# Artificial Intelligence for Designing Games

Antonios Liapis

**Abstract** Making a game requires a diverse set of skills and talents: visual design, sound design, level design and narrative design require developers to be creative within their own domain but also to consider how their creations impact other facets of the game. Since games are interactive media, however, players also exercise their own creativity in order to overcome challenges introduced by the game's designers or by other players in multi-player settings. This chapter will analyze games under the prism of the six facets of creativity which go into the play experience: visuals, audio, narrative, levels, rules, and gameplay. Important examples of human creativity in commercial games will be highlighted for each facet, as well as how artificial intelligence has been applied to generate content for each facet. The chapter concludes with a discussion on how an artificially intelligent system can facilitate the collaboration of different generators, and possibly human designers and artists.

## 1 Introduction

Over the last five decades, digital games have become a mainstay source of entertainment for millions. Games are now available for a broad variety of devices including personal computers, dedicated game consoles, mobile devices, social media, virtual reality headsets and much more. As digital games are becoming ubiquitous, the demographics of players continuously shift: in the United States, for instance, 45% of gamers are women while more than a quarter are aged 50 years or older [30]. The massive number of potential players, and the breadth of interests they may have, has pushed the game industry towards diversifying the themes in games as well as the interaction paradigms: from massively-multiplayer online games which can be played over a period of many hours, to simple-to-grasp puzzle games played within

Antonios Liapis
Institute of Digital Games, University of Malta, Msida, Malta, e-mail: antonios.liapis@um.edu.mt

minutes on a mobile device. The massive market for games (the US video game industry generates more than $30 billion a year [30]) has incentivized many aspiring game developers to join large game studios with thousands of employees, to make games on their own or with a small group of friends, and anything in-between. As more accessible tools for game development become available, such as the *Unity* game engine [184], the demographics of game developers similarly shift: today, it is easier than ever to create a game with minimal technical knowledge. This has led to a huge variety of games designed by — and for — a very diverse set of people.

While making a game is now easier than ever in terms of technical competence, the creativity required from game designers can not be understated. A game's visuals, soundscape and plot structure are reminiscent of traditional art forms such as painting and sculpting, music and storytelling. While confined to an art direction that lends itself well to an interactive digital medium (for instance, game narrative follows specific themes and presumes that it will be exposed in a non-linear fashion), game artists and writers face similar challenges as their peers in traditional arts. Games, however, are interactive media tailored towards entertainment and prolonged engagement. Therefore, other dimensions of the creativity going into game development is the design of game levels and game rules. Finally, end-users also insert their own creativity when playing the game in order to overcome challenges introduced by the game's designers or by other players in multi-player settings.

Creativity in game design and development therefore must be introduced into several different facets of the game experience, and at different times — from concept development to interaction with the final artifact. Traditionally, these creative roles were undertaken by teams of human game developers, designers and artists. However, time and budget constraints as well as the desire to innovate has led commercial studios to offload some content design and development efforts to algorithms. Currently, procedural content generation (PCG) is often a selling point for games, especially promoting the benefits of perpetually fresh and unique game content such as new levels or enemies in every playthrough. One of the most ambitious games which largely relied on PCG for its promotion and marketing was *No Man's Sky* [161], with trailers preceded with the text "Every atom procedural". The appeal of a vast galaxy that *No Man's Sky* could produce thanks to PCG led its small team of developers to enjoy a large commercial success, with the game being the second-highest selling game in North America by revenue in the month of its release. Similarly, negative criticisms raised towards *No Man's Sky* were largely towards the expressivity of its generators [89] (with new planets not being meaningfully different than previously visited ones) as well as more mundane issues such as the lack of multi-player interactions.

Beyond commercial games, there has been a growing academic interest in artificial intelligence (AI) for generating game content for over a decade [108]. Evolutionary computation, machine learning, constraint programming and a plethora of other approaches have been applied to generate game content of different facets, from game levels to visuals and plot structure. In the process of such research, many important findings regarding design patterns, evaluations of game quality, and good design practice for specific types of games have surfaced, enriching our knowledge

of game design. One of the core aims of PCG research is the design of complete games by AI; to achieve this, algorithms must show skill in creating quality content in one or more facets. Additionally, algorithms must assess how harmonious the entire game outcome is, i.e. *orchestrate* different generative algorithms [73].

This chapter will analyze games under the prism of the different facets of creativity which go into the play experience. In Section 2, six facets are identified (visuals, audio, narrative, levels, rules, and gameplay) and important examples of human creativity in commercial games are highlighted for each facet. Section 3 highlights how the different facets must be orchestrated in order to produce a cohesive whole. Section 4 provides an overview of generative algorithms in commercial games, while Section 5 focuses on generative algorithms broadly based on artificial intelligence principles. Algorithms and key examples of AI-based content generation for specific types of content (usually in one facet) are described in 5.1, while approaches for AI-based orchestration are described in 5.2 and important cases of full or partial game generation are highlighted in 5.3. The chapter concludes in Section 6 with key stepping stones and hurdles on the road ahead towards a fully orchestrated artificially intelligent game designer.

## 2 Games as a Multi-faceted Creative Domain

Games lie 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. Games are by their very nature multifaceted: based on [73], we identify six facets of games where creative input by either a computer or a human is necessary.

### *2.1 Visuals*

Like any digital medium, the vast majority of games are displayed on a screen. A game's visuals are primarily geared towards representation: in *Super Mario Bros.* [170] for example, visuals allow players to identify that their player's avatar is a plumber, that the enemies are turtles, or that some of the tiles can be walked on. More than that, however, visuals provide hints about the functions of the elements they represent: cracked floor tiles in *Prince of Persia* [152] hint that these tiles may fall if the player stands on them, while a boss monster's windup animation may hint at an imminent barrage of attacks that the player must avoid. Finally, on-screen visuals may depict items outside the gameworld, such as user interfaces in the form of menus or heads-up displays.

The gameworld where players interact with in-game objects (enemies, puzzles, hazards, other players, etc.) can have two-dimensions (2D) or three-dimensions

(a) **Photorealism:** Madden NFL 18 ©Electronic Arts, 2017

(b) **Caricaturism:** Limbo ©Playdead, 2011

(c) **Abstract:** Geometry Wars ©Activision, 2007

Fig. 1: Different visual styles in games.

(3D). For example, classic 2D platformer games such as *Super Mario Bros.* are displayed as a cross-section of a hypothetical world, from a side-view perspective. Modern games such as *Candy Crush Saga* [165] follow a similar 2D appearance, from a top-down perspective. Both side-scrolling and top-down perspectives have been extensively used for 2D games from the 8-bit era of the 1980s to modern-day mobile devices. The illusion of the third dimension (depth) can be simulated using two-dimensional graphics through scaling and depth ordered drawing, as in *Wolfenstein 3D* [141], or using an angled isometric view with a three-quarters perspective [19]. Truly 3D games became mainstream in the 1990s, and have since then dominated the marketplace, especially for larger (AAA) game companies. Games set in 3D environments offer an immersive experience, and can allow the player to see through their avatar's eyes (first-person camera). When players have full control of their first-person camera, interaction becomes more natural. Therefore, virtual reality games almost exclusively use the first-person camera view.

Due to the variety of ways that games can be rendered, visual design offers game makers immense creative freedom. On the other hand, game visuals are representational and may be constrained to resemble real-world references. This is particularly relevant for 3D games that aim for a realistic and immersive experience.

*Photorealism* in games attempts to closely match the real world, from the appearance of the assets themselves (e.g. high-polygon meshes, high-resolution textures) to the way that light reflects on different surfaces. Similar to several art movements with similar goals (realism of the 1800s being an obvious example), photorealistic game art may attempt to painstakingly detail everyday real-world objects, such as cars' rims in racing games or athlete's faces in *Madden NFL 18* [155] (see Fig. 1a). More interestingly, however, photorealism can be used for game objects with no real counterparts or references in the real world, although rules of physics and lighting are presumed to still exist. This is particularly relevant in fantasy or science-fiction gameworlds, which abound in the game marketplace. Järvinen [49] identifies the former type of photorealism in game art as *televisualism* and the latter as *illusionism* or fictional photorealism.

Contrary to photorealism, many games borrow from comic art style to exaggerate certain elements of a gameworld or the characters within it. The motivations for such *caricaturism* in game art are manifold: caricatures draw players' gaze towards

important visual cues, but also elicit and accentuate specific aesthetics to the players. Caricaturism was necessary during the early eras of games (e.g. the 8-bit era of the 1980s), where each visual element consisted of a few dozen pixels and a handful of stark and vibrant colors. In this constrained medium, game makers had to design monochrome or duochrome enemies which could stand out in a monochrome background: this often resulted in oversized weapons and eyes versus undersized bodies and legs. While newer technologies have alleviated many such constraints, the ability of caricatures to grasp player attention is still exploited in games with many disparate elements on screen at any time. Caricaturization allows players to control diverse hordes of units in real-time strategy games such as *Warcraft II* [148], and to notice patterns in a screen full of sweets in *Candy Crush Saga*. Caricatures geared towards eliciting specific aesthetics also offer a broad range of options to a visual designer, from the monochrome and unwelcoming world of *Limbo* [176] where all characters are black outlines with white eyes (see Fig. 1b) to the vibrant particle effects and chaotic shifting backgrounds in *Super Smash Bros.* [172].

On the extreme end of caricaturization, game art veers into non-representational approaches similar to abstract expressionism in the visual arts. Since games usually revolve around a player trying to overcome challenges in an environment, the environment (and the player's avatar) usually must be somehow represented. Therefore, abstract game art is fairly rare: however, games focusing on mechanics rather than a player's immersion can use abstract game art (see Fig. 1c). Examples of early games that feature abstract game art are *Tetris* [174] or *Pong*[1] [142]. *Thomas Was Alone* [168] is an interesting example of a modern abstract game where an elaborate story and character dialog is juxtaposed by an abstract representation of the world and characters as rectangles.

## *2.2 Audio*

Similarly to films, a game's audio has a complementary role to the game's visuals and is often overlooked. Early game audio was challenged by technological limitations and largely revolved around a handful of monophonic channels, i.e. "the blips and bloops of Pac-Man and Super Mario Brothers" [132]. However, music and audio in games has evolved rapidly in recent decades and has been recognized with awards by BAFTA ("Audio Achievement", "Music"), Spike TV ("Best Original Score", "Best Song in a Game") and most game festivals. Coupled with visuals, game audio aims to immerse the player through "a shift of perceptual focus, from an awareness of 'being in and part of' reality to 'being in and part of' virtuality" [35]. As argued by Collins [18], sound in interactive media is dissimilar to film or music (sound that you listen to) but instead merges with the player's actions in-game (such as an avatar jumping) and out-of-game (such as a player pressing the "D" button on a keypad) as much as it merges with the player's visual stimuli.

[1] Pong is a reference to ping-pong (or table tennis) although the rectangular paddles, square ball and its movement do not resemble table tennis in any way.

