Design Metaphors for Procedural Content Generation in Games

Rilla Khaled
Department of Digital Games
University of Malta
rilla.khaled@um.edu.mt

Mark J. Nelson
Center for Computer Games Research, ITU Copenhagen
mjas@itu.dk

Pippin Barr
Department of Digital Games
University of Malta
pippin.barr@gmail.com

ABSTRACT

Procedural content generation (PCG), the algorithmic creation of game content with limited or indirect user input, has much to offer to game design. In recent years, it has become a mainstay of game AI, with significant research being put towards the investigation of new PCG systems, algorithms, and techniques. But for PCG to be absorbed into the practice of game design, it must be contextualised within design-centric as opposed to AI or engineering perspectives. We therefore provide a set of design metaphors for understanding potential relationships between a designer and PCG. These metaphors are: TOOL, MATERIAL, DESIGNER, and EXPERT. By examining PCG through these metaphors, we gain the ability to articulate qualities, consequences, affordances, and limitations of existing PCG approaches in relation to design. These metaphors are intended both to aid designers in understanding and appropriating PCG for their own contexts, and to advance PCG research by highlighting the assumptions implicit in existing systems and discourse.

Author Keywords

Game Design; Metaphor; Game AI; Procedural Content Generation; Adaptive Games

ACM Classification Keywords

H.5.m Information Interfaces and Presentation: Miscellaneous; I.2.1 Artificial Intelligence: Applications and Expert Systems—*Games*; J.6 Computer-aided Engineering: Computer-aided design (CAD)

INTRODUCTION

Procedural content generation (PCG) has been defined as "the algorithmical creation of game content with limited or indirect user input" [38], and generally rests on the assumption that the generated content should mimic human-authored content [34]. As such, PCG is intrinsically linked to game design, of which the creation of content is a major part.

In recent years, PCG has become an integral part of the broader domain of game AI. But since the research agendas and approaches in the PCG literature tend to be technology-oriented, it is less clear how PCG systems and techniques can be adopted by game designers who are not as technologically motivated. Indeed, uptake of PCG systems and techniques by game designers has been slow [46]. We suggest that this is partly because PCG has not yet been conceptualised from a design-centric perspective, making it difficult for designers to envision how to use these concepts, techniques, and innovations in their practice. To reduce the distance between design and engineering perspectives on PCG, we propose a set of metaphors for understanding the nature of potential relationships between a designer and PCG that are familiar from day-to-day experience. These metaphors are: TOOL, MATERIAL, DESIGNER, and EXPERT.

The contributions of this work are as follows. First, by presenting PCG through familiar metaphors, we are able to highlight qualities, consequences, affordances, and limitations that suggest future directions for PCG research.[1] Second, in explaining these metaphors we articulate a design-centric perspective on what state-of-the-art PCG offers to the design community, making the technologies more accessible and providing ways for designers to position PCG in relation to their own practice. Finally, our work represents a bridge between the HCI and PCG communities, which is necessary for the two fields to benefit from each other's expertise.

BACKGROUND

The use of PCG in games dates back to the early 1980s, where it first appeared in the dungeon crawler *Rogue* [41]. *Rogue* popularised the use of PCG for increasing game replayability through automatic generation of environments. PCG was also used as a solution for memory constraints: instead of needing to store game content such as terrain and other assets, PCG algorithms were used to generate such content only when needed. As storage capacity increased over the years, memory conserving qualities of PCG were backgrounded and now commercial uses of PCG have tended to target automating or assisting with the development of game content in order to

[1] That is, we hope to take one step towards fostering a *critical technical practice* in PCG in the sense of Agre [1]. As Agre argues, AI systems bake into their designs assumptions about what they do "intelligently", as well as where, how, and with whom they do it. Critical analysis of these systems should not just be a matter for external commentators from "non-technical" perspectives such as philosophy, sociology, and design, it should be an integral part of rigorous AI research. As Boehner *et al.* [6] note, resolving recurring technical impasses sometimes requires critical attention to the core metaphors of a technical field.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CHI 2013, April 27–May 2, 2013, Paris, France.
Copyright 2013 ACM 978-1-4503-1899-0/13/04...$15.00.

reduce designer effort. For example, *SpeedTree* is a system that procedurally generates large numbers of trees [14].

While most contemporary industry uses of PCG focus on generating graphical assets, recent years have seen a surge of interest in PCG techniques within the game AI community. Researchers from the intelligent tutoring systems, affective computing, and expressive AI communities have all begun exploring more complex and sophisticated ways to use PCG within the game development process. Recent game AI research on PCG can roughly be categorised into two trends.

The first trend concerns the development of systems to facilitate the game development process. This includes work on systems to assist game designers in their workflow [35, 37, 42]. It also includes systems for bringing people from non-gaming sectors into the game development process, ranging from tools to assist novice game designers [25, 42] to ones that enable subject-matter experts without game design experience to contribute to serious game design [4, 13].

The second trend concerns adapting game content to the play patterns, preferences, skills, and experience levels of players, both offline (prior to play) and online (during play). This work is often driven by an interest in automating the process of providing players entertaining experiences by personalising game features and challenges to be more "fun" for them [39]. Adaptive and personalisation-focused PCG has also been positioned as a solution to providing players with what they need in order to achieve learning or training goals in serious games [21, 26]. Indeed, in some contexts, personalisation is a prerequisite: individuals with dyslexia learning how to read require teaching that takes their specific reading difficulties into account. For both entertainment and serious games, personalisation-focused PCG has also been positioned as a way of increasing the replay value of games, as it can enable ongoing game adaptation in accordance with a player's changing skills and expectations [21].

