and game rules are far less so. Moreover, game rules may need to be translated into machine-readable input by a human designer, as there are no explicit descriptions for them in most of the cases.

C. Orchestration With Human Designers

Section V highlighted that a human can directly interact with the orchestration process (e.g., via interactive evolution), while Section VI highlighted that the final output could be a canvas for designers to work on. It is interesting, however, to further hypothesize how content generators could work along a human development team to create games from start to finish. Questions of coherence evaluation can be trivially answered by having a human expert verifying which elements match and possibly manually removing others from a blackboard [69]. Rather than treating designers as input (for parameter tweaks) or intervention for verification, it is more interesting to consider designers in other roles in the proposed frameworks. Can designers work alongside generators, not as a composer that dictates the frame of the generated game as in Fig. 4, but instead jam with generators in a bottom-up approach? Generated content has already been used as a seed for human creativity during a creative process [26], [88], [114], and algorithmic expressiveness can be used to broaden the creative potential of casual creators [115]. However, game design where the computer is providing creative suggestions in domains dissimilar to the domain the human designer is working on (e.g., suggesting visual art while the human designer is editing rulesets) is largely unexplored. The closest attempt to such a human-inclusive orchestration is likely the game sprite recommender of [116], where similarity to game rules edited by a human user is used to find recommendations for sprites (including their rules and visuals). However, more computationally creative approaches, e.g., where the computer provides the frame or where more game facets are combined, could lead to breakthroughs in AI-assisted orchestration.

D. Evaluating Orchestration

One of the core challenges of any computationally creative approach is assessing its performance and its creativity. The task becomes more difficult when generation is on aesthetic domains such as game art and music, and it is likely a reason for a stronger research focus on the more quantifiable facets of level generation and ruleset generation. Unfortunately, if evaluating one such domain on its own is already difficult, evaluating the orchestration of several domains is a far greater challenge. Such evaluation may be subjective and likely qualitative, while the results that are shown in a publication may be curated. The conclusions drawn

from such work may be subject to human biases by each author, reviewer, and/or reader. Admittedly, identifying which facets were mapped in each of the case studies of Section VII—as well as deciding which facets were considered too trivial a contribution, and even the relative size of each facet in the figures—was based on personal intuition and, ultimately, opinion.

There is no easy solution to this problem, and the best way to move forward for orchestration research is to look at domains which have struggled with such issues already. As noted above, the research community around *computational creativity* has faced similar issues for several years, and many frameworks have been proposed for assessing creativity [65], [68], [117]. Even today the topic of trivial versus creative generation is pertinent [57] and is reminiscent of our discussion on trivial generation of gameplay in Section VII-J.

Rather than question the creativity of their systems, *interactive narrative* research has struggled with how the authored narrative components impact the user's experience. In interactive narrative, "the dynamism of the content makes authoring for these goals a challenge" [118]. Similarly, in games, it is unknown—and largely untraceable—how players perceive (in their own gameplay experience) the world and the interaction between game mechanics. As with stories, the "readability" of the game environment, the points of visual or aural interest, and the use of mechanics to overcome challenges can all be presumed by a human or computational designer, but remain difficult to benchmark based on human play.

Finally, evaluation is a contentious topic in *mixed-initiative co-creative systems* [119], where the system provides suggestions to human users. Whether suggestions are selected, which patterns the selected suggestions have (over suggestions that are not selected, for instance), and at which point in the creative process they were selected can be monitored and reported in a quantitative manner [88], [119], [120]. However, assessing whether such suggestions are of value to users as creative stimuli is difficult, as users may not explicitly select a suggestion received, but instead be simply inspired by a suggestion to change their design patterns. Similarly, users may not necessarily correctly perceive at which point in the creative process they were influenced, although this could be assessed by an audience of human experts instead [119]. The challenge is similar for AI-based orchestration, as a system can randomly generate content (e.g., in a bottom-up approach) and only an audience (of, e.g., human experts) could assess whether at any point in the process, there was a creative breakthrough when pieces (i.e., content of different facets) fall into place and become more than a sum of parts.