(a) **Voice Acting** by Jack Black in Brütal Legend ©Double Fine, 2013

(b) **Sound Effects** as hints to the player in Thief: The Dark Project ©Looking Glass, 1998

(c) **Soundtrack** and matching its rhythm in Guitar Hero ©Harmonix, 2005

Fig. 2: Different uses of audio in games.

Game audio could be further split into background music, sound effects, voice acted dialog, and musical score for the game's introduction or cinematics. The latter follows a pre-scripted format as it can not be interacted with but only consumed in a similar way to film or music; it is therefore less relevant in our analysis. Voice acting is now a staple feature in major commercial games, especially by larger companies; an abundance of Hollywood actors have lent their voices in games, such as Sir Patrick Stewart and Sean Bean in *The Elder Scrolls IV: Oblivion* [143] or Jack Black in *Brütal Legend* [154] (see Fig. 2a). While a player's interaction with non-player characters (NPCs) through dialog can be a core part of the gameplay experience, such interaction is with the narrative content of the dialog; voice acting merely adds a level of immersion, making characters more relatable or memorable but is consumed in a similar way as the cinematic musical score.

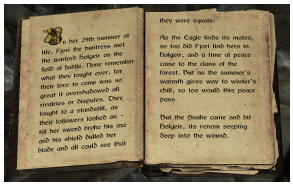
Sound effects are particularly strong examples of Collins' position for game audio merging with a player's actions [18]. A player's actions are often accompanied with a sound effect, usually unique to this action. Examples include the jumping sound for the player-controlled Mario in *Super Mario Bros.*, or the short acknowledgment in English by a unit receiving a harvest command by the player in *Warcraft II*. Some sound effects are tied to the outcomes of player actions: an example is the delayed sound effect when three or more jewels are removed in *Bejeweled* [177] due to a player swapping their positions. Such sound effects (and the accompanying particle effects) provide an audiovisual reward to the player and further motivate interaction with the game. Similarly, opponents' actions are often followed by sound effects, such as the "huh?" sound of guards starting to investigate a suspicious sound in the stealth game *Thief: The Dark Project* [166] (see Fig. 2b). Such sound effects attract players' attention, alerting them of dangers or notifying them that their actions have had unexpected outcomes. For instance, players may pick up a weapon by walking through it without noticing in a first person shooter game; they are alerted of the fact by a weapon cocking sound. Finally, there are sound effects that help situate the player in the environment: examples include the sound of the player's footsteps which may change depending on the surface that is being walked on. Such environmental sounds, often modified based on its source's position in the world

and the head position or rotation or the player's avatar [40], can be coupled with the background music to become part of the soundscape (described below).

Background music in games is often perceived as decorative, providing ambience or some aural stimulus during certain repetitive tasks. However, background audio can be foregrounded when it affects gameplay: examples include rhythm games such as *Guitar Hero* [160] where the music hints when certain keys must be pressed by the player (see Fig. 2c), or similarly in *Crypt of the NecroDancer* [151] where timing player actions to the musical beat is the key to avoiding getting hurt and losing the game. Background music is often customized to a specific game level (or part thereof) or to a specific player activity. For example, the 28 tracks in the awarded soundtrack of *The Elder Scrolls IV: Oblivion* are split between "explore", "public", "dungeon", "battle" and "special" categories: a random track is played when the player is exploring the overworld (from the "explore" category) or underground locations (from the "dungeon" category), fading into a random track from the "battle" category when the player is attacked. Transitions between background music tracks may be as basic as the above, with simple fade-ins and one random track in one category chosen when the current track finishes. Interest in more dynamic background music in games has led to adaptive music in games such as *Tom Clancy's EndWar* [181], where the music does not loop or fade but is dynamically adapted to the evolving narrative of the strategy game being played, or *Proteus* [164] where the carefully crafted and impeccably synchronized music channels are turned on or off based on the location and camera view of the player.

## *2.3 Narrative*

While not all games rely on a story structure, dialog, or plot exposition, a plethora of large-scale digital games rely on their narrative to draw players into their story-world and action. Early digital games such as *ZORK* [163] were closer to hypertext [53] or "choose-your-own-adventure" booklets, where pre-scripted narrative events were presented textually as a response to specific player commands (such as "examine leaflet" or "go north"). Many contemporary adventure games are similarly structured to reveal segments of a pre-authored narrative (usually via an embellished audiovisual cutscene) when players find pre-determined solutions to puzzles. Similarly, a gameworld's history is often revealed through in-game interactable items, such as books that can be read (see Fig. 3a); such items often contain short stories which are individually authored with a clear beginning and end. The freedom of players to explore a vast gameworld poses a challenge to narrative structures as designers can not anticipate where players may go first, which actions they will take, or which characters they will converse with. This very *interactivity* has been identified by Ryan [104] as a crucial component for deciding what kind of stories can be told by digital media: specifically, a user being able to change the conditions. Ryan goes further to distinguish interactivity along the axes of internal (the users projecting themselves as a member of the fictional world) versus external (when the




(a) Books with complete stories provide atmosphere and clues in The Elder Scrolls V: Skyrim ©Bethesda, 2011

(b) Uncovering a pre-written narrative structure in L.A. Noire ©Rockstar Games, 1999

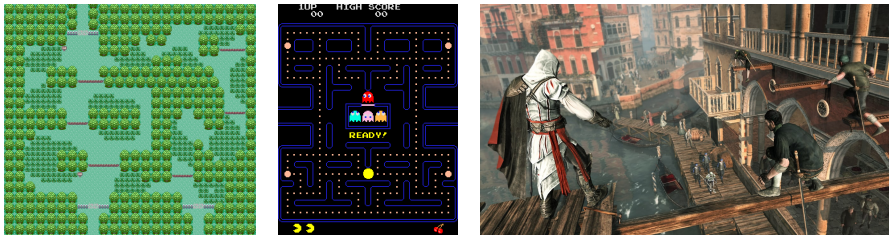
(c) Players making their own stories in The Sims ©Maxis, 2000

Fig. 3: Examples of narrative structures in games.

user is outside the fictional world), and exploratory ("the user is free to move around the database but does not make history alter the plot" [104]) versus ontological (the user can send the story down one road of a forking path). Narrative in games can follow many different paradigms along this typology, such as the exploratory nature of mystery games such as *L.A. Noire* [178] (see Fig. 3b) where players can move freely but will inevitably uncover a pre-written history about what happened (exploratory-internal), or games such as *The Sims* [167] (see Fig. 3c) where players can swap control of different characters and decide who they will fall in love with, fight and so on (external-ontological).

## *2.4 Levels*

A player's actions within a digital game take place in a virtual space traditionally referred to as a *game level*. Game levels can be as abstract and minimal as those of *Tetris* [174] or *Pong* [142], where only the surrounding walls (which are often invisible) are designed to constrain the player's actions. A game may consist of one vast level, such as the gameworld of *The Elder Scrolls V: Skyrim* [145] or the galaxy of *Stellaris* [175]; it can be made out of a large number of self-contained levels experienced sequentially, as in *Super Mario Bros.* or *Arkanoid* [179]; different levels can also be presented as options to the player as in *Cities: Skylines* [153]. Similar to visuals, game levels may not necessarily reflect the real world: as noted above, abstract levels such as those of *Tetris* and *Arkanoid* are merely spaces designed to challenge a player without reflecting the real world. On the other hand, a large number of games rely on real-world tropes for designing levels regardless of whether their visuals are photorealistic or caricaturized. For example, levels in *Pokémon* games [158] feature villages, beaches, forests and rivers, even though the trees in its forests are unrealistically small and set in a dense grid to create a maze-like structure (see Fig. 4a). When games' visuals and general direction is closer to the real world, levels have to closely mimic natural habitats, from the type and den-




(a) **Unrealistic** sprites as obstacles in Pokémon: Gold ©Nintendo, 1999

(b) **Full view** of the level in Pac-Man ©Namco, 1980

(c) **Open World** based on real locations in Assassin's Creed II ©Ubisoft, 2009

Fig. 4: Examples of level structures in games.

sity of fauna around rivers in *Far Cry 5* [183] to the layout of European cities in *Call of Duty* [162]. Similarly, level design in science-fiction or fantasy worlds must balance between the known and the evocative unknowns, with large-scale mega-structures reminiscent of modern skyscrapers as in *Mass Effect* [146] or bombed and repurposed Washington landmarks in *Fallout 3* [144].

Regardless of their style or proximity to real-world structures and physics, designing game levels is similar in many ways to architecture. Similar to architecture, level design must find the appropriate balance between "form" and "function", i.e. how evocative and appealing the game level is versus how easy it can be for the player to navigate and recognize. In the spirit of the Bauhaus movement, game levels could be designed to be purely functional without any ornamentation: an example is the *Pac-Man* [169] maze where the goal is function, i.e. allowing the player (Pac-Man) to clearly see the entire level, the game's goals (pellets) and the game's challenge (ghosts), and navigate it (see Fig. 4b). Conversely, adventure games are known to include useless clutter in their levels for the purpose of increasing the challenge of the primary mechanic (identifying a useful if obscure item or solution on the screen) as well as providing humorous descriptions or responses by the protagonist when other non-solutions are selected by the player. Between these two extremes, most game levels have to provide some decoration to help the player identify with the game's setting, such as Venetian architecture in *Assassin's Creed II* [182] (see Fig. 4c), while at the same time ensuring that players can take advantage of most of the game's mechanics at their disposal in a rewarding and straightforward way. In the same example, buildings in *Assassin's Creed II* must be tall (perhaps taller than normal buildings in 15th century Venice) in order to motivate and reward the climbing mechanic while roads must be wide and obstacles evenly spaced apart to allow for the Parkour-style movement that the game is known for.

## 2.5 Rules

The rules of a game are paramount to its interactivity, as they constrain and guide the player towards specific goals and gameplay patterns. Some rules determine the game's ending conditions, i.e. when the game is won and when the game is lost, which respectively provide the player with a goal and a challenge. Among other game rules, of particular interest are those pertaining to player actions: these *mechanics* dictate how the player can interact with the game and are the primary drivers of player agency. Mechanics are usually described as verbs [50] such as "jump" in *Super Mario Bros.* or "take cover" in *Gears of War* [156]. Mechanics are often differentiated from other game rules, which determine the transition between game states, as "rules are modeled after [player] agency, while mechanics are modeled for agency" [110]. The use of a mechanic, e.g. Mario jumping over a Goomba enemy, can trigger a rule-driven state transition, e.g. removing the Goomba from the game. The interaction between the many rules, mechanics and winning conditions lead to complex dynamics and shape the player experience, which we discuss in Section 2.6.

## 2.6 Gameplay

Designing a digital game can be a complex creative task, but a game's ultimate purpose is to be played by the end-user. Gameplay refers to the experience of playing the game, or "the phenomenological process of an epistemic agent interacting with a formal system" [111, page 104]. During gameplay, a player interacts simultaneously with all other facets; their interrelations can shape the player's affective state and immersion [11]. 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 *Far Cry 5*).