Similar motivations concerning design can be seen in recent interaction design (IxD) and HCI research where developing tools to support designers and other stakeholders during development has become a common approach. Indeed, a deep focus on the user, and specifically on different facets of human experience, arguably characterises the shift between dominant HCI paradigms over the last 30 years [12]. From an overarching perspective, the motivations underlying current PCG research can be aligned within the broader agenda of IxD and HCI, even if the approaches and implementations have differed. Despite this overlap in mutual interests, the communities have remained separate. Research within the PCG and game AI community has mostly focused on novel techniques and the implementation of systems, algorithms, and processes for PCG. PCG research has chiefly been a technology-focused endeavour, targeted at a future in which designers naturally incorporate PCG into their practice and workflow. Perspectives on how to integrate these techniques and tools within a wider design ecosystem that is not primarily engineering focused, and within designers' workflows in particular, have not been especially forthcoming. Current PCG research rarely considers how PCG tools and techniques are used "in the wild" or in contexts outside engineering for instance. Consequently, scholars have noted that industry uptake of novel PCG techniques has been slow [46]. We suggest that this is especially because PCG has not yet been conceptualised from a design-centric perspective, making it difficult for designers to envision how these concepts, techniques, and innovations might fit with their everyday practices and approaches to design.

PCG DESIGN METAPHORS

For people from different communities to understand and collaborate with each other, they need a shared language and conceptual framework. Metaphors have often played this role in HCI. Lakoff and Johnson argue that metaphors are fundamental to our basic sense-making processes, stating that "our ordinary conceptual system, in terms of which we both think and act, is fundamentally metaphorical in nature" [19]. In a usability context, Carroll states that using interface metaphors "seeks to increase the initial familiarity of actions, procedures and concepts by making them similar to actions, procedures and concepts that are already known" [7]. From a design perspective, Lawson and Loke use metaphorical roles to explain possible dynamics between a system and a designer: learner, informer, critic, collaborator, and initiator [20].

Following in this tradition, we propose a set of metaphors for understanding the nature of potential relationships between a designer and PCG. We developed these metaphors by surveying the existing PCG literature, and identifying how PCG systems relate to the design process. This includes looking at how researchers position their systems, which is often implicit, and how their research and goals connect to existing interaction design and game design perspectives. It is worth noting that these metaphors are derived from the PCG literature rather than ethnographic study with game designers: our goal here is to re-conceptualise the PCG literature in a design-sensitive manner.[2] Each metaphor characterises PCG by its relation to design processes, rather than by its technical approaches, aiming to foreground the contextual aspect of who and what PCG is for, rather than the engineering approach of how it works.

The metaphors we propose are: TOOL, MATERIAL, DESIGNER, and EXPERT. To date, TOOL has often been used as an all-encompassing metaphor: *PCG tool* and *PCG system* are common generic descriptors meaning effectively "a thing with PCG". We feel that this has reduced its explanatory power, as researchers have very different ideas of what a TOOL is, and how (and by whom) it will be used. We pare back the notion of TOOL to a more specific meaning, and identify supporting metaphors to form an ecosystem of interacting meanings and contexts that describe how designers can relate to PCG systems. Crucially, these metaphors do not form a taxonomy of PCG systems or technologies, but a set of lenses through which to view PCG in its varying contexts. Although many systems are positioned by their creators in a way that comfortably fits within one of these metaphors,

[2]We will conduct future empirical research on how game designers approach PCG themselves, but the present work is not a study of how game designers view the subject.

many systems can be approached and analysed through all four metaphors. Indeed, looking at a system through lenses other than the ones employed by its creator can be a productive way of understanding its potential and limitations.

To avoid confusion, when we refer to "designers" and "domain experts" we mean *human* designers and domain experts; likewise by "creators", we mean the *human* developers of PCG systems. All references to the metaphors themselves will make use of small caps, allowing us to differentiate between, for example, designers and DESIGNERS.

TOOL

PCG has most often been positioned as a TOOL to aid game design. As an example, Sullivan *et al.* describe the system *QuestBrowser*:

> To truly achieve playable quests, the designer will need to be able to create a large number of possible solutions for each quest. We have created the QuestBrowser brainstorming tool ... to help designers with this challenge as well as help alleviate the difficulties in thinking up multiple interesting solutions for each quest. [37]

Tools are commonly understood as devices or instruments manipulated for the purpose of achieving specific goals, changing the environment and acting as an extension of the user [3]. TOOLS, then, can be understood as devices or instruments manipulated for the purpose of achieving specific game design goals, that enhance and extend a designer's abilities. Consistent with how tools are intended to enable their users to perform tasks more efficiently and effectively, TOOLS should also be designed to optimise the performance of their designer users.

TOOLS have been developed to assist designers in a range of contexts. These include assisting in the design of game mechanics [42], the design of game levels [35], and the authoring of game quests [37]. They have mostly been used in an offline context but can also be used online. For example, *Virtual Iraq*, a game-like PTSD therapy tool premised on exposure therapy, enables therapists to edit simulations in real-time by adding and removing world elements as they are being experienced by players [29].