X. Conclusion

This paper analyzed the current state of the art in game orchestration of different generative systems for multiple game facets such as audio, visuals, levels, rules, narrative, and gameplay. The topic was discussed along the dimensions of how, from where, and for whom orchestration takes place, and for what types of content. We provided several suggestions regarding the relationships between different game facets and envisioned several high-level orchestration processes along a spectrum between purely hierarchical top-down generation and organic bottom-up generation. Nine case studies were presented and compared along the above dimensions of orchestration. While this paper does not answer many of the questions it poses, it aims to create a roadmap so that orchestration of game generation can be more systematically and more ambitiously explored in the coming years.

Acknowledgment

This paper extends the work performed by the same authors during the 2015 Dagstuhl seminar 15051 on "Artificial and Computational Intelligence in Games: Integration." A preliminary report titled "Creativity Facet Orchestration: the Whys and the Hows" was included in [121].

References

[1] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. New York, NY, USA: Springer, 2016.

[2] A. Liapis, G. N. Yannakakis, and J. Togelius, "Towards a generic method of evaluating game levels," in *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entertainment*, 2013, pp. 30–36.

[3] G. N. Yannakakis and J. Togelius, *Artificial Intelligence and Games*. New York, NY, USA: Springer, 2018.

[4] A. Liapis, G. N. Yannakakis, and J. Togelius, "Computational game creativity," in *Proc. Int. Conf. Comput. Creativity*, 2014.

[5] B. Kybartas and R. Bidarra, "A semantic foundation for mixed-initiative computational storytelling," in *Proc. 8th Int. Conf. Interact. Digit. Storytelling*, 2015, pp. 162–169.

[6] A. Järvinen, "Gran Stylissimo: The audiovisual elements and styles in computer and video games," in *Proc. Comput. Games Digit. Cultures Conf.*, 2002, pp. 113–128.

[7] A. Lagae *et al.*, "State of the art in procedural noise functions," in *Proc. Eurographics 2010 State of the Art Reports*, 2010.

[8] K. Perlin, "An image synthesizer," in *Proc. 12th Annu. Conf. Comput. Graph. Interact. Techn.*, 1985, pp. 287–296.

[9] G. J. de Carpentier and R. Bidarra, "Interactive GPU-based procedural heightfield brushes," in *Proc. Int. Conf. Found. Digit. Games*, 2009, pp. 55–62.

[10] R. M. Smelik, T. Tutenel, R. Bidarra, and B. Benes, "A survey on procedural modeling for virtual worlds," *Comput. Graph. Forum*, vol. 33, no. 6, pp. 31–50, 2014.

[11] A. Howlett, S. Colton, and C. Browne, "Evolving pixel shaders for the prototype video game subversion," in *Proc. AISB Symp. AI Games*, 2010.

[12] T. Tutenel, R. van der Linden, M. Kraus, B. Bollen, and R. Bidarra, "Procedural filters for customization of virtual worlds," in *Proc. FDG Workshop Procedural Content Gener.*, 2011.

[13] A. Liapis, "Exploring the visual styles of arcade game assets," in *Evolutionary and Biologically Inspired Music, Sound, Art and Design* (ser. Lecture Notes in Computer Science), vol. 9596. New York, NY, USA: Springer, 2016.

[14] K. Collins, *Playing With Sound: A Theory of Interacting With Sound and Music in Video Games*. Cambridge, MA, USA: MIT Press, 2013.

[15] P. Lopes, A. Liapis, and G. N. Yannakakis, "Targeting horror via level and soundscape generation," in *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entertainment*, 2015.

[16] M. Scirea, B. C. Bae, Y.-G. Cheong, and M. Nelson, "Evaluating musical foreshadowing of videogame narrative experiences," in *Proc. Audio Mostly: Conf. Interact. Sound*, 2014, Art. no. 8.

[17] K. Collins, "An introduction to procedural music in video games," *Contemporary Music Rev.*, vol. 28, no. 1, pp. 5–15, 2009.

[18] Y.-G. Cheong, M. O. Riedl, B.-C. Bae, and M. J. Nelson, "Planning with applications to quests and story," in, *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. New York, NY, USA: Springer, 2016.