Of particular interest is how players interact with the formal systems: the game's rules and mechanics. While this ruleset is pre-scripted by the game's designers, each player can use it in different ways — potentially exploiting it in unforeseen ways which can break the intended game balance. An example unforeseen exploit was discovered by players in *The Elder Scrolls V: Skyrim*: a player could pick up a bucket or other concave container and place it over an NPC's head, effectively blocking their line of sight completely and allowing the player to steal all of the NPC's valuables lying around. More interestingly, the interaction between different rules and mechanics could lead to emergent *dynamics* [46]. Such dynamics can be influenced by social and competitive concerns on the part of the player community, which can lead to an ever-changing *metagame* [15] of strategies and counter-strategies — especially in (multi-player) e-sports games such as *StarCraft* [147]. Beyond the primarily functional concerns of dynamics, however, the interaction among all facets (and especially visuals and audio) can evoke strong emotional responses from the player.

These responses range from basic emotions such as fear and joy [28] to a broader range of *aesthetics* such as sensation and discovery [46]. 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.

## 3 Orchestration

While the different creative domains incorporated in games are fairly well-defined, it is important to note that games are more than the sum of their levels, winning conditions or visual styles. Elements in each facet contribute to the final play experience and need to be harmonious with other elements in this or other facets. At the simplest level, a decision made by a creator of one facet can constrain the possibilities of designers in the same or other facets; for example, a rule that determines the delay between the player avatar's death and the game restarting constrains the duration of the animation of the avatar's death and the sound effect that accompanies it. More broadly, the style adopted in one facet will likely constrain the style in other facets: an obvious example is the choice for photorealistic visuals, which necessitates that game levels' architecture closely follows real-world tropes and game rules that also match real-world physics.

In most cases, however, the interaction of facets and their impact on the play experience is less straightforward. As an example, *Amnesia: the Dark Descent* [157] is a horror game set in an ancient manor or castle, where the lone male protagonist must explore the different floors armed only with a lantern while avoiding enemies which can either kill him or drive him insane. The losing conditions (death or insanity) are reinforced by the rule that darkness slowly lowers sanity and the rule that the oil for a lantern (which can be turned on or off) can run out. These rules force the levels to be designed with some areas already lit by ever-burning torches, which are connected via dark or lit areas: the player is bound to navigate through the lit corridors to avoid using up the lantern's oil. On the other hand, the level design places most oil resource pickups in dark rooms, forcing the player to explore in unsafe, scary areas. The visuals reinforce the tension of the experience, with post-processing filters on the player's screen making it more difficult to see when sanity is low: since sanity drops in dark areas, the filters on the visuals make it even harder to see in an already dark space, and thus increase frustration and panic. Finally, ambient sound effects such as creaks or moans (even when nothing is there) keep players on edge, while at low health or sanity the background audio includes the character's erratic heartbeat as an aural indication of the game state (rules) but also as another source of stress. While the design decisions on each of these facets — and more, such as the protagonist's backstory or the design of the enemies — are harmonized and perfectly aligned towards an intended gameplay experience, it is difficult to identify which facet was fleshed out first and constrained others in this

process. It is far more likely that the level design, visuals, audio and rules all evolved together through iteration and playtesting (i.e. feedback from the gameplay facet).

Finally, the design decisions in different facets (and their harmonization) can be guided by established game patterns such as player repertoire and game genre. Player repertoire consists of "the skills and methods for overcoming the challenges of the game" [54, page 56]. Playing any game requires the player to expand their repertoire and perfect their reflex time or tactical response to challenges. Moreover, "a player approaches every game with whatever repertoire of skills he or she has" [54, page 5] and thus the design of new games can take advantage of patterns of play from past games that players are likely to have experienced. Elements such as the avatar's death abound in games, and thus will be easily understood by the player community. Designers can take advantage of this fact, and either use common rule, level or visual patterns (respectively, avatar death, maze-like structures, or hovering icons above important NPCs) to make the game more accessible. Alternatively, designers can go against a few of those patterns in order to subvert and challenge the players' repertoire. Game genre plays a similar role: players expect specific elements from specific types of games, and *genre* has been a staple way of categorizing games since the 1980s (e.g. adventure games, shooter games, puzzle games). Player repertoire is tied to genre: most puzzle games do not feature the avatar's death, while most shooter games do. All facets are tied in one way or another to genre: music rhythm games presume simple visuals so that the player easily understands which button to press while the audio needs to feature tracks of high musicality; horror games almost always feature a ghost story (narrative), overpowering monsters (rules) and intense negative feelings during gameplay. Using these pre-existing formulas in the design of, and relationships between, game facets can be a shortcut for many of the challenges in designing a new game.

## 4 Procedural Content Generation in Games

Digital games have de facto relied on algorithmic processes for handling mundane tasks such as collision checking, capturing key strokes and transforming them into in-game actions, or rendering visuals onto the screen. However, even in the earliest days of digital games, algorithms were at times allowed to take creative decisions which affected the player's engagement and perceived challenge. For instance, in the computer game *Rogue* [180] an algorithmic process was tasked with creating a fresh new dungeon every time the player started a new game. The dungeon consisted of rooms and corridors, as well as monsters and traps that were designed to challenge the player. The game was designed explicitly to highlight and take advantage of this algorithmic generation process: the game would delete the player's save game if their avatar died while exploring the dungeon, and thus the player would need to start anew, in a new dungeon, without being able to anticipate the layout or encounters within it. Another instance of algorithmic generation is the video game *Elite* [150] where a vast universe (8 galaxies, each with 256 planets) is generated

(a) **Level Generation** in Diablo III ©Blizzard, 2012


(b) **Weapon Generation** in Borderlands ©Gearbox, 2009

(c) **Enemy Generation** in Shadow of Mordor ©Warner Bros, 2014

Fig. 5: Different types of generated content in commercial games.

at the start of the game, letting players explore new galaxies if they wish to start a new game. As *Elite* is an open-ended game where exploration, trading and combat can be undertaken over a very long period of gameplay, the generative process allows the vast gameworld to be stored in a few parameters (e.g. a random seed) and past games can be saved and loaded despite the small memory available to 1980s computers. Many of the early instances where games offloaded creative tasks to algorithmic processes were motivated by these two factors: the desire to provide new experiences in every playthrough, and the ability to compress complex game assets (such as a vast gameworld) into a few bytes of memory. Development of new generative algorithms was largely driven by these factors, in a similar way that graphics renderers were developed to be as efficient and as impressive to players as possible despite limitations of the technology available at the time.

As digital games diversified over the past 30 years, algorithms have similarly been used for creative tasks in a variety of roles. Some of the original motivations for algorithmic generation are still applicable; for instance, *No Man's Sky* based its advertisement campaign on the vast number of procedurally generated planets (and their contents), which could be compacted into a small save file due to the compressible generator parameters. Similarly, when presenting *Diablo III* [149] the developers touted the series' "hallmarks: randomized levels, the relentless onslaught of monsters and events in a perpetually fresh world, unique quests, tons of items, and an epic story [...]" [8], similar to the perpetually fresh world of *Rogue* in each playthrough. However, modern games often use algorithmic processes during development and not just to generate content unique for each player. For example, to recreate the mountainous landscape of Montana in *Far Cry 5* developers had to encode some of the physical characteristics (e.g. erosion) or biomes (e.g. vegetation) into a generative algorithm which could create the vast gameworld. This first draft of the gameworld was then enhanced by game designers, e.g. to add game-specific objects such as quest locations or characters [14].

While the majority of commercial games that use generative algorithms focus on the creative facet of level design (see Fig. 5a), there are interesting examples of generation in commercial games for the other facets. Of note is the algorithmic design of weapons in *Borderlands* [159] (see Fig. 5b), which generates both visuals

of the 3D weapon and its game rules (e.g. bullet speed, damage, etc.). *Borderlands* created new weapons by combining 3D models of existing weapons (e.g. long barrel, sniper sight) and calculating the weapon's in-game behavior based on those (e.g. a laser rifle with a sniper scope which freezes targets hit by its slow bullets). Other games use algorithms to generate non-player characters: of particular note is *Middle-Earth: Shadow of Mordor* [185] in which orc antagonists are generated with a matching name and appearance (e.g. "Ruktuk the corrupt" is likely to have a face full of sores) as well as special powers and weaknesses that the player must discover and exploit in order to defeat them (see Fig. 5c). Several games use algorithms to generate aspects of the narrative, such as the Radiant Storytelling system in *The Elder Scrolls V: Skyrim* which customizes quests' locations and characteristics, or Nemesis quests in *Middle Earth: Shadow of Mordor* which are instantiated according to orc chiefs' personalities and their history with the player. The algorithmic generation of 3D models and textures abounds in games for mundane features, such as small variations in plant life, through generative grammars and L-systems [128]. Similarly, procedural textures are a mainstay of the game industry — although they arguably lack any aspect of computational creativity or decision-making from the algorithms' part. Finally, a small number of games have experimented with procedural music, such as *Tom Clancy's EndWar* which can provide endless music or *Proteus* where the soundscape adapts to the player's view. In all known cases of procedural audio in games, the algorithms are scripted to turn human-authored bits of music on or off, or to apply filters to pre-made sound effects.

## 5 Artificial Intelligence and Game Design

While commercial games for the most part use simple scripts to design parts of a game, there is an increased interest within academia to ascribe more creative freedom (and more complex artificial intelligence) to algorithmic game design. For over 10 years, a number of conferences[2] and journals[3] have tackled artificial intelligence (AI) applied to games — both for playing and for designing them — while AI algorithms for game design have been accepted in many highly-ranked journals. Two books on the topic of procedural content generation have been published in the last couple of years, one focusing on the academic [108] and one on the practitioner [109] perspective, while a comprehensive handbook on AI applied in games more broadly has also been recently published [135].

[2] Indicatively, the IEEE conference on Computational Intelligence in Games, the AAAI conference on Artificial Intelligence in Interactive Digital Entertainment, and several tracks of the ACM conference on Foundations of Digital Games.

[3] A core journal on the topic is the IEEE Transactions on Games and its predecessor, the IEEE Transactions on Computational Intelligence and Artificial Intelligence in Games.

## *5.1 Generating Content through AI*

Research in PCG has explored a broad range of algorithms to generate an equally broad range of game content. Characteristics of such PCG algorithms were put forth by Togelius *et al.* [129], considering the moment of use (online while the game is played versus offline during development), the type of content produced (content necessary for gameplay versus optional or decorative content), as well as the potential for parameterization and the stochasticity of PCG algorithms. More broadly, Togelius *et al.* [129] also distinguished between *constructive* algorithms, which use scripts to generate content without assessing whether the result is satisfactory, and *generate-and-test* algorithms which verify whether the output satisfies certain constraints. While commercial games almost exclusively use carefully scripted generators to ensure that all output is playable (constructive approaches), the main interest within game AI research has been on generate-and-test methods. What makes generate-and-test algorithms so compelling is that they require theorizing and formalizing of constraints and evaluations on what constitutes an appropriate type of content; often, such assessment necessitates that the game is played with the new content.