St. Amant and Horton point out that within HCI the metaphor of "tool" has often been used as a catch-all to describe systems with little in common other than being interactive software [36]. To clarify what is invoked when using the tool metaphor, they draw a distinction between *effective tools* that produce a persistent effect and *instruments* that provide information. Instrumentation may also be built into effective tools, such as in *Tanagra*, a procedural level generation tool for platformer games [35]. In terms of AI terminology, *Tanagra* is described as "mixed initiative": its levels are created over a series of iterative cycles requiring input and modifications from both a human designer and procedural support. It can be used as an *effective tool*, allowing users to shape platforms by directly manipulating their shapes, and as an *instrument*, providing visual feedback if the user makes a change that leads to an unplayable level.

As mentioned earlier, tools often effect change. Both the tool and its user are *active* with regards to causing such change, but at the same time, tools are generally understood to lack agency. As such, tool users are expected to accept responsibility for the results of tool use, as negatively illustrated by the proverb, "a bad workman blames his tools". Taking into consideration the computational qualities of TOOLS, scholars have argued that computers still do not meet the criteria for being moral agents [10]. Suppose, for example, that a tool such as Hullett and Mateas's system for generating levels for emergency rescue training games [13] generated a training scenario that emphasised incorrect pedagogical objectives. While any problems incurred through the use of this scenario would partly be the fault of the creators of the tool, they would also be the responsibility of the human user who made the scenario available for play, as this user should have been monitoring the output.[3] Although a TOOL may generate confusing, broken, or substandard output, the responsibility for whether or not to make use of that output, and indeed how to use the tool largely rests with the designer using it.

Cast in this light, part of the designer's task in using a TOOL is to determine its suitability for the design goal in question, and to monitor the results and consequences of its use. This is easier to control in offline uses, where the design tasks and processes the PCG contributes to take place prior to the game being played. In such cases, the designer can more easily intervene prior to actual gameplay. In online uses of TOOLS, where content generation takes place during play, designers should incorporate phases of monitoring post-generation in case further modification is required.

Tool use is inseparable from goal-driven activity and instrumentality. However, a strict focus on instrumentality in understanding how designers can use TOOLS would be misleading. Some researchers have advocated more open, explorative ways of engaging with TOOLS, such as emphasising *instrument* use to provide insight into the space and activity of game design. Treanor *et al.*, for example, developed *Game-O-Matic*, a game authoring tool intended for nontechnical users to rapidly create editorial games [42]. From a user-defined concept map depicting relationships between actors, *Game-O-Matic* generates multiple simple yet rhetorically meaningful arcade-style editorial games. By automatically creating varied games, *Game-O-Matic* can serve as a brainstorming tool for novice editorial game designers, allowing them to explore alternate gameplay interpretations of their input.

A commonly observed quality of tools is that a familiar tool "disappears" from immediate awareness, becoming part of the user and part of the task [3], and "appears" again only when a problem is encountered. Creators of TOOLS need to be aware of "disappearance" as a quality in determining what kinds of functionality to support in TOOLS. If creators choose not to pursue disappearance, they should aim to make TOOL interfaces and feedback well-signposted and informative in line with the needs and interests of designers.

[3] Within the PCG research community, creators of TOOLS also often act as their users. Of course, this does not mean that a user of a TOOL must be its creator.

Critically, creators of TOOLS should aim to develop tools that can be integrated within a game designer's *practice*. Of course, designers also often adapt to new tools, but what is important is that designers are included and considered in the process of TOOL development. TOOLS should support ends that designers are genuinely invested in achieving, enable design behaviours that designers already adopt or are open to adopting, and minimise costs of use in a way that empowers the designer. If new approaches to design cannot be appropriated and integrated within existing contexts their uptake will be low and they will remain relegated to research contexts.

MATERIAL

PCG has also been presented as a MATERIAL in the context of game development. The following excerpt from a *SpeedTree* support page underscores the presence of this metaphor in commercial development:

> To ensure that materials are imported as seen in the Modeler, you must process the materials with this utility. [15]

Materials are generally understood as physical substances with no specific form, which can be shaped, modified, and manipulated according to need [43]. MATERIALS, therefore, are dynamic, reconfigurable, procedurally generated substances that can be deployed and molded by the game designer. Unlike more familiar materials, MATERIALS can also perform or be the subject of computation. As Vallgårda and Redstrom have argued in the context of interaction design, computers can and should be considered as materials that happen to compute and to change state [43].

Like *SpeedTree*, *World Machine* generates a specific kind of MATERIAL: a reactive, procedural space of sculptable terrains. These can be modified by a designer or used directly as the landscape of the game world [30]. While it can be viewed as a TOOL in the sense of design support, *World Machine* provides designers with actual implementations of MATERIALS, enabling them to mold procedural spaces. The weapons generated in the game *Borderlands* provide another example of MATERIALS: each weapon provides a different combination of properties [11]. But MATERIALS need not just generate "things" that exist at the ontological level of the game world. Procedurally generated levels, such as those used in *Rogue* and *Rogue*-likes, can be considered MATERIALS, as game levels may also be understood as game substances. Unlike the level building tools discussed earlier, these levels do not require further manipulation from a designer.