[19] M. Mateas and A. Stern, "Procedural authorship: A case-study of the interactive drama Façade," in *Proc. Digit. Arts Culture*, 2005.

[20] J. McCoy, M. Treanor, B. Samuel, A. A. Reed, M. Mateas, and N. Wardrip-Fruin, "Prom Week: Designing past the game/story dilemma," in *Proc. Int. Conf. Found. Digit. Games*, 2013.

[21] B. Kybartas and R. Bidarra, "A survey on story generation techniques for authoring computational narratives," *IEEE Trans. Comput. Intell. AI Games*, vol. 9, no. 3, pp. 239–253, Sep. 2017.

[22] N. Shaker, A. Liapis, J. Togelius, R. Lopes, and R. Bidarra, "Constructive generation methods for dungeons and levels," in *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. New York, NY, USA: Springer, 2016, pp. 31–55.

[23] J. Dormans and S. C. J. Bakkes, "Generating missions and spaces for adaptable play experiences," *IEEE Trans. Comput. Intell. AI Games*, vol. 3, no. 3, pp. 216–228, Sep. 2011.

[24] J. Togelius, G. Yannakakis, K. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," *IEEE Trans. Comput. Intell. AI Games*, vol. 3, no. 3, pp. 172–186, Sep. 2011.

[25] R. M. Smelik, T. Tutenel, K. J. de Kraker, and R. Bidarra, "A declarative approach to procedural modeling of virtual worlds," *Comput. Graph.*, vol. 35, no. 2, pp. 352–363, 2011.

[26] G. Smith, J. Whitehead, and M. Mateas, "Tanagra: Reactive planning and constraint solving for mixed-initiative level design," *IEEE Trans. Comput. Intell. AI Games*, vol. 3, no. 3, pp. 201–215, Sep. 2011.

[27] M. Sicart, "Defining game mechanics," *Game Studies*, vol. 8, 2008.

[28] A. Järvinen, "Games without frontiers: Theories and methods for game studies and design," Ph.D. dissertation, Univ. Tampere, Tampere, Finland, 2008.

[29] J. Togelius, M. J. Nelson, and A. Liapis, "Characteristics of generatable games," in *Proc. FDG Workshop Procedural Content Gener.*, 2014.

[30] C. Browne and F. Maire, "Evolutionary game design," *IEEE Trans. Comput. Intell. AI Games*, vol. 2, no. 1, pp. 1–16, Mar. 2010.

[31] C. Browne, *Evolutionary Game Design*. New York, NY, USA: Springer, 2011.

[32] A. M. Smith and M. Mateas, "Variations Forever: Flexibly generating rulesets from a sculptable design space of mini-games," in *Proc. IEEE Symp. Comput. Intell. Games*, 2010, pp. 273–280.

[33] J. Togelius and J. Schmidhuber, "An experiment in automatic game design," in *Proc. IEEE Symp. Comput. Intell. Games*, 2008, pp. 111–118.

[34] M. Sicart, "Digital games as ethical technologies," in *The Philosophy of Computer Games*. New York, NY, USA: Springer, 2012, pp. 101–124.

[35] R. Hunicke, M. Leblanc, and R. Zubek, "MDA: A formal approach to game design and game research," in *Proc. AAAI Workshop Challenges Games AI*, 2004.

[36] M. Carter, M. Gibbs, and M. Harrop, "Metagames, paragames and orthogames: A new vocabulary," in *Proc. Int. Conf. Found. Digit. Games*, 2012.

[37] P. Ekman, "Emotional and conversational nonverbal signals," in *Proc. 6th Int. Colloq. Cogn. Sci.*, 2004.

[38] J. Laird and M. V. Lent, "Human-level AI's killer application: Interactive computer games," *AI Mag.*, vol. 22, 2001.

[39] D. Perez, J. Togelius, S. Samothrakis, P. Rohlfshagen, and S. M. Lucas, "Automated map generation for the physical traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 18, no. 5, pp. 708–720, Oct. 2014.