There are many different approaches to generate and test game content: a simple approach would be to generate one artifact, test it and re-generate it if it fails any designer-specified constraint. While this simple approach can work well when the generative scripts are likely to produce viable artifacts, it can lead to an endless or nigh-endless re-generation loop in cases where the constraints are strict or the generators are very likely to produce unwanted content. A subset of generate-and-test approaches are search-based [129], in which content that is not satisfactory is iterated upon to improve it — rather than re-generate it from scratch. Evolutionary computation (EC) is often used to perform this cycle of iterative improvement. At its most high-level description, EC evaluates a population of candidate solutions, selecting the best among them to create new candidate solutions via a set of genetic operators, and replacing some or all past solutions with the newly created ones [52]. This iterative testing and improvement is an ideal paradigm for search-based content generation: an initial population of often random content is iteratively improved upon, based on a *fitness function* which can provide a measure of game content quality, until an individual satisfies some minimal threshold of quality or until the computational budget is spent. Evolutionary approaches have been used extensively for content generation, with many variations. Notable variations include fitness functions which evaluate content based on simulations of gameplay, requiring an automated playtesting framework so that the notion of playability, balance or others can be computed based on the playtraces [69]; using human input instead of mathematically formulated fitness functions to assess content based on visual inspection or human playtests with the candidate solutions [123, 72]; constrained optimization by combining a granular fitness function for playable content with hard constraints on what constitutes unplayable content [74]; and evolution towards novelty [83] or surprise [33] to create a broader range of output rather than explicitly "better" output.

Other PCG algorithms follow alternative paradigms for constructing or assessing game content. Declarative programming has been used extensively, for example, to formalize a design space of desirable content via hard constraints [116]. Using variations of declarative programming such as Answer Set Programming (ASP), search problems can be reduced to stable models and subsequently searched in a straightforward fashion, for example via backtracking. The appeal of ASP-based content generation is the fact that these algorithms always terminate, returning all possible solutions to the given constraints with a minimal computational budget, while its knowledge representation as "what to compute instead of how to compute it" [116] can be highly compact and error-free. These algorithms have limitations, however, as they are ideal for a specific set of problems, e.g. puzzle games [114], and can not handle stochasticity well (such as random weapon damage); moreover, formulating the design problem as a set of (usually complex) constraints may be as difficult as finding the solution to the problem itself.

Related to declarative programming, planning approaches have often been used to generate content in games — especially for game narrative. Since planning approaches create a plan, i.e. "a temporally ordered sequence of operations" [101], they are ideal representations for a linear plot where events are chronologically and logically ordered (e.g. a player must first pick up a sword and learn the location of a monster, then kill it). Planning algorithms such as the one used by Mimesis [136] rely on *steps* for each event in the story: each step "is defined by a set of *preconditions*, the conditions in the world that must hold immediately prior to the step's execution in order for the step to succeed, and a set of *effects*, the conditions in the world that are altered by the successful execution of the action" [137]. Plans need to address *flaws* (i.e. open preconditions that have not been established by a prior plan step) and *threats* which can undo an established causal relationship in the plan [16]. Variations of this basic architecture such as hierarchical planners [124, 60] which decompose abstract task into primitive sub-tasks and partial planners where temporal orderings are only established to resolve threats [55] have been used for creating both static and interactive narrative. Interactive narrative and drama management, which can often be found in games, comes with challenges of its own as the player's agency may break certain preconditions for the execution of some of the plan's steps. This would require re-planning in order to adhere to the author's original intent or to satisfy the newly established intent of the player's character [100].

In a different direction, game generation algorithms often rely on external data: the vast real-world knowledge encoded in modern online or offline data repositories can prove useful in discovering mappings between in-game and real-world information. Two different approaches for exploiting game-external data are common, broadly identified here as a direct and an indirect approach. The direct approach queries online databases and transforms the results into game content. Examples of this approach include a Google image search with a specific AI-discovered keyword, and the subsequent transformation of the image into an appropriately sized and colored sprite [23]. The transformation can be more or less direct; Barros *et al.* [4] provide examples of game generators spanning the spectrum of transforma-

tion versus fidelity to the original data. The indirect approach parses a vast database — often consisting of content in games similar to the one the content is intended for — to discover patterns and exploit them to generate new content. The term "procedural content generation via machine learning" (PCGML) identifies machine learning methods which can be used to directly form new content [122]. Examples include probabilistic Markov models for deciding the next tile in a platformer game level based on preceding tiles [118], or filters that can automatically repair poorly formed platformer game levels based on patterns of well-formed levels [48]. Machine-learned models can also be used to assess content, for example by using computer vision models to assess what real-world object matches an artifact [66], or by optimizing game levels based on learned models of a designer's style [76, 70, 82].

Using this broad range of AI techniques, game content of many different types has been generated. Indicative examples per facet are discussed below:

- **Visuals:** While most methods for generating visuals are based on computer graphics techniques and orginate from mathematical models of noise [64], there have been a few attempts at AI-based generation of visuals. In [45], graphic shaders were evolved towards a designer-specified color palette: the designer could specify the intended color for a certain scene and shaders would be evolved in order for that color to be dominant in the final rendered scene. In [131], procedural filters on textures could apply slight visual changes to modeled scenes based on semantic information such as "high vandalism" on models of houses. The Galactic Arms Race game [39] allows players to interactively evolve the particle effects which represent the players' weapons, influencing their color, animation and trajectory. In the *Petalz* social game [102], the appearance of flowers was generated through pattern-producing networks [120]; the flowers themselves could be shared among users and further evolved via mutation or recombination with the current player's flower collection. In [67], arcade-style spaceships were evolved towards designer-specified visual properties such as symmetry, simplicity and other patterns or towards visual novelty. More broadly, there is extensive work on evolving shapes for spaceships to match a machine-learned model of user taste [75] or to disrupt current patterns in generated content [71].
- **Audio:** While games such as *Proteus* [164] used pre-authored pieces and simple rules to adapt the soundscape, AI has been used sparsely for game audio. Of note is the *Sonancia* system [84] which chooses from a range of pre-written sound tracks to play for specified events or areas of a game. Scirea *et al.* [106] use music generated in real-time to foreshadow game events according to a pre-written narrative arc. *AudioInSpace* [44] uses pattern-producing networks which the player can evolve to decide the timing and pitch of the game's background audio. *Audioverdrive* [42] is a side-scrolling space shooter with a procedural audio system that interacts via a rule-based system with the level design: for instance, the height of the bottom terrain controls the pitch of the bass synth, while treble sound events trigger the placement and timing of enemies. Earlier examples of procedural music in games are surveyed in [17].
- **Narrative:** Building on extensive work in interactive narrative and drama management [16], games such as *Façade* [91] and *Prom Week* [92] model the game

state in a way that allows the manager to choose which NPCs utter which lines of pre-authored dialog. In these cases, a sequence of utterances is chosen as a reaction to a player's action, or as a way to introduce a new narrative beat (such as introduce an argument between NPCs). On the other hand, a repository of stories can be used as a basis for automatically constructing interactive plots in a fairly author-free manner [36]. Not all games are focused on story and dialog, but may still use AI to craft a companion narrative: *Charbitat* [2] generates game worlds and then generates quests to fit them, while the General Mediation Engine [103] 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).

- **Levels:** Level generation is by far the most popular PCG subdomain, both in academia and in commercial titles of the last 30 years. Level generation can be performed in a constructive manner [107], especially in commercial games. However, many AI techniques have been applied for level generation such as generative grammars [27], artificial evolution [129], declarative modeling [112], and constraint solving [93, 117]. Level generation has often been driven from a mathematically defined quality formulation based on the similarity with an archetypical "example" [1], a popular level design pattern [80] or an intended affective response [97]. Level evaluation can also be offered directly to players, e.g. using interactive evolution to directly select a preferred level [12], as input to a learned model of users' level preferences [70], or indirectly based on gameplay logs such as time spent in combat [13]. Patterns found via a corpus of similar levels [121] or based on gameplay feeds [37] have also been used directly to generate game levels [122] rather than as an evaluation function. Finally, AI has been used for level generation in design interfaces, as a companion to human level designers [78, 79, 117, 31].
- **Rules:** AI-based generation of rules and mechanics is one of the most challenging aspects of game generation [126] for two main reasons: (a) rules greatly affect the playability of the game [3]; (b) their quality can arguably only be assessed via playtesting. In board games, the *Ludi* system evolved interaction rules [10, 9] in a complex evolutionary cycle of level and rule generation and subsequent gameplay generation (see Section 5.3 for more details). In a more constrained analog medium, symmetric chess-like games have been generated by Pell *et al.* in order to create robust game playing agents [98], while EC was used in [62] to guide the search of new rules for new pieces based on simulations with different heuristic-based game playing agents. 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 [115] or based on the ability of an AI controller to learn the game [127]. Game mechanics have been generated via planning algorithms in [140], attempting to make the least possible change to existing mechanics in order to achieve the satisfaction of different playability constraints such as end-game goals (e.g. reach the exit), maintenance goals (e.g. stay alive) and engine constraints (e.g. not occupy the same space as another entity). Finally, game mechanics in [88] were generated

based on similarities with existing sprites and their rules or interactions; importantly, these mechanics were provided as suggestions for a human game designer rather than automatically added and tested in a game.

- **Gameplay:** As noted in Section 2, gameplay is unique among the facets as it is a task not undertaken by designers but by players. Simulating a human player through artificial intelligence is much closer to domains such as robotics [65] or self-driving cars [90, 32]. In the context of AI-based game design, game playing agents are primarily useful for *automated playtesting* of other game content. These automated playtests can then inform the evaluation of the underlying game content, and test for constraint satisfaction or provide a heuristic that can be optimized via search-based PCG [129]. In simulated playthroughs, it is common that the game playing agent plays optimally, especially when creating levels for an AI competition [99]. However, some degree of human-like behavior can be desirable as well from an automated playtester. For instance, artificial drivers [43] maximized an "objective" efficiency (i.e. distance covered in a preset time) while minimizing deviations from captured player data in terms of steering and acceleration. In other work on AI playtesting, artificial agents attempt to play the *MiniDungeons* puzzle game [41] with different objectives, such as collecting the most treasure or taking the fewest steps.