Physical materials are often used in combination to create composite materials that give rise to new qualities, properties, or experiences. In the context of game development, a designer may combine one or more MATERIALS to create a particular game experience. The play experience of *Borderlands*, for example, is a composite partly comprised of game challenges combined with the range of MATERIAL weapons the player can access. As Smith points out, this affects the dynamics of the overall game [33].

In contrast to tools, materials are typically passive: they are acted on. As such, MATERIALS are often generated offline prior to play for later shaping and modification by a designer. Alternatively, generation is performed in a just-in-time, possibly online manner, as a component of a larger game experience. In these cases, designers have often never experienced the full range of possible MATERIALS, as these change from one play session to the next. In both cases, any responsibility relating to the use of a MATERIAL remains with the designer. Part of the shaping, modifying, and manipulating designers perform also extends to decisions concerning MATERIAL use.

Integral to the conceptualisation of materials are their properties, used for description, comparison, and classification. As Vallgårda notes, defining the properties of a given material is not just a matter of accuracy, but requires acknowledgement of certain interests or perspectives [43]. That is, properties are in the eye of the beholder. Thus, properties that may seem useful to a game world designer may be less relevant to an engineer and vice versa.[4]

MATERIALS are usually generated in relation to specific parametric constraints, where parameters can be considered as numeric expressions of properties. Constraints, in turn, can be viewed as a way to describe a class of suitable potential materials. *SpeedTree*, for example, requires its users to express property constraints for the purpose of describing potential trees [14]. Typical constraints might include bounding tree heights, branch angles, and branch widths. Of course, different trees are generated if different property constraints are used, such as restricting branches to occur at approximately equal intervals around each tree trunk. Part of the designer's task in relation to MATERIALS, then, is to articulate a set of property constraints that describe a class of potential materials that are desirable from a design or aesthetic perspective.

In an ideal world, a designer is able to choose between materials. An advantage afforded by MATERIALS is speed of generation. Multiple MATERIALS satisfying constraints can be generated offline, affording choice and abundance to the designer. The designer's task can then shift towards choosing, selecting, and tweaking materials, rather than developing them manually in their entirety. If MATERIALS are generated for just-in-time or online use, however, then manual intervention is not required from human designers, and often is not possible. Of course, many materials cannot be generated procedurally (yet), and MATERIALS are not necessarily superior. But just as there are situations and contexts in which a mass-produced object is preferable to an artisanal one, there are design contexts in which the designer may prefer to make use of MATERIALS in lieu of manually developing materials herself. For example, a designer who has been tasked with creating landscapes for a large open world game may opt to use *World Machine* in order to save time and effort. While the generated landscapes may require some shaping and tweaking to match her design vision, *World Machine* obviates the need to start from a blank slate.

As stated above, PCG systems can often be understood through more than one of the metaphors presented here. For example, some PCG systems can be viewed as both TOOLS

[4]The notion of properties can in fact be used to partly explain why PCG has experienced relatively poor uptake amongst designers: it has, to date, been defined more in terms of engineering properties.

and MATERIALS. Hullett and Mateas's system for generating scenarios for emergency rescue training games [13] can be viewed as a TOOL: it is used to generate training scenarios supporting pedagogical objectives. At the same time, it can be viewed as generating MATERIALS: the scenarios can be understood as MATERIALS meeting specific constraints, and abundance of generation is a core advantage Hullett and Mateas emphasise about their system [13]. What can be confusing about MATERIALS is that they perform computation, a quality usually attributed to tools. Which metaphor is applicable is determined by context of use and perspective. TOOLS are acted with, are used to facilitate particular tasks, and serve the designer. MATERIALS, in contrast, are usually acted on, can be put towards a wider range of tasks, and form a part of the final product.

Throughout this section, we have cast game design as a process of assembling and combining materials, where materials may be art assets, sounds, mechanics, levels, current social context, etc. Casting game design in this way moves the conversation away from strictly procedural or mechanical perspectives. It also emphasises that games can be thought of as a composite of experiences, contingent on socio-cultural, mechanical, and aesthetic factors. While *material* is admittedly a loose class, we see this flexibility as an advantage, especially as game experiences become increasingly amorphous and complex in terms of description.

DESIGNER

Another common way PCG has been positioned is as a DESIGNER. The following passage about Nitsche *et al.*'s game *Charbitat* ascribes it with designerly intentions and responsibilites:

> Charbitat not only arranges entities within each individual tile of the game world, but also provides for larger, overarching structures that span over multiple sections. Rivers, cliffs, walls, and roads are elements that continue seamlessly from one tile to another and can form obstacles and landmark features. [27]

DESIGNERS can be understood as PCG algorithms tasked with solving game design problems and conducting design tasks with little or no intervention from a designer. This is in contrast with TOOLS and MATERIALS, which are dependent on designer manipulation. In fact, a DESIGNER may itself be tasked with the manipulation of TOOLS and MATERIALS. DESIGNERS might perform any design task, such as determining suitable game mechanics, inserting new plot points into a game, adapting the difficulty of a game level, or making decisions about game aesthetics. The designer's role in the context of DESIGNERS may then largely concern design organisation and meta-design activities such as task delineation and selection, synthesis, and quality control.