[40] R. Bartle, "Hearts, clubs, diamonds, spades: Players who suit MUDs," *J. MUD Res.*, vol. 1, no. 1, 1996.

[41] N. van Hoorn, D. Wierstra, J. Togelius, and J. Schmidhuber, "Robust player imitation using multiobjective evolution," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 652–659.

[42] C. Holmgård, A. Liapis, J. Togelius, and G. N. Yannakakis, "Evolving personas for player decision modeling," in *Proc. IEEE Conf. Comput. Intell. Games*, 2014, pp. 1–4.

[43] K. Hartsook, A. Zook, S. Das, and M. O. Riedl, "Toward supporting stories with procedurally generated game worlds," in *Proc. IEEE Conf. Comput. Intell. Games*, 2011, pp. 297–304.

[44] A. Liapis, G. N. Yannakakis, and J. Togelius, "Adapting models of visual aesthetics for personalized content creation," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 3, pp. 213–228, Sep. 2012.

[45] M. Cook and S. Colton, "A Rogue Dream: Automatically generating meaningful content for games," in *Proc. AIIDE Workshop Exp. AI Games*, 2014.

[46] G. A. B. Barros, M. C. Green, A. Liapis, and J. Togelius, "Who killed Albert Einstein? from open data to murder mystery games," *IEEE Trans. Games*, 2018, accepted for publication.

[47] M. Treanor, B. Blackford, M. Mateas, and I. Bogost, "Game-O-Matic: Generating videogames that represent ideas," in *Proc. FDG Workshop Procedural Content Gener.*, 2012.

[48] D. Karavolos, A. Bouwer, and R. Bidarra, "Mixed-initiative design of game levels: Integrating mission and space into level generation," in *Proc. Int. Conf. Found. Digit. Games*, 2015.

[49] T. Veale, "From conceptual "mash-ups" to "bad-ass" blends: A robust computational model of conceptual blending," in *Proc. Int. Conf. Comput. Creativity*, 2012, pp. 1–8.

[50] T. Veale, "Coming good and breaking bad: Generating transformative character arcs for use in compelling stories," in *Proc. Int. Conf. Comput. Creativity*, 2014.

[51] M. Cook and S. Colton, "Automated collage generation—With more intent," in *Proc. Int. Conf. Comput. Creativity*, 2011.

[52] M. Guzdial and M. Riedl, "Toward game level generation from gameplay videos," in *Proc. Int. Conf. Found. Digit. Games*, 2016.

[53] A. Summerville and M. Mateas, "Sampling Hyrule: multi-technique probabilistic level generation for action role playing games," in *Proc. AIIDE Workshop Exp. AI Games*, 2015.

[54] M. G. Friberger *et al.*, "Data games," in *Proc. FDG Workshop Procedural Content Gener.*, 2013.

[55] G. A. B. Barros, A. Liapis, and J. Togelius, "Playing with data: Procedural generation of adventures from open data," in *Proc. Int. Joint Conf. DiGRA FDG*, 2016.

[56] M. Cook, S. Colton, and A. Pease, "Aesthetic considerations for automated platformer design," in *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entertainment*, 2012.

[57] D. Ventura, "Mere generation: Essential barometer or dated concept?" in *Proc. Int. Conf. Comput. Creativity*, 2016.

[58] M. Riedl and R. M. Young, "Narrative planning: Balancing plot and character," *J. Artif. Intell. Res.*, vol. 39, pp. 217–268, 2014.

[59] A. Liapis, C. Holmgård, G. N. Yannakakis, and J. Togelius, "Procedural personas as critics for dungeon generation," in *Proc. Eur. Conf. Appl. Evol. Comput.*, 2015, pp. 331–343.

[60] P. Lopes, A. Liapis, and G. N. Yannakakis, "Modelling affect for horror soundscapes," *IEEE Trans. Affect. Comput.*, 2017, to be published.

[61] W. Royce, "Managing the development of large software systems," in *Proc. IEEE WESCON*, 1970, pp. 1–9.

[62] D. Horowitz, "Representing musical knowledge in a jazz improvisation system," in *Proc. Workshop Artif. Intell. Music*, 1995.