## *5.2 Orchestrating Game Generation*

The ideal of AI-based game design presumes an AI taking design decisions on most — if not all — aspects of a game. To reach this ideal, the AI system must be able to orchestrate the various facets of a game. Similar to a human designer or team, the computer must understand how choices in the game rules may require a tweak in the game's visuals or a complete re-write of its backstory. As argued in Section 3, most elements of a game are intertwined. When building a generator of game levels in an existing game, the programmer can make assumptions on the genre, rules, visuals and general gameplay style from a design document, prequels to the title, or tropes in similar games. In a game designed primarily by an artificial intelligence, however, these links between facets and game elements must be considered during the process of design. In [73], purely algorithmic game design was hypothesized along a spectrum between a purely hierarchical, top-down process and an organic, bottom-up process. Specifically, a top-down approach would involve a generated high-level frame for the game (e.g. a game pitch) which would be used as a guide to generators that could iteratively add details to it — going from a vague color direction/palette down to the most minute details such as wall texture generation. A bottom-up approach could use a system similar to the blackboard [29] and allow expressive generators able to create a large variety of content (e.g. labyrinthine single-player game levels as well as strategy game levels for multi-player competitions) to contribute to a general framework. Cohesion between the elements on the blackboard could be tested by an internal or external algorithm (or even by a human

designer) and incoherent content could be regenerated until there was a cohesive whole game. As suggested in [73], AI orchestration could target the production of a complete game directly playable by end-users, or create a draft schema which could be further refined, edited or partially re-generated by designers. In the latter case, the AI system could be partially interactive, allowing designers to tweak parts of the requirements such as the AI-generated game pitch for a top-down approach or "manually" test for cohesion in a bottom-up approach. Finally, the source of real-world knowledge on game design could be embedded into the algorithm, provided as human input directly from the designer (e.g. via parameterization or initial seeds), or discovered from crowdsourced data such as open data repositories, direct player feedback and playtests, or computational models learned from a large corpus of past games and/or player annotations.

A core challenge of AI orchestration and thus AI-based game design is finding a computational mapping between dissimilar facets. At a smaller scale, a number of research projects have attempted to address this issue in a more constrained space. For example, a mapping between the color of sprites in the *Pokémon* games (visuals) and their in-game type (rules) was built as a decision tree and then used to drive generation of appropriate visuals for designer-specified variant Pokémon [68]. Similarly, the mapping between the strategic qualities of chess pieces was mapped to their visual appearance, allowing the generation of pieces for newly generated chess-like rulesets [61]. Mappings between players' affect during gameplay and the structures of a game level has been attempted in numerous ways [134], while similarly the affective response of players to game audio has been learned from experiments done using crowdsourcing [87]. Machine learning seems to be an ideal tool for finding such mappings, although the task is challenging due to the limited or non-uniformly formatted game data available [122].

Other ways of combining facets, without using a data-driven model of their relationships, have been fairly successful in generating games. Using simulations of gameplay as an evaluation function is an example of hierarchical orchestration where game content such as levels or rules is generated first, and then the gameplay is generated based (usually) on artificial playtraces. Unlike previous examples where the mapping from, e.g., levels to a player's affective response during gameplay was predicted via a computational model, simulations actively generate the gameplay facet rather than predict high-level responses to it. Gameplay is simulated in level generators via pathfinding between the level's start and finish, e.g. in [119], or via rules regarding when players should use a mechanic, e.g. in [47], or via solvers for optimal puzzle solution, e.g. in [117, 113]. Such simulation-based evaluations usually trivialize the player's expected experience or aesthetics. On the other hand, [21] uses the same trivial A* pathfinding playtraces but takes into account the computer agent's camera view in order 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). Player precision is simulated via rule-based systems in [47] 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. *Ludi* [10] created gameplay logs using agents as they were learning to play the game. Similarly, [127] evaluated generated collision and scoring rules for simple arcade games based on controllers evolved explicitly for this game. Unlike *Ludi*, the evaluation was 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. Finally, racing track generation in [125] was informed by gameplay traces of computational agents that simulated specific players' skills captured via machine learning. Focusing instead on human players' different priorities when playing a game, levels for *MiniDungeons* [69] were evolved for different procedural personas, i.e. artificial agents with archetypical player goals such as treasure collection, challenge (monster killing), or survival [41].

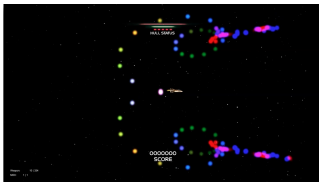
## *5.3 Cases of AI-based Game Design*

While the task of producing a playable game and all its facets by an artificially intelligent creator is an ambitious goal, there have been significant steps towards it. The following sections highlight important stepping stones towards fully orchestrated, fully autonomous game design.

**Angelina**, in its 2012 implementation [25], 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, 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 is 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.

**Game-O-Matic** [130] generates games by transforming human-provided schemas into simple but playable arcade games. Human input is provided in the form of a graph where nodes are nouns which are transformed into game objects and edges are verbs which are transformed into mechanics. An example is "man (node) eats (edge) burgers (node)" which 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 [130] or games which can not 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 transforms human-authored concept maps (micro-rhetorics) into a complete ruleset (i.e. with custom game mechanics, goals and instruction sets). Visuals generation is of minor importance as it is based on a direct online search of human-provided nouns; simi-


(a) **Data Agent:** in-game footage of the latest data adventures generator, used with permission from [34].



(b) **AudioInSpace:** in-game footage, used with permission from [44].

(c) **Sonancia:** in-game footage, used with permission from [86].

Fig. 6: Cases of AI-based Game Design.

larly, level generation is superficial as it decides the placement and number of game objects in the level but does not generate an elaborate level structure.

**A Rogue Dream** [23] uses online sources to discover associations between game objects for a rogue-like game where a player-controlled avatar must combat enemies, gather healing items and reach the exit. The four game objects (the avatar, enemies, healing items, exit) are chosen based on the name of the player's avatar, which is provided as human input before generation. The *avatar name*, as provided by the player, acts as a proto-narrative: semantic associations for the other three game objects are discovered via Google's autocomplete results. For instance, enemies are identified as the next word in a Google query "why do *avatar name* hate…". The visuals for the game objects are sprites transformed from Google image searches on the names of the avatar and discovered associations. The player's abilities, which can be used against the enemies, are based on pre-authored rules templates but influenced by the Google query "why do *avatar name*…". The game takes place in a generated level, using fairly simple scripted algorithms which are not influenced by the avatar's name or any other facets.

**Data Adventures** [6, 7] is a suite of generators which transforms open data into playable adventure games. Different versions of the generators placed a different emphasis on the narrative, with the latest installment [34] having an elaborate handcrafted narrative regarding a murder committed by a time-traveling doppelganger masquerading as a famous historical figure (see Fig. 6a). The player takes the role of a detective who travels around the world to meet different non-player characters (who are historical figures in their own right) in order to find who killed a famous person. The only human input to the system is the name of the victim, while the suspects, locations (i.e. levels), items, and dialogue is generated based on open data. Using primary sources of open content such as Wikpedia for data, Wikimedia Commons for images and OpenStreetMap for levels, Data Adventures recombine that information in novel (and often absurd) ways to create adventures [5]. As an exemplar where semantically linked open data has been used to impart real-world information into a fairly semantically-dependent generated game (as adventure games often are),

it has been an important case for exploring ways to use crowdsourced data as input for orchestration [4].

**Game Forge** [38] generates a game's narrative and then the level in which this narrative can take place. While the game is fairly pre-determined in terms of the theme, tileset, visual identity, and gameplay, it is an important example of hierarchical generation where a thorough narrative (unlike proto-narratives in other examined cases) drives the level generation. The narrative is generated as a sequence of hero and NPC actions at specific plot points; a level layout is generated as a graph so that the locations specified in the plot are visited in the right order. 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 towards the player-specified features and the narrative constraints. Game Forge uses the designer-provided or computer-generated story to guide level generation, but also accounts for player preferences as additional human input.

**AudioInSpace** [44] is a space shooter game where specific elements — the weapons and the soundtrack — are generated based on the player's in-game feedback (see Fig. 6b). The weapon's bullets are represented as particles, the position and color of which are controlled by a compositional pattern producing network (CPPN) [120] 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 [123], choosing their favorite weapon among 12 options. On the reverse, 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 new note's pitch and its duration is controlled by a second CPPN which takes the color and position of the last fired bullet and its time delay as input. 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 who is indirectly controlling the mapping between facets.

**Sonancia** [85] generates levels and their soundscapes for horror games based on a desired progression of tension (see Fig. 6c). 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 provided by a game designer [84] and acts as the blueprint which the level generator tries to adhere to. Levels are evolved so that the level's tension progression matches the intended model of tension. In the level, tension increases if there is a monster in a room along the path to the exit, and decays if there are no monsters. The generated level's tension model is used to allocate pre-authored background sounds to each room in the level. Each sound has a tension value, which can be defined by an expert designer [84] or derived from crowdsourcing [87]. Sonancia uses a hierarchical generative pipeline, starting from a proto-narrative (the desired progression of tension) to drive the level generation which in turn influences the background audio choices for each room.

**Mechanic Miner** [26] 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 only 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. Mechanic miner is another instance of a hierarchical generative pipeline, where the game rules are generated first and influence the level generation; the generated gameplay is a trivial simulation as it uses random actions, and is merely used to ensure playability of the generated game.

**Ludi** [10, 9] generates two-player adversarial games, abstract in nature and similar to checkers or tic-tac-toe. Ludi generates all relevant facets of such games, although admittedly the fact that board games of this type do not rely on visuals, audio or narrative makes orchestration more feasible. Ludi combines rules and board layout (levels) in the same "game description" which is evolved towards a plethora of desired gameplay properties. Gameplay is simulated in most evolving game descriptions (provided their rules are well-formed and not derivative of past games) 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 are parsed to assess objective properties (e.g. completion rate, game duration) but also aesthetic properties (e.g. drama, uncertainty). Unlike many other examined cases, 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.

The **Extensible Graphical Game Generator** (EGGG) [96] is an early automated programming system that generates playable user interfaces for games that are specified in a description language. The generated interfaces respect features of the ruleset, such as hiding information which is intended to be hidden. In addition, for two-player games it generates an AI player specialized to that game. EGGG is a hierarchical generator which matches rules to visuals (as the user interface) and to gameplay (as the AI opponent). This is the only instance among the examined cases where the generated AI agents are intended as opponents to the end-user rather than only for simulating gameplay as a player proxy.

The work of Karavolos *et al.* [58, 57] on **shooter game generation based on surrogate models** offers an example where the mappings between different facets are encoded as a computational model. Deep learning is applied on a custom corpus of first person shooter games, where the levels and character classes of two opposing players are used as input to predict the gameplay outcomes of the match duration and balance (in terms of number of kills scored). The mappings between the level, which is provided as an image of the top-down map, and the rules as parameters referring to the players' hit points or their weapons' accuracy could be used instead of simulations of gameplay. This allows for a fast (although not always accurate) search-based generation of levels or character classes towards desired gameplay outcomes such as a balanced long match. The system is able to make small changes to a level created by a human or generated through a constructive process, to balance a specific matchup against two character classes [58]; conversely, it can tweak the character classes' parameters to balance their match in a specific game level [57].

This is the only instance among the examined cases where gameplay simulations are replaced by a priori learned models.

In their work on **generating WarioWare-style micro-games**, Nelson and Mateas [94, 95] proposed a four-facet model that partly overlaps with the facets of Section 2, and implemented a generator of *WarioWare* [173] style games orchestrating a subset of those facets. Those four facets were: abstract mechanics (similar to the rules facet), concrete game representation (a mixture of the visual and audio facets), thematic mapping (similar to the narrative facet, plus the aspects of visuals that establish setting and meaning), and control mappings (subsumed in the rules facet). The generator takes a high-level micro-narrative provided by the user (such as a game about *chasing*), and finds a combination of game mechanics and sprites from a pre-authored set to produce the narrative. It thus follows a hierarchical process that starts from the proto-narrative facet, using ConceptNet and WordNet to decide the mechanics and names for game objects based on a player-provided verb, and then jointly searches the rules and visuals facets for a suitable content pair.