Fernandez-Vara and Thomson's *Puzzle-Dice* system is a DESIGNER that procedurally generates narrative puzzles [9]. The developers were motivated by wanting to make replayable adventure games that would contain new puzzles in each playing. *Puzzle-Dice* operates on a database of game objects and a puzzle map containing puzzle patterns, both of which are devised by a human designer. It then outputs puzzles populated with game objects based on the specified patterns. *Puzzle-Dice* was used in the game *Symon*, in which the player is in the dreams of a paralyzed patient. Another example is the aforementioned game *Charbitat*, which generates the game world in response to how a player is playing, incorporating pre-fabricated objects that have been made by a human designer [27]. In line with the underlying research focus on space generation, the goal of the game is to explore and generate space. In both these examples, the developers established game objectives and narratives that support and account for the surreality that the DESIGNER might introduce.

The way we have defined DESIGNER assumes a certain autonomy. Depending on the nature of a designer's interaction with a DESIGNER, the DESIGNER might be viewed as an ASSISTANT performing delegated tasks or as a LEAD DESIGNER with more comprehensive responsibilities. An ASSISTANT is tasked with a sub-component of an overall design situated within a larger design problem space that has been mostly pre-determined by a human designer. *Puzzle Dice* is one example of this. A LEAD DESIGNER is tasked with establishing most or all of the design, and the designer instead takes a meta-design role, overseeing and managing the DESIGNER. An example of a LEAD DESIGNER is Togelius and Schmidhuber's experiment in automatic game design [40]. They used evolutionary computation to evolve both rules and agent logics for simple, *Tetris*-like games. As their intention was to generate entire rule sets without the intervention of a human designer, Togelius and Schmidhuber implemented a computational interpretation of Koster's theory of fun [18] as a fitness function for the evolution of a rule set. An alternative use of a LEAD DESIGNER is not to perform design tasks, but to investigate how design itself works. Work in this vein includes Smith and Mateas's notion of *computational caricatures*, implementations of highly simplified, "caricatured" theories of game design [32]. Computational caricatures invite dialogue around automated game design rather than aiming to solve it.

LEAD DESIGNER uses of PCG are uncommon, but an active area of recent research [23]. In the coming years, when LEAD DESIGNERS are more feasible, designers will have to decide how much design responsibility to delegate to them. Whether a DESIGNER should be given an ASSISTANT role or a LEAD DESIGNER role depends in part on pragmatic factors, such as whether the system in question can generate part or all of an envisioned game. But it also depends on trust and how comfortable designers feel about relying on automated systems to undertake game design tasks, especially if the systems rely on complex and somewhat opaque technologies such as neural networks. For example, just because a DESIGNER has previously been able to successfully generate an envisioned game, does that mean that it always will? Trust and certainty are also issues to take into consideration for aesthetic reasons: will the aesthetic experience created by a DESIGNER live up to the standards and tastes of a designer? As an example, although the system developed by Togelius and Schmidhuber was ground-breaking, they described the fittest game generated by their system as "unremarkable" [40].

Depending on the game in question, it can be ethically problematic to leave game design completely up to the DESIGNER. This is especially true of serious games which, because of their "non-entertainment" objectives, are associated with a degree of accountability that entertainment games are often exempt from. Suppose a game for treating PTSD patients errs in determining whether a certain player is sensitive to the sound of children, and operates under the mistaken assumption that the player is not sensitive to the sound, when in fact it is an anxiety trigger for the player. The system generates a scenario featuring children in a playground, causing the player to have an anxiety attack, which in turn makes the player not want to play the game. In such situations, for reasons of ethical accountability, it is important that a human is present serving in the role of an online designer – even if the individual is not a game designer. This individual should be able to modify the game scenario during play sessions, changing the design dynamic to one of *co-design*, in which the DESIGNER and a human collaborate to achieve design tasks.

Whether a DESIGNER is treated as an ASSISTANT, a LEAD DESIGNER, or whether the design dynamic is one of *co-design*, for reasons of accountability, evaluation, transparency, and trust, it is helpful for designers to have personal insight into how a DESIGNER transforms inputs to outputs. Simply seeing quantitative metrics after design is not sufficient. Rather, creators of DESIGNERS should take into consideration how they interface with other game technologies and how they can fit within a development pipeline more generally. In addition, creators should consider supporting more transparent DESIGNERS, allowing designers to make sense of the automated design process themselves. Ideally, creators should develop design tools to be used in conjunction with DESIGNERS. Fernandez-Vara and Thomson's *Puzzle Dice* system, for example, features a puzzle tool editor adapted over time to provide designers direct access to code, and the ability to change puzzles independently of programmers [9].

EXPERT

PCG is often positioned as an EXPERT. For example, Andersen posits the following about PCG-enhanced serious games, implying the need for a virtual expert:

> ...educational games should have an internal model of the concepts that players must learn and track which concepts the player knows and does not know. [2]

An expert, in everyday understanding, has comprehensive and authoritative knowledge with regards to a particular domain. The metaphor of an expert has been appropriated within AI in the form of *expert systems*, computer systems that emulate the decision-making and problem solving of human experts by reasoning about knowledge. Focusing on problem solving qualities rather than decision-making, we propose EXPERTS as uses of PCG related to monitoring, analysing, interpreting, and assessing data resulting from gameplay. Usually, EXPERTS do not rely on intervention by the designer. While game design constitutes an area of expertise, we distinguish EXPERTS and DESIGNERS, retaining the notion of expert for expertise extending beyond standard game design objectives. We extend the EXPERT metaphor by suggesting that there are two dominant forms: the PLAYER EXPERT and the DOMAIN EXPERT.