[63] P. Lopes, A. Liapis, and G. N. Yannakakis, "Framing tension for game generation," in *Proc. Int. Conf. Comput. Creativity*, 2016.

[64] T. Scaltsas and C. Alexopoulos, "Creating creativity through emotive thinking," in *Proc. World Congr. Philosophy*, 2013.

[65] S. Colton, J. Charnley, and A. Pease, "Computational creativity theory: The FACE and IDEA descriptive models," in *Proc. Int. Conf. Comput. Creativity*, 2011.

[66] M. O. Riedl and A. Zook, "AI for game production," in *Proc. IEEE Conf. Comput. Intell. Games*, 2013, pp. 1–8.

[67] A. Zook and M. O. Riedl, "Game conceptualization and development processes in the global game jam," in *Proc. FDG Workshop Global Game Jam*, 2013.

[68] G. Ritchie, "Some empirical criteria for attributing creativity to a computer program," *Minds Mach.*, vol. 17, pp. 76–99, 2007.

[69] R. Engelmore and T. Morgan, *Blackboard Systems*. Reading, MA, USA: Addison-Wesley, 1988.

[70] H. P. Nii, "The blackboard model of problem solving and the evolution of blackboard architectures," *AI Mag.*, vol. 7, no. 2, pp. 38–53, 1986.

[71] D. D. Corkill, K. Q. Gallagher, and K. E. Murray, "GBB: A generic blackboard development system," in *Proc. Nat. Conf. Artif. Intell.*, 1986, pp. 1008–1014.

[72] B. Hayes-Roth, "A blackboard architecture for control," *Artif. Intell.*, vol. 26, no. 3, pp. 251–321, 1985.

[73] D. R. Hofstadter and M. Mitchell, "The Copycat project: A model of mental fluidity and analogy-making," in *Fluid Concepts and Creative Analogies*. New York, NY, USA: Basic Books, 1995, pp. 205–267.

[74] D. R. Hofstadter and G. McGraw, "Letter Spirit: Esthetic perception and creative play in the rich microcosm of the Roman alphabet," in *Fluid Concepts and Creative Analogies*. New York, NY, USA: Basic Books, 1995, pp. 407–466.

[75] A. Liapis, "Multi-segment evolution of dungeon game levels," in *Proc. Genetic Evol. Comput. Conf.*, 2017, pp. 203–210.

[76] R. Jain, A. Isaksen, C. Holmgård, and J. Togelius, "Autoencoders for level generation, repair, and recognition," in *Proc. ICCC Workshop Comput. Creativity Games*, 2016.