## 6 Conclusions

Throughout this chapter, we have argued that game design is a highly creative activity as designers need to consider the audiovisual presentation, the plot and its exposition, the rules and spatial layouts of a game as well as how their interrelations might influence players' perceptions and emotions while playing through it. In order to impart the creativity inherent in game design to algorithms, artificial intelligence must reach new heights and overcome a number of challenges. Throughout the history of commercial games and academic projects on content generation, several shortcuts have been taken to balance the quality of the resulting game content, in order to be appealing to end-users, with a unique and unexpected player experience when interacting with "perpetually fresh" content. Recent cases of AI-based game design, surveyed in Section 5.3, highlight such shortcuts through the use of broader databases such as Wikipedia for Data Adventures, newspaper articles for Angelina, Google's image search or autocomplete in A Rogue Dream, or player preference in AudioInSpace. Due to academic efforts in the last 10 years, a plethora of new AI algorithms have been devised for game design or game content design — and even more non-game AI algorithms have been repurposed for these tasks.

### *6.1 Future Directions*

The road ahead for algorithmic game design has several interesting directions. Indeed, research efforts in all directions will be necessary for purely algorithmic game design to be achievable.

In one direction, the interrelations between different facets must be better imparted to generative systems in order to create a cohesive game that goes beyond unrelated constituent parts. Efforts in orchestration so far feature constrained design spaces such as board games [10], arcade games [130] or adventure games [34]. Computational models so far learn patterns within a very narrow scope of games — often the same game [121] or a game prototype [56]. Recent work in combining learned patterns from different arcade games [105] is a promising step, although the lack of playable output shows that extensive work is still needed in this direction.

Another important direction is in the algorithm's ability to explain its decisions to a designer working alongside it: explainable AI is becoming a necessity in other fields such as data mining but is only now being considered as a feedback mechanism for game designers [139]. Along with the ability to explain their decisions, the generative algorithms should also impart a sense of intent, where each decision can be traced to an overall goal, style or aesthetic (generated or otherwise) [51, 77]. The ability to provide context and intent in a natural language format easily consumable by a human audience has been argued for and actualized in several creative software programs [63, 22, 20] but has only been attempted in very few AI-based game designers such as Angelina [24].

Finally, an important direction is the human-computer interaction aspect of AI-based game designers; while the focus of this chapter has been on fully automated game design, there is a significant benefit in using AI as a companion to game designers in a mixed-initiative setting [133] where both the human and the computational creator can assess, provide suggestions or take over parts of the creative process. Research in interfaces, design paradigms, or points in the creative process where human input is most needed [72, 59] can be of use in the short term for commercial game design — where human designers prefer to be in creative control — but also to identify key points where AI needs to be strengthened for a fully automated game designer in the longer term.

## *6.2 Challenges*

While there is intense interest from both the game industry and the research community for pursuing algorithmic game design, there are a number of challenges which must be tackled or circumvented in order to make tangible progress towards a creative computational game designer.

An important challenge is the questionable capability of computational models to capture patterns within content of the same facet (such as levels [121]) or between facets (such as levels and rules [56]). Despite leaps of machine learning algorithms and a reinvigorated research interest in the topic, the recent success of deep learning is in no small part due to the vast data repositories made available. Games have similarly enjoyed a boost in the number of titles released yearly as well as the size and breadth of their gameworlds, levels, lines of dialog or textures. However, data

related to games are difficult to use from a machine learning perspective for a variety of reasons such as copyright concerns or non-uniform data formats.

On the one hand, in-game data such as textures are understandably covered by strict copyrights. On the other hand, player data (such as game logs and churn patterns) are proprietary and often carefully guarded secrets as they can inform the design of future titles or patches. Game companies are also concerned that machine learning systems could overfit and unwittingly recreate content from another game. This could cause copyright infringement issues for any other company or institution that uses such a system.

On the other hand, the broad variety of games is detrimental as far as machine learning is concerned. Content such as levels can be formatted very differently from one game to the next: examples include 3D levels with overlapping floors which can not be represented as a top-down map in the same way as, for example, 2D levels in *The Legend of Zelda* [171]. Similarly, while some facets of games — such as visuals [138] — may be easy to apply machine learning to, other facets such as the game's ruleset are far less straightforward. Rulesets of different games would require extensive processing to be compatible with each other and also useful as input to a machine learning system.

### *6.3 Parting Words*

Despite the challenges ahead, the path towards automated intelligent design within games is exciting, especially for academic research in artificial intelligence and computational creativity. As AI becomes ever more relevant, its application as a creative force in the highly complex task of game design would be a milestone. An important stepping stone towards such an end would be an increased interest and commercial appeal from the game industry towards full game generation, to complement the already burgeoning academic interest. Similar to games such as *Rogue*, *Elite* or even *No Man's Sky* which pioneered level generation and succeeded based on it, commercially successful games which embrace full game generation would invigorate commercial and broader public attention towards such a task. There are important initiatives, outreach activities, and ongoing research on the topic which we expect will bring us closer to a fully orchestrated AI-based game.

## 7 Acknowledgments

An important part of this work is based on the publications of [81, 73] which introduced and refined the six creative facets of games and proposed ways of orchestrating them via AI. This chapter builds on [73], including descriptions of game facets and case studies of AI-based game design. This work is largely inspired by the contributions of researchers and developers of the respective systems cited in the paper,

particularly those highlighted in Section 5.3. Finally, I would like to thank Georgios N. Yannakakis, Mark J. Nelson, Mike Preuss and Rafael Bidarra for their contributions to [73] and also to Michael Cook and Julian Togelius for discussions on computational creativity applied to games.

## References

1. Ashlock, D., McGuinness, C.: Landscape automata for search based procedural content generation. In: Proceedings of IEEE Conference on Computational Intelligence and Games (2013)
2. Ashmore, C., Nitsche, M.: The quest in a generated world. In: Proceedings of the 2007 DiGRA Conference, pp. 503–509 (2007)
3. Barros, G.A., Togelius, J.: Exploring a large space of small games. In: Proceedings of the IEEE Conference on Computational Intelligence and Games (2014)
4. Barros, G.A.B., Green, M.C., Liapis, A., Togelius, J.: Data-driven design: A case for maximalist game design. In: Proceedings of the International Conference of Computational Creativity (2018)
5. Barros, G.A.B., Green, M.C., Liapis, A., Togelius, J.: Who killed Albert Einstein? from open data to murder mystery games. IEEE Transactions on Games (2018). Accepted
6. Barros, G.A.B., Liapis, A., Togelius, J.: Data adventures. In: Proceedings of the FDG workshop on Procedural Content Generation in Games (2015)
7. Barros, G.A.B., Liapis, A., Togelius, J.: Playing with data: Procedural generation of adventures from open data. In: Proceedings of the International Joint Conference of DiGRA and FDG (2016)
8. Blizzrd: What is Diablo III? `https://us.diablo3.com/en/game/what-is`. Accessed 26 July 2018
9. Browne, C.: Evolutionary Game Design. Springer (2011)
10. Browne, C., Maire, F.: Evolutionary game design. IEEE Transactions on Computational Intelligence and AI in Games **2**(1), 1–16 (2010)
11. Calleja, G.: In-Game: From Immersion to Incorporation. The MIT press (2011)
12. Cardamone, L., Loiacono, D., Lanzi, P.L.: Interactive evolution for the procedural generation of tracks in a high-end racing game. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 395–402. ACM (2011)
13. Cardamone, L., Yannakakis, G., Togelius, J., Lanzi, P.L.: Evolving interesting maps for a first person shooter. In: Applications of Evolutionary Computation, pp. 63–72. Springer (2011)
14. Carrier, E.: The procedural world generation of Far Cry 5. `https://blog.us.playstation.com/2018/03/22/the-procedural-world-generation-of-far-cry-5/` (2018). Accessed 26 July 2018
15. Carter, M., Gibbs, M., Harrop, M.: Metagames, paragames and orthogames: A new vocabulary. In: Proceedings of the Foundations of Digital Games Conference (2012)
16. Cheong, Y.G., Riedl, M.O., Bae, B.C., Nelson, M.J.: Planning with applications to quests and story. In: Procedural Content Generation in Games: A Textbook and an Overview of Current Research. Springer (2016)
17. Collins, K.: An introduction to procedural music in video games. Contemporary Music Review **28**(1), 5–15 (2009)
18. Collins, K.: Playing with Sound. MIT press (2013)
19. Collins, S.: Game graphics during the 8-bit computer era. SIGGRAPH Computer Graphics **32**(2), 47–51 (1998)
20. Colton, S., Valstar, M.F., Pantic, M.: Emotionally aware automated portrait painting. In: DIMEA, *ACM International Conference Proceeding Series*, vol. 349, pp. 304–311. ACM (2008)

21. Cook, M.: Would you look at that! Vision-driven procedural level design. In: Proceedings of the AIIDE Workshop on Experimental AI in Games (2015)
22. Cook, M., Colton, S.: Automated collage generation - with more intent. In: Proceedings of the International Conference on Computational Creativity (2011)
23. Cook, M., Colton, S.: A Rogue Dream: Automatically generating meaningful content for games. In: Proceedings of the AIIDE Workshop on Experimental AI in Games (2014)
24. Cook, M., Colton, S.: Ludus ex machina: Building a 3d game designer that competes alongside humans. In: Proceedings of the International Conference on Computational Creativity (2014)
25. Cook, M., Colton, S., Pease, A.: Aesthetic considerations for automated platformer design. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (2012)
26. Cook, M., Colton, S., Raad, A., Gow, J.: Mechanic miner: Reflection-driven game mechanic discovery and level design. In: Proceedings of Applications of Evolutionary Computation, vol. 7835, LNCS (2012)
27. Dormans, J., Bakkes, S.C.J.: Generating missions and spaces for adaptable play experiences. IEEE Transactions on Computational Intelligence and AI in Games **3**(3), 216–228 (2011)
28. Ekman, P.: Emotional and conversational nonverbal signals. In: Proceedings of the Sixth International Colloquium on Cognitive Science. Springer (2004)
29. Engelmore, R., Morgan, T.: Blackboard Systems. Addison-Wesley (1988)
30. Entertainment Software Association: Essential facts about the computer and video game industry report. `http://www.theesa.com/wp-content/uploads/2018/05/EF2018_FINAL.pdf` (2018). Accessed: 25 July 2018
31. Font, J., Alvarez, A., Holmberg, J., Dahlskog, S., Nolasco, C., Österman, A.: Fostering creativity in the mixed-initiative evolutionary dungeon designer. In: Proceedings of the FDG workshop on Procedural Content Generation in Games (2018)
32. Gaidon, A., Wang, Q., Cabon, Y., Vig, E.: Virtual worlds as proxy for multi-object tracking analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016)
33. Gravina, D., Liapis, A., Yannakakis, G.N.: Constrained surprise search for content generation. In: Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG) (2016)
34. Green, M.C., Barros, G.A.B., Liapis, A., Togelius, J.: Data agent. In: Proceedings of the Foundations of Digital Games Conference (2018)
35. Grimshaw, M.: Sound and immersion in the first-person shooter. International Journal of Intelligent Games & Simulation **5**(1) (2008)
36. Guzdial, M., Harrison, B., Li, B., Riedl, M.O.: Crowdsourcing open interactive narrative. In: Proceedings of the Foundations of Digital Games Conference (2015)
37. Guzdial, M., Riedl, M.: Game level generation from gameplay videos. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (2016)
38. Hartsook, K., Zook, A., Das, S., Riedl, M.O.: Toward supporting stories with procedurally generated game worlds. In: Proceedings of the IEEE Conference on Computational Intelligence in Games (2011)
39. Hastings, E.J., Guha, R.K., Stanley, K.O.: Automatic content generation in the Galactic Arms Race video game. IEEE Transactions on Computational Intelligence and AI in Games **1**(4), 245–263 (2009)
40. Henein, M.: 6 ways 3d audio can expand gaming experiences. Gamasutra article (2013)
41. Holmgård, C., Liapis, A., Togelius, J., Yannakakis, G.N.: Evolving personas for player decision modeling. In: Proceedings of the IEEE Conference on Computational Intelligence and Games (2014)
42. Holtar, N.I., Nelson, M.J., Togelius, J.: Audioverdrive: Exploring bidirectional communication between music and gameplay. In: Proceedings of the International Computer Music Conference, pp. 124–131 (2013)