While a PLAYER EXPERT and DOMAIN EXPERT may both analyse and interpret a game state, for clarity we suggest that the actual adaptation of the game experience falls under the responsibility of a DESIGNER. A DESIGNER encodes expertise in aspects of game design, such as appropriate use of game mechanics and controls [24], while an EXPERT supplies external expertise to be taken into account in the game design. This division mirrors how serious game projects currently employ domain experts within the game development process. While an expert's input is necessary when serious games tackle domains beyond the expertise of the development team, determining how expert input can be incorporated into the game design generally falls under the role of a designer. We also believe that separating expert and designer perspectives brings additional benefits, particularly in providing a clearer understanding of accountability and increasing the transparency of the design process. Thus, in contrast to the PCG metaphors we have proposed so far, EXPERTS do not directly determine the game experience; instead they work in tandem with DESIGNERS.

PLAYER EXPERT

The PLAYER EXPERT encompasses any analysis, interpretation, and adaptation suggestions specifically related to player experience. Its output is used to personalise the game content toward a specific player's experience. Within the PCG literature, this relates to *player experience modelling*, in which player experience is modeled as a function of game content with respect to a player's playing style and cognitive and affective responses to gameplay stimuli [46].

PLAYER EXPERTS are present in any use of PCG that uses player behaviour and experience as input. Togelius *et al.* use a form of PLAYER EXPERT in their work on personalising race tracks to the driving style of players of racing games [39]. Their system analyses player behaviour, then evolves "controller" AI agents that mimic their activity. Once a good enough model of the player exists, the system generates a race track that is determined to be fun by the controller, where fun is operationalised as a function of qualities of the race track in relation to the kinds of challenges it poses for the controller.

PLAYER EXPERTS have often been used to assist DESIGNERS in adapting the contents of games to be sufficiently challenging for players. Kazmi and Palmer describe a system, embodying both a PLAYER EXPERT and a DESIGNER, premised on analysing and interpreting player actions in terms of player skill and style [16]. In the context of a first-person shooter, their system matches a player to a playing style (novice or advanced), and adapts several aspects of the game and its mechanics accordingly. For example, for advanced players levels are modified to be more difficult to navigate safely, while for novices firing directions of weapons are fixed on enemies only, avoiding accidental suicides and wasted ammunition.

Just as experts are expected to be highly knowledgeable about their area of expertise, PLAYER EXPERTS are often trained on data from individuals representative of a target audience of

players. The PLAYER EXPERT is then brought into use once it has demonstrated the ability to analyse or interpret player behaviour in ways that model reality to a sufficient degree, as in Togelius *et al.*'s racing game controllers. Alternatively, PLAYER EXPERTS may be deemed ready once they are capable of establishing patterns in data that are also seen as noteworthy by human designers and players. For example, in a PLAYER EXPERT-enhanced variant of *Super Mario Bros.*, subjective experience reports were taken from hundreds of players after playing particular game levels probing qualities including fun, challenge, and frustration. As gameplay metrics were also tracked for players, offline analyses were conducted to associate player data attributes with subjectively reported player states. The resulting predictors of fun included metrics such as the number of times players kicked turtle shells and the proportion of time spent running [28].

At present, entertainment-oriented PLAYER EXPERTS often rely on "scientificated" notions of fun, typically making use of computational interpretations of the work of Malone [22] and Koster [18]. Aside from the question of whether these works can be proceduralised, they also represent a conservative view of what people find enjoyable about games. Furthermore, attempting to satisfy fun "thresholds" for individual players according to these theories can limit the aesthetic expression of the designer. In operationalising fun, creators of PLAYER EXPERTS should consider moving towards more complex views of what makes games engaging, such as, for example, abusive game design [45].

DOMAIN EXPERT

A DOMAIN EXPERT provides domain-specific analysis and interpretation that is fed into the game design via a DESIGNER. DOMAIN EXPERTS have most frequently been proposed in the context of serious games, which foreground domain knowledge outside game design and player experience. As authenticity of experience and representation becomes more important in game design, DOMAIN EXPERTS will likely also become more common in entertainment games.[5]

An example of a DOMAIN EXPERT is found in the game *Refraction* [2]. *Refraction* is a learning game developed to teach players about fractions. Levelling up in *Refraction* occurs when players demonstrate mastery of particular fraction problem solving skills described in nodes of a concept map. The concept map has three key parameters. *Aggressiveness* concerns how quickly the difficulty advances when a player successfully completes a level. *Thoroughness* determines how many times a player must demonstrate a skill to indicate mastery. *Forgiveness* is the degree to which progression slows when a player fails. From the player's current skill level determined by the concept map, levels are generated according to the skills that players need further practice with. Maintenance of the concept map and assessment of player skills can be considered as the role of the DOMAIN EXPERT, while the level generation according to the concept map can be considered as the role of the DESIGNER.

[5]Domain experts have long been involved in the development of entertainment games. *Police Quest*, for example, was designed by a police officer [44].