[77] H. Takagi, "Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation," *Proc. IEEE*, vol. 89, no. 9, pp. 1275–1296, Sep. 2001.
[78] L. Cardamone, D. Loiacono, and P. L. Lanzi, "Interactive evolution for the procedural generation of tracks in a high-end racing game," in *Proc. Genetic Evol. Comput. Conf.*, 2011, pp. 395–402.
[79] S. Risi, J. Lehman, D. B. D'Ambrosio, R. Hall, and K. O. Stanley, "Petalz: Search-based procedural content generation for the casual gamer," *IEEE Trans. Comput. Intell. AI Games*, vol. 8, no. 3, pp. 244–255, Sep. 2016.
[80] L. Cardamone, G. N. Yannakakis, J. Togelius, and P. L. Lanzi, "Evolving interesting maps for a first person shooter," in *Proc. Appl. Evol. Comput.*, 2011, pp. 63–72.
[81] E. J. Hastings, R. K. Guha, and K. O. Stanley, "Automatic content generation in the Galactic Arms Race video game," *IEEE Trans. Comput. Intell. AI Games*, vol. 1, no. 4, pp. 245–263, Dec. 2009.
[82] A. K. Hoover, W. Cachia, A. Liapis, and G. N. Yannakakis, "AudioInSpace: exploring the creative fusion of generative audio, visuals and gameplay, " in, *Evolutionary and Biologically Inspired Music, Sound, Art and Design*. New York, NY, USA: Springer, 2015.
[83] A. Liapis, G. N. Yannakakis, and J. Togelius, "Designer modeling for personalized game content creation tools," in *Proc. AIIDE Workshop Artif. Intell. Game Aesthetics*, 2013.
[84] A. Liapis, G. N. Yannakakis, and J. Togelius, "Designer modeling for Sentient Sketchbook," in *Proc. IEEE Conf. Comput. Intell. Games*, 2014, pp. 1–8.
[85] N. Shaker, G. N. Yannakakis, and J. Togelius, "Towards automatic personalized content generation for platform games," in *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entertainment*, 2010.
[86] N. Shaker *et al.*, "The 2010 Mario AI championship: Level generation track," *IEEE Trans. Comput. Intell. AI Games*, vol. 3, no. 4, pp. 332–347, Dec. 2011.
[87] A. Hamilton, "Inside DICE: Creating the beautiful environments in Battlefield 3: Armored Kill," 2012. [Online]. Available: https://www.ea.com/en-gb/news/battlefield-3-armored-kill-environments
[88] A. Liapis, G. N. Yannakakis, and J. Togelius, "Sentient Sketchbook: Computer-aided game level authoring," in *Proc. Int. Conf. Found. Digit. Games*, 2013.
[89] M. Treanor, "Investigating procedural expression and interpretation in videogames," Ph.D. dissertation, , Univ. California, Santa Cruz, CA, USA, 2013.
[90] M. Treanor, B. Schweizer, I. Bogost, and M. Mateas, "The micro-rhetorics of Game-o-Matic," in *Proc. Int. Conf. Found. Digit. Games*, 2012.
[91] M. C. Green, G. A. B. Barros, A. Liapis, and J. Togelius, "DATA agent," in *Proc. 13th Conf. Found. Digit. Games*, 2018.
[92] K. O. Stanley, "Exploiting regularity without development," in *Proc. AAAI Fall Symp. Develop. Syst.*, 2006.
[93] P. Lopes, A. Liapis, and G. N. Yannakakis, "Sonancia: A multi-faceted generator for horror," in *Proc. IEEE Conf. Comput. Intell. Games*, 2016, pp. 1–2.
[94] M. Cook and S. Colton, "A Puzzling Present: Code modification for game mechanic design," in *Proc. Int. Conf. Comput. Creativity*, 2013.
[95] M. Cook, S. Colton, A. Raad, and J. Gow, "Mechanic Miner: Reflection-driven game mechanic discovery and level design," in *Proc Eur. Conf. Appl. Evol. Comput.*, 2012, pp. 284–293.
[96] N. Sorenson, P. Pasquier, and S. DiPaola, "A generic approach to challenge modeling for the procedural creation of video game levels," *IEEE Trans. Comput. Intell. AI Games*, vol. 3, no. 3, pp. 229–244, Sep. 2011.
[97] A. Isaksen, D. Gopstein, J. Togelius, and A. Nealen, "Exploring game space of minimal action games via parameter tuning and survival analysis," *IEEE Trans. Games*, vol. 10, no. 2, pp. 182–194, Jun. 2018.