43. Niels van Hoorn Julian Togelius, D.W., Schmidhuber, J.: Robust player imitation using multiobjective evolution. In: Proceedings of the IEEE Congress on Evolutionary Computation (2009)
44. Hoover, A.K., Cachia, W., Liapis, A., Yannakakis, G.N.: AudioInSpace: exploring the creative fusion of generative audio, visuals and gameplay. In: Evolutionary and Biologically Inspired Music, Sound, Art and Design, vol. 9027, LNCS. Springer (2015)
45. Howlett, A., Colton, S., Browne, C.: Evolving pixel shaders for the prototype video game Subversion. In: AISB 2010 Symposium on AI and Games (2010)
46. Hunicke, R., Leblanc, M., Zubek, R.: MDA: A formal approach to game design and game research. In: Proceedings of AAAI Workshop on the Challenges in Games AI (2004)
47. Isaksen, A., Gopstein, D., Togelius, J., Nealen, A.: Exploring game space of minimal action games via parameter tuning and survival analysis. IEEE Transactions on Computational Intelligence and AI in Games (2017)
48. Jain, R., Isaksen, A., Holmgård, C., Togelius, J.: Autoencoders for level generation, repair, and recognition. In: Proceedings of the ICCC Workshop on Computational Creativity and Games (2016)
49. Järvinen, A.: Gran stylissimo: The audiovisual elements and styles in computer and video games. In: CGDC Conference (2002)
50. Järvinen, A.: Games without frontiers: Theories and methods for game studies and design. Ph.D. thesis, University of Tampere (2008)
51. John, B.E., Kieras, D.E.: The GOMS family of user interface analysis techniques: Comparison and contrast. ACM Transactions on Computer-Human Interaction **3**(4), 320–351 (1993)
52. Jong, K.A.D.: Evolutionary computation - a unified approach. MIT Press (2006)
53. Joyce, M.: Of Two Minds: Hypertext Pedagogy and Poetics. University of Michigan Press (1995)
54. Juul, J.: Half Real. Videogames between Real Rules and Fictional Worlds. The MIT Press (2005)
55. Kambhampati, S., Knoblock, C., Yang, Q.: Planning as refinement search: A unified framework for evaluating the design tradeoffs in partial order planning. Artificial Intelligence **76**(1-2), 167–238 (1995)
56. Karavolos, D., Liapis, A., Yannakakis, G.N.: Learning the patterns of balance in a multiplayer shooter game. In: Proceedings of the FDG workshop on Procedural Content Generation in Games (2017)
57. Karavolos, D., Liapis, A., Yannakakis, G.N.: Pairing character classes in a deathmatch shooter game via a deep-learning surrogate model. In: Proceedings of the FDG Workshop on Procedural Content Generation (2018)
58. Karavolos, D., Liapis, A., Yannakakis, G.N.: Using a surrogate model of gameplay for automated level design. In: Proceedings of the IEEE Conference on Computational Intelligence and Games (2018)
59. Karimi, P., Grace, K., Maher, M.L., Davis, N.: Evaluating creativity in computational co-creative systems. In: Proceedings of the International Conference on Computational Creativity (2018)
60. Kelly, J., Botea, A., Koenig, S.: Offline planning with hierarchical task networks in video games. In: Proceedings of the 4th Artificial Intelligence and Interactive Digital Entertainment Conference (2008)
61. Kowalski, J., Liapis, A., Żarczyński, Ł.: Mapping chess aesthetics onto procedurally generated chess-like games. In: Applications of Evolutionary Computation. Springer (2018)
62. Kowalski, J., Szykula, M.: Evolving chess-like games using relative algorithm performance profiles. In: Proceedings of Evolutionary Applications, vol. 9597, LNCS, pp. 574–589. Springer (2016)
63. Krzeczkowska, A., El-Hage, J., Colton, S., Clark, S.: Automated collage generation - with intent. In: Proceedings of the International Conference on Computational Creativity, pp. 36–40 (2010)

64. Lagae, A., Lefebvre, S., Cook, R., DeRose, T., Drettakis, G., Ebert, D., Lewis, J., Perlin, K., Zwicker, M.: State of the art in procedural noise functions. In: Eurographics 2010 State of the Art Reports (2010)
65. Laird, J., Lent, M.V.: Human-level AI's killer application: Interactive computer games. AI Magazine **22** (2001)
66. Lehman, J., Risi, S., Clune, J.: Creative generation of 3d objects with deep learning and innovation engines. In: Proceedings of the International Conference on Computational Creativity (2016)
67. Liapis, A.: Exploring the visual styles of arcade game assets. In: Proceedings of Evolutionary and Biologically Inspired Music, Sound, Art and Design. Springer (2016)
68. Liapis, A.: Recomposing the pokémon color palette. In: Applications of Evolutionary Computation. Springer (2018)
69. Liapis, A., Holmgård, C., Yannakakis, G.N., Togelius, J.: Procedural personas as critics for dungeon generation. In: Applications of Evolutionary Computation, vol. 9028, LNCS. Springer (2015)
70. Liapis, A., Martínez, H.P., Togelius, J., Yannakakis, G.N.: Adaptive game level creation through rank-based interactive evolution. In: Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG) (2013)
71. Liapis, A., Martínez, H.P., Togelius, J., Yannakakis, G.N.: Transforming exploratory creativity with DeLeNoX. In: Proceedings of the Fourth International Conference on Computational Creativity, pp. 56–63 (2013)
72. Liapis, A., Smith, G., Shaker, N.: Mixed-initiative content creation. In: N. Shaker, J. Togelius, M.J. Nelson (eds.) Procedural Content Generation in Games: A Textbook and an Overview of Current Research, pp. 195–214. Springer (2016)
73. Liapis, A., Yannakakis, G.N., Nelson, M.J., Preuss, M., Bidarra, R.: Orchestrating game generation. IEEE Transactions on Games **11**(1), 48–68 (2019)
74. Liapis, A., Yannakakis, G.N., Togelius, J.: Neuroevolutionary constrained optimization for content creation. In: Proceedings of IEEE Conference on Computational Intelligence and Games, pp. 71–78 (2011)
75. Liapis, A., Yannakakis, G.N., Togelius, J.: Adapting models of visual aesthetics for personalized content creation. IEEE Transactions on Computational Intelligence and AI in Games **4**(3), 213–228 (2012)
76. Liapis, A., Yannakakis, G.N., Togelius, J.: Limitations of choice-based interactive evolution for game level design. In: Proceedings of AIIDE Workshop on Human Computation in Digital Entertainment (2012)
77. Liapis, A., Yannakakis, G.N., Togelius, J.: Designer modeling for personalized game content creation tools. In: Proceedings of the AIIDE Workshop on Artificial Intelligence & Game Aesthetics (2013)
78. Liapis, A., Yannakakis, G.N., Togelius, J.: Sentient sketchbook: Computer-aided game level authoring. In: Proceedings of the Foundations of Digital Games Conference, pp. 213–220 (2013)
79. Liapis, A., Yannakakis, G.N., Togelius, J.: Sentient world: Human-based procedural cartography. In: Proceedings of Evolutionary and Biologically Inspired Music, Sound, Art and Design (EvoMusArt), vol. 7834, LNCS, pp. 180–191. Springer (2013)
80. Liapis, A., Yannakakis, G.N., Togelius, J.: Towards a generic method of evaluating game levels. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (2013)
81. Liapis, A., Yannakakis, G.N., Togelius, J.: Computational game creativity. In: Proceedings of the International Conference on Computational Creativity (2014)
82. Liapis, A., Yannakakis, G.N., Togelius, J.: Designer modeling for sentient sketchbook. In: Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG) (2014)
83. Liapis, A., Yannakakis, G.N., Togelius, J.: Constrained novelty search: A study on game content generation. Evolutionary Computation **23**(1), 101–129 (2015)