A common quality of effective teachers is their adaptation of teaching methods and content delivery according to the needs of their students. In games like *Refraction*, as well as Niehaus and Riedl's *Scenario Adaptor* system, a DOMAIN EXPERT performs analyses to provide the player with challenges of appropriate difficulty that require the use of skills that the player is perceived as needing to practice. This personalising of learning and training toward a player's learning preferences, current knowledge and skills, and learning context has been emphasised by scholars as a promising avenue of PCG application [21, 26].

As with PLAYER EXPERTS, the value of DOMAIN EXPERTS lies in their ability to encode and draw on relevant aspects of domain knowledge, as well as to analyse and interpret game state and player behaviour in its context. Some of these concerns have been examined in the knowledge elicitation literature [8]; involving (human) domain experts in determining whether a DOMAIN EXPERT provides acceptable interpretations would facilitate even more accuracy and accountability. Smith *et al.* describe a level generation system that can be used with *Refraction*, that enables designers to generate, modify, and constrain levels to meet pedagogical and aesthetic objectives [31]. It can be difficult, however, for domain experts to engage in the game development process if game design and programming are not their usual ways of conceptualising domain knowledge.

In a move towards reducing this distance, Belotti *et al.* devised an engine for learning games that decouples game design expertise and domain expertise [4]. In their engine, designing a learning game becomes a process of *task authoring*, where domain experts specify and annotate domain knowledge to appear in the game as tasks, and *task selection*, where game designers determine how tasks are selected, along with their representation. At runtime, a game experience module selects a subset of tasks to present to the player according to their current profile. The game experience module thus performs the role of both a DOMAIN EXPERT establishing the player's current level of knowledge and proficiency and a DESIGNER presenting game tasks that are appropriate.

EXAMPLE: VILLAGE VOICES

Prior to introducing our four PCG design metaphors, we claimed that metaphors could facilitate shared understanding between designers and engineers. Here we show how our metaphors can be used to clarify, from a design perspective, uses of PCG in *Village Voices*, a game we are developing as part of an EU-funded serious games project.

Village Voices is intended to teach children about conflict resolution management [17]. It is a multiplayer open world game that takes place in a virtual village, and emphasises friendship, reputation management, interdependence, and mastery of conflict resolution. As part of daily life in the village, players are required to take various actions related to their characters' livelihoods and responsibilities. As all the characters are interdependent, situations often arise that lead to conflicts, with the players responsible for determining how to manage them. For example, the alchemist may wish to obtain a plant for a potion from the innkeeper, but a longstand-

ing history of prior conflict between them may mean that the innkeeper is reluctant to engage in trade.

Village Voices involves various forms of PCG. An online DOMAIN EXPERT based on the conflict resolution management theories of Bodine and Crawford [5] tracks how well players demonstrate specific conflict resolution skills and identifies those they need more practice with. An online PLAYER EXPERT identifies relationships between players, as our design requires knowledge of whether game characters are on good terms with one another. The PLAYER EXPERT also detects players' affective states via facial expressions and posture. Certain project partners are working towards using this information to obtain insight into players' emotional states and engagement levels. Analyses from the DOMAIN EXPERT and the PLAYER EXPERT serve as input for the game's DESIGNER.

The DESIGNER determines quests and events that are likely to trigger, exacerbate, or diminish conflicts between certain characters, based on which skills the players controlling the characters require further mastery over. Continuing the previous example, the innkeeper may only have a limited quantity of the plant that the alchemist wants. To exacerbate conflict between the alchemist and the innkeeper, the DESIGNER may create a quest that encourages the blacksmith, who is on friendly terms with the innkeeper, to obtain the same plant, creating a situation in which it is even less likely that the innkeeper will trade with the alchemist. Along with specific quests, the DESIGNER may orchestrate events in the game world that create or intensify conflicts for players to respond to. The alchemist may have woken up with a sickness that can only be remedied with a potion containing the aforementioned plant, escalating the situation. The DESIGNER is also tasked with coordinating MATERIALS in the game landscape. MATERIALS include flora and fauna in the village, such as the innkeeper's plants, all of which have changing rates of growth linked to game events.

Village Voices is designed to be played in a classroom under teacher supervision. As teachers possess classroom-specific knowledge that is not possible to model in the DOMAIN EXPERT, such as histories of social incidents between players, *Village Voices* is equipped with a game interface for teachers to support a form of *co-design*. The interface allows modification of game events and actions before and during play in order to customise the game to specific teaching contexts.

Describing *Village Voices* in terms of these metaphors has allowed us to more clearly articulate the different ways in which PCG is being used, and interdependencies between these uses. The metaphors also shed light on design consequences, implications, and improvements regarding our use of PCG, which we address now.

Our system currently relies on a DESIGNER to select suitable game quests and events at runtime. This has created trust concerns in the project team with regards to the abilities of the DESIGNER to select appropriately. The role of our designers has moved toward one of meta-level design organisation, as some of their design agency has now been passed to the DESIGNER, as well as to team members more familiar with the DESIGNER technology. In establishing the concept for *Village Voices*, our designers had to focus particularly on what tasks a DESIGNER could reliably perform. As *Village Voices* is a serious game, these tasks need to interface with established best practice for conflict resolution, as well as insights from the user research undertaken to explore the problem space for the project. Quests and events under the responsibility of the DESIGNER therefore have to be coherent, well-formed, and connected to external reality: our DESIGNER cannot rely on the surrealist context sometimes used in other projects. Our design process thus became an iterative one of planning a new feature for the game concept, determining ways to use a DESIGNER given the updated concept, and then establishing what needed to be changed about the game overall to integrate the DESIGNER. This is akin to establishing an overall game design concept on the basis of what assistant designers on a development team enjoy or are capable of designing.