[98] A. M. Smith, E. Andersen, M. Mateas, and Z. Popović, "A case study of expressively constrainable level design automation tools for a puzzle game," in *Proc. Int. Conf. Found. Digit. Games*, 2012.
[99] M. Cook, "Would you look at that! Vision-driven procedural level design," in *Proc. AIIDE Workshop Exp. AI Games*, 2015.
[100] J. Togelius, R. De Nardi, and S. M. Lucas, "Towards automatic personalised content creation for racing games," in *Proc. IEEE Symp. Comput. Intell. Games*, 2007, pp. 252–259.
[101] M. J. Nelson and M. Mateas, "Towards automated game design," in *Proc. Cong. Italian Assoc. Artif. Intell.*, 2007, pp. 626–637.
[102] M. J. Nelson and M. Mateas, "An interactive game-design assistant," in *Proc. 13th Int. Conf. Intell. User Interfaces*, 2008.
[103] J. Orwant, "EGGG: Automated programming for game generation," *IBM Syst. J.*, vol. 39, no. 3.4, pp. 782–794, 2000.
[104] N. I. Holtar, M. J. Nelson, and J. Togelius, "Audioverdrive: Exploring bidirectional communication between music and gameplay," in *Proc. Int. Comput. Music Conf.*, 2013, pp. 124–131.
[105] C. Ashmore and M. Nitsche, "The quest in a generated world," in *Proc. DiGRA Conf.*, 2007, pp. 503–509.
[106] J. Robertson and R. M. Young, "Automated gameplay generation from declarative world representations," in *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entertainment*, 2015.
[107] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3128–3137.
[108] Y. Aytar, C. Vondrick, and A. Torralba, "SoundNet: Learning sound representations from unlabeled video," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016.
[109] D. Karavolos, A. Liapis, and G. N. Yannakakis, "Learning the patterns of balance in a multi-player shooter game," in *Proc. 12th Int. Conf. Found. Digit. Games*, 2017, Art. no. 70.
[110] D. Karavolos, A. Liapis, and G. N. Yannakakis, "Using a surrogate model of gameplay for automated level design," in *Proc. IEEE Conf. Comput. Intell. Games*, 2018.
[111] A. Summerville *et al.*, "Procedural content generation via machine learning (PCGML)," *IEEE Trans. Games*, vol. 10, no. 3, pp. 257–270, Sep. 2018.
[112] A. Liapis, "Recomposing the Pokémon color palette," in *Proc. Int. Conf. Appl. Evol. Computat.*, 2018, pp. 308–324.
[113] J. Kowalski, A. Liapis, and Ł. Żarczyński, "Mapping chess aesthetics onto procedurally generated chess-like games," in *Proc. Int. Conf. Appl. Evol. Comput.*, 2018, pp. 325–341.
[114] P. Lucas and C. Martinho, "Stay awhile and listen to 3Buddy, a co-creative level design support tool," in *Proc. Int. Conf. Comput. Creativity*, 2017.
[115] K. Compton and M. Mateas, "Casual creators," in *Proc. Int. Conf. Comput. Creativity*, 2015.
[116] T. Machado, I. Bravi, Z. Wang, A. Nealen, and J. Togelius, "Shopping for game mechanics," in *Proc. FDG Workshop Procedural Content Gener.*, 2016.
[117] S. Colton, "Creativity versus the perception of creativity in computational systems," in *Proc. AAAI Spring Symp. Creative Intell. Syst.*, 2008.
[118] B. Samuel, J. McCoy, M. Treanor, A. A. Reed, M. Mateas, and N. Wardrip-Fruin, "Introducing story sampling: Preliminary results of a new interactive narrative evaluation technique," in *Proc. Int. Conf. Found. Digit. Games*, 2014.
[119] G. N. Yannakakis, A. Liapis, and C. Alexopoulos, "Mixed-initiative co-creativity," in *Proc. Int. Conf. Found. Digit. Games*, 2014.
[120] A. Liapis, "Mixed-initiative creative drawing with webIconoscope," in *Computational Intelligence in Music, Sound, Art and Design* (ser. Lecture Notes in Computer Science), vol. 10198. New York, NY, USA: Springer, 2017.
[121] S. M. Lucas, M. Mateas, M. Preuss, P. Spronck, and J. Togelius, "Artificial and Computational Intelligence in Games: Integration (Dagstuhl Seminar 15051)," *Dagstuhl Rep.*, vol. 5, no. 1, pp. 207–242, 2015.