84. Lopes, P., Liapis, A., Yannakakis, G.N.: Targeting horror via level and soundscape generation. In: Proceedings of the AAAI Artificial Intelligence for Interactive Digital Entertainment Conference (2015)
85. Lopes, P., Liapis, A., Yannakakis, G.N.: Framing tension for game generation. In: Proceedings of the International Conference on Computational Creativity (2016)
86. Lopes, P., Liapis, A., Yannakakis, G.N.: A holistic approach for semantic-based game generation. In: Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG) (2016)
87. Lopes, P., Liapis, A., Yannakakis, G.N.: Modelling affect for horror soundscapes. IEEE Transactions of Affective Computing (2017)
88. Machado, T., Bravi, I., Wang, Z., Nealen, A., Togelius, J.: Shopping for game mechanics. In: Proceedings of the FDG Workshop on Procedural Content Generation (2016)
89. Maiberg, E.: 'No Mans Sky' is like 18 quintillion bowls of oatmeal. `https://www.vice.com/en_us/article/nz7d8q/no-mans-sky-review` (2016). Accessed 13 May 2019
90. Marin, D., G., V.D., Lopez, A.M.: Learning appearance in virtual scenarios for pedestrian detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2010)
91. Mateas, M., Stern, A.: Procedural authorship: A case-study of the interactive drama Façade. In: Digital Arts and Culture (2005)
92. McCoy, J., Treanor, M., Samuel, B., Reed, A.A., Mateas, M., Wardrip-Fruin, N.: Prom Week: Designing past the game/story dilemma. In: Proceedings of the Foundations of Digital Games Conference (2013)
93. Nelson, M., Smith, A.M.: ASP with applications to mazes and levels. In: N. Shaker, J. Togelius, M.J. Nelson (eds.) Procedural Content Generation in Games: A Textbook and an Overview of Current Research, pp. 143–157. Springer (2016)
94. Nelson, M.J., Mateas, M.: Towards automated game design. In: AI*IA 2007: Artificial Intelligence and Human-Oriented Computing, pp. 626–637. Springer (2007). Lecture Notes in Computer Science 4733
95. Nelson, M.J., Mateas, M.: An interactive game-design assistant. In: Proceedings of the 13th International Conference on Intelligent User Interfaces, pp. 90–98 (2008)
96. Orwant, J.: EGGG: Automated programming for game generation. IBM Systems Journal **39**(3.4), 782–794 (2000)
97. Pedersen, C., Togelius, J., Yannakakis, G.N.: Modeling player experience in super mario bros. In: Proceedings of the international conference on Computational Intelligence and Games, pp. 132–139 (2009)
98. Pell, B.: A strategic metagame player for general chess-like games. Computational Intelligence **12**(1), 177–198 (1996)
99. Perez, D., Togelius, J., Samothrakis, S., Rohlfshagen, P., Lucas, S.M.: Automated map generation for the physical traveling salesman problem. IEEE Transactions on Evolutionary Computation **18**(5), 708–720 (2014)
100. Riedl, M., Young, R.: Narrative planning: balancing plot and character. Journal of Artificial Intelligence Research **39**(1), 217268 (2010)
101. Riedl, M.O.: Narrative generation: Balancing plot and character. Ph.D. thesis, North Carolina State University (2004)
102. Risi, S., Lehman, J., D'Ambrosio, D.B., Hall, R., Stanley, K.O.: Petalz: Search-based procedural content generation for the casual gamer. IEEE Transactions on Computational Intelligence and AI in Games **8**(3), 244–255 (2016)
103. Robertson, J., Young, R.M.: Automated gameplay generation from declarative world representations. In: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, pp. 72–78 (2015)
104. Ryan, M.L.: Beyond myth and metaphor: The case of narrative in digital media. Game Studies **1**(1) (2001)
105. Sarkar, A., Cooper, S.: Blending levels from different games using LSTMs. In: Proceedings of the AIIDE workshop on Experimental AI in Games Workshop (2018)

106. Scirea, M., Bae, B.C., Cheong, Y.G., Nelson, M.: Evaluating musical foreshadowing of videogame narrative experiences. In: Proceedings of Audio Mostly (2014)
107. Shaker, N., Liapis, A., Togelius, J., Lopes, R., Bidarra, R.: Constructive generation methods for dungeons and levels. In: Procedural Content Generation in Games: A Textbook and an Overview of Current Research, pp. 31–55. Springer (2016)
108. Shaker, N., Togelius, J., Nelson, M.J.: Procedural Content Generation in Games: A Textbook and an Overview of Current Research. Springer (2016)
109. Short, T.X., Adams, T.: Procedural Generation in Game Design. CRC Press (2017)
110. Sicart, M.: Defining game mechanics. Game Studies **8** (2008)
111. Sicart, M.: Digital games as ethical technologies. In: The Philosophy of Computer Games, pp. 101–124. Springer (2012)
112. Smelik, R.M., Tutenel, T., de Kraker, K.J., Bidarra, R.: A declarative approach to procedural modeling of virtual worlds. Computers & Graphics **35**(2), 352–363 (2011)
113. Smith, A.M., Andersen, E., Mateas, M., Popović, Z.: A case study of expressively constrainable level design automation tools for a puzzle game. In: Proceedings of the Foundations of Digital Games Conference, pp. 156–163 (2012)
114. Smith, A.M., Butler, E., Popović, Z.: Quantifying over play: Constraining undesirable solutions in puzzle design. In: Proceedings of the Foundations of Digital Games Conference (2013)
115. Smith, A.M., Mateas, M.: Variations Forever: Flexibly generating rulesets from a sculptable design space of mini-games. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games (2010)
116. Smith, A.M., Mateas, M.: Answer set programming for procedural content generation: A design space approach. IEEE Transactions on Computational Intelligence and AI in Games **3**(3), 187–200 (2011)
117. Smith, G., Whitehead, J., Mateas, M.: Tanagra: Reactive planning and constraint solving for mixed-initiative level design. IEEE Transactions on Computational Intelligence and AI in Games **3**(3) (2011)
118. Snodgrass, S., nón, S.O.: Experiments in map generation using markov chains. In: Proceedings of the Foundations of Digital Games Conference (2014)
119. Sorenson, N., Pasquier, P., DiPaola, S.: A generic approach to challenge modeling for the procedural creation of video game levels. IEEE Transactions on Computational Intelligence and AI in Games **3**(3), 229–244 (2011)
120. Stanley, K.O.: Exploiting regularity without development. In: Proceedings of the AAAI Fall Symposium on Developmental Systems. AAAI Press (2006)
121. Summerville, A., Behrooz, M., Mateas, M., Jhala, A.: The learning of zelda: Data-driven learning of level topology. In: Proceedings of the Foundations of Digital Games Conference (2015)
122. Summerville, A., Snodgrass, S., Guzdial, M., Holmgård, C., Hoover, A.K., Isaksen, A., Nealen, A., Togelius, J.: Procedural content generation via machine learning (PCGML). IEEE Transactions on Games (2018)
123. Takagi, H.: Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. Proceedings of the IEEE **89**(9), 1275–1296 (2001). Invited Paper
124. Tate, A.: Generating project networks. In: Proceedings of the International Joint Conference on Artificial Intelligence, p. 888893 (1977)
125. Togelius, J., De Nardi, R., Lucas, S.M.: Towards automatic personalised content creation for racing games. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games, pp. 252–259 (2007)
126. Togelius, J., Nelson, M.J., Liapis, A.: Characteristics of generatable games. In: Proceedings of the FDG Workshop on Procedural Content Generation (2014)
127. Togelius, J., Schmidhuber, J.: An experiment in automatic game design. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games (2008)
128. Togelius, J., Shaker, N., Dormans, J.: Grammars and l-systems with applications tovegetation and levels. In: N. Shaker, J. Togelius, M.J. Nelson (eds.) Procedural Content Generation in Games: A Textbook and an Overview of Current Research, pp. 73–98. Springer (2016)

129. Togelius, J., Yannakakis, G.N., Stanley, K.O., Browne, C.: Search-based procedural content generation: A taxonomy and survey. IEEE Transactions on Computational Intelligence and AI in Games **3**(3) (2011)
130. Treanor, M., Blackford, B., Mateas, M., Bogost, I.: Game-O-Matic: Generating videogames that represent ideas. In: Proceedings of the FDG Workshop on Procedural Content Generation (2012)
131. Tutenel, T., van der Linden, R., Kraus, M., Bollen, B., Bidarra, R.: Procedural filters for customization of virtual worlds. In: Proceedings of the FDG Workshop on Procedural Content Generation. ACM (2011)
132. Yabsley, A.: Back to the 8-bit: A study of electronic music counter-culture. `http://www.gamemusic4all.com/back-to-the-8-bit/`. Accessed 27 July 2018
133. Yannakakis, G.N., Liapis, A., Alexopoulos, C.: Mixed-initiative co-creativity. In: Proceedings of the Foundations of Digital Games Conference (2014)
134. Yannakakis, G.N., Togelius, J.: Experience-driven procedural content generation. IEEE Transactions on Affective Computing **99** (2011)
135. Yannakakis, G.N., Togelius, J.: Artificial Intelligence and Games. Springer (2018). `http://gameaibook.org`
136. Young, R.M.: An overview of the mimesis architecture: Integrating intelligent narrative control into an existing gaming environment. In: The Working Notes of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment (2001)
137. Young, R.M.: Story and discourse: A bipartite model of narrative generation in virtual worlds. Interaction Studies **8**(2), 177–208 (2007)
138. Zhang, X., Zhan, Z., Holtz, M., Smith, A.M.: Crawling, indexing, and retrieving moments in videogames. In: Proceedings of the Foundations of Digital Games Conference (2018)
139. Zhu, J., Liapis, A., Risi, S., Bidarra, R., Youngblood, G.M.: Explainable ai for designers: A human-centered perspective on mixed-initiative co-creation. In: Proceedings of the IEEE Conference on Computational Intelligence and Games (2018)
140. Zook, A., Riedl, M.O.: Automatic game design via mechanic generation. In: Proceedings of the AAAI Conference on Artificial Intelligence (2014)

## Referenced Games

141. Apogee: Wolfenstein 3D (1992)
142. Atari: Pong (1972)
143. Bethesda: The Elder Scrolls IV: Oblivion (2006)
144. Bethesda: Fallout 3 (2008)
145. Bethesda: The Elder Scrolls V: Skyrim (2011)
146. Bioware: Mass Effect (2007)
147. Blizzard: StarCraft (1998)
148. Blizzard: Warcraft II (1999)
149. Blizzard: Diablo III (2012)
150. Braben, D., Bell, I.: Elite (1984)
151. Brace Yourself Games: Crypt of the NecroDancer (2015)
152. Brøderbund: Prince of Persia (1989)
153. Colossal Order: Cities: Skylines (2015)
154. Electronic Arts: Brütal Legend (2009)
155. Electronic Arts: Madden NFL 18 (2017)
156. Epic Games: Gears of War (2006)
157. Frictional Games: Amnesia: The Dark Descent (2010)
158. Game Freak: Pokémon Red and Blue (1996)
159. Gearbox Software: Borderlands (2009)

160. Harmonix: Guitar Hero (2005)
161. Hello Games: No Man's Sky (2016)
162. Infinity Ward: Call of Duty (2003)
163. Infocom: ZORK (1980)
164. Key, E., Kanaga, D.: Proteus (2013)
165. King: Candy Crush Saga (2012)
166. Looking Glass Studios: Thief: The Dark Project (1998)
167. Maxis: The Sims (2000)
168. Mike Bithell: Thomas Was Alone (2012)
169. Namco: Pac-Man (1980)
170. Nintendo: Super Mario Bros. (1985)
171. Nintendo: The Legend of Zelda (1986)
172. Nintendo: Super Smash Bros. (1999)
173. Nintendo: WarioWare (2003)
174. Pajitnov, A.: Tetris (1984)
175. Paradox Interactive: Stellaris (2016)
176. Playdead: Limbo (2010)
177. PopCap Games: Bejeweled (2001)
178. Rockstar Games: L.A. noire (2011)
179. Taito: Arkanoid (1986)
180. Toy, M., Wichman, G.: Rogue (1980)
181. Ubisoft: Tom Clancy's EndWar (2008)
182. Ubisoft: Assassin's Creed II (2009)
183. Ubisoft: Far Cry 5 (2018)
184. Unity Technologies: Unity (2005)
185. Warner Bros: Middle-Earth: Shadow of Mordor (2014)