In terms of learning aspects, the interface provided to teachers enables some design transparency and agency, and allows humans to be involved in potential problem mitigation at runtime. At the same time, conflict resolution experts have no technological means to participate in game design. Similarly, the DOMAIN EXPERT does not provide ways for conflict resolution experts to readily contribute their knowledge or expertise. As a result, we are reliant on the DOMAIN EXPERT providing an accurate assessment of players' skills with regards to conflict resolution. This is a difficult task as to the best of our knowledge the conflict resolution theory we are using has not been tested in a quantitative context, let alone a computational one. Yet it is imperative that our DOMAIN EXPERT can assess conflict resolution skills well. Not only does the DOMAIN EXPERT drive what game content players experience, it informs and shapes their learning experiences, and forms the very basis of the game.

The inclusion of TOOLS in our system would alleviate multiple pressure points. A TOOL enabling designer participation and intervention would result in less burden on the DESIGNER in terms of "getting the design right" and would reduce the meta-design duties of designers, helping to simplify a complex design workflow. A TOOL facilitating domain experts in contributing their expertise directly would reduce the need to move straight from theory that has not yet been operationalised to learning design, and would lower ethical risks currently shouldered by the design team. It is worth noting that while a TOOL may in some ways be the least adventurous use of PCG, it serves a pivotal role in terms of granting human designers and experts agency and empowerment.

CONCLUSION

Procedural content generation has much to offer. For it to be absorbed into the practice of game design, however, it must be contextualised within design-centric as opposed to AI or engineering perspectives. We have sought to reorient PCG research towards these concerns by providing a set of four design-oriented metaphors that describe potential relationships between a designer and PCG, clarifying relationships and dynamics implicit in the existing literature. These metaphors are: TOOL, MATERIAL, DESIGNER, and EXPERT.

By examining PCG through familiar metaphors, we gain the ability to articulate and consider qualities, consequences, affordances, and limitations of existing game-based PCG approaches. The metaphors also serve a generative function, suggesting ways to extend existing uses of PCG to better support the needs of designers. To illustrate how these metaphors aid understanding in terms of design consequences, we discussed our use of PCG in a large-scale project, revealing consequences and limitations of our current design, largely regarding a lack of TOOL support.

Conceptualisation of PCG approaches in terms of the metaphors has raised several key concerns that should be addressed in future research. When PCG is used to create game content, the role of the designer moves toward choosing, selecting, and editing. Depending on how much design responsibility is given to PCG, for example when it is a DESIGNER or DOMAIN EXPERT, the human designer may be required to take on significant organisational duties. Present uses of PCG have also often assumed designers are knowledgeable about the underlying systems. Especially in research contexts, the individuals designing with PCG systems have often also been their creators. This confuses the notion of what a non-creator designer would use, or would be amenable to using, as PCG creators have research agendas and relationships to technology that do not necessarily map to the expectations or practices of typical game designers.

For all uses of PCG, designers and other users such as domain experts must be able to vet the PCG systems they use. This is especially important for applications of PCG where outcomes might be unpredictable and even ethically problematic. Some PCG creators have developed systems and tools that allow their users to trace and affect PCG processes, and we believe this is a direction more PCG creators should follow. This broadens the concept of "evaluation" from something the PCG researcher does up front (in a user study to be published in a paper that claims to have validated the system), towards a more long-term, designer-centric concept. That is, there should be enough information externalized so designers can *themselves* evaluate a system's suitability for, and adapt it to, their own purposes and practices. Not only does this return some agency to designers and improve design accountability, it also prevents PCG applications from becoming divorced from existing game design practice, and facilitates designers and other users in gradually acclimatising to PCG.

PCG has the potential to radically change how we conceptualise games, but also faces significant challenges regarding its integration into design practice. There is much to be done to reduce the gap between the technology and design perspectives. Our metaphors form part of a bridge, aiding designers in understanding, positioning, and appropriating PCG for their own contexts of use, potentially realising powerful, creative, and profound uses of these exciting new technologies.

ACKNOWLEDGEMENTS

This research was partly funded by the EU FP7 ICT project SIREN (project number: 258453).

REFERENCES

1. Agre, P. E. *Computation and Human Experience*. Cambridge University Press, New York, 1997.
2. Andersen, E. Optimizing adaptivity in educational games. In *Proc. Foundations of Digital Games 2012*, ACM Press (2012), 279–281.
3. Baber, C. *Cognition and Tool Use: Forms of Engagement in Human and Animal Use of Tools*. Taylor & Francis, London, 2003.
4. Bellotti, F., Berta, R., De Gloria, A., and Primavera, L. Adaptive experience engine for serious games. *IEEE Transactions on Computational Intelligence and AI in Games 1*, 4 (2009), 264–280.
5. Bodine, R., and Crawford, D. *The Handbook of Conflict Resolution Education: A Guide to Building Quality Programs in Schools