Antonios Liapis received the Ph.D. degree in information technology from the IT University of Copenhagen, Copenhagen, Denmark, in 2014.

He is currently a Lecturer with the Institute of Digital Games, University of Malta, Msida, Malta, where he bridges the gap between game technology and game design in courses focusing on human–computer creativity, digital prototyping, and game development. His research focuses on artificial intelligence as an autonomous creator or as a facilitator of human creativity. His work includes computationally intelligent tools for game design, as well as computational creators that blend semantics, visuals, sound, plot, and level structure to create horror games, adventure games, and more.

Dr. Liapis is the General Chair of the EvoMusArt conference (2018–2019) and has served as the Local Chair (2016) and Demonstrations Chair (2019) of the Computational Intelligence and Games Conference. He is an Associate Editor for the IEEE TRANSACTIONS ON GAMES. He has coorganized many workshops in a diverse set of conferences. He has received several awards for his research contributions and reviewing effort.

Georgios N. Yannakakis (SM'14) received the Ph.D. degree in informatics from the University of Edinburgh, Edinburgh, U.K., in 2006.

He is currently a Professor and Director of the Institute of Digital Games, University of Malta, Msida, Malta. In 2012, he was an Associate Professor with the Center for Computer Games Research, IT University of Copenhagen. He has authored or coauthored more than 220 papers and his work has been cited broadly. His research has been supported by numerous national and European grants (including a Marie Skłodowska-Curie Fellowship) and has appeared in *Science Magazine* and *New Scientist* among other venues. His research interests include artificial intelligence, computational creativity, affective computing, advanced game technology, and human–computer interaction.

Dr. Yannakakis is an Associate Editor for the IEEE TRANSACTIONS ON GAMES. He was an Associate Editor for the IEEE TRANSACTIONS ON AFFECTIVE COMPUTING and the IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES. He was the General Chair of key conferences in the area of game artificial intelligence (2010 IEEE Conference on Computational Intelligence and Games) and games research (2013 International Conference on the Foundations of Digital Games). He is the recipient of the IEEE TRANSACTIONS ON AFFECTIVE COMPUTING Most Influential Paper Award and the IEEE TRANSACTIONS ON GAMES Outstanding Paper Award.

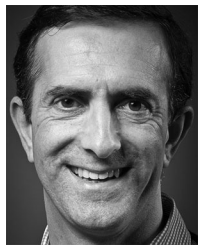
Mark J. Nelson received the Ph.D. degree in computer science from the Georgia Institute of Technology, Atlanta, GA, USA, in 2015.

He is Senior Research Fellow with the MetaMakers Institute, Games Academy, Falmouth University, Falmouth, U.K. Prior to joining Falmouth University, he was with the Center for Computer Games Research, IT University of Copenhagen, and before that, with the Expressive Intelligence Studio, University of California Santa Cruz. His research spans a range of topics in artificial intelligence and games, including technical research on automated gameplay and analysis, design-support work using artificial intelligence to empower both professional and novice designers, and conceptual work on formal models of games' mechanical and meaning-making elements.

Mike Preuss received the Ph.D. degree in computer science from the Technical University of Dortmund, Dortmund, Germany, in 2013.

He is currently an Research Associate with the European Research Center for Information Systems, University of Münster, Münster, Germany. Previously, he was with the Chair of Algorithm Engineering, Technical University of Dortmund. Since 2016, he has been involved in the PropStop project that deals with the detection of propaganda on social media. His research interests focus on the field of evolutionary algorithms for real-valued problems, namely on multimodal and multiobjective optimization and on computational intelligence methods for computer games, especially in procedural content generation and real-time strategy games. He is also interested in applying the game artificial intelligence techniques to engineering problems, e.g., chemical retrosynthesis.

Dr. Preuss is an Associate Editor for the IEEE TRANSACTIONS ON GAMES and an Advisory Board Member of Springer's *Natural Computing* book series and has been part of the organizational team of several conferences of the last years, in various functions, as a General co-Chair, Proceedings Chair, Competition Chair, Workshops Chair, notably also as the PC co-Chair for the 2018 Computational Intelligence and Games Conference.



Rafael Bidarra received the Graduate degree in electronics engineering from the University of Coimbra, Coimbra, Portugal, in 1987, and the Ph.D. degree in computer science from the Delft University of Technology, The Netherlands, in 1999.

He is currently an Associate Professor of game technology with the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology. He leads the research laboratory on game technology with the Computer Graphics and Visualization Group. He has numerous publications in international journals and conference proceedings, integrates the editorial board of several journals, and has served in many conference program committees. His current research interests include procedural and semantic modeling techniques for the specification and generation of both virtual worlds and gameplay; serious gaming; and game adaptivity and interpretation mechanisms for in-game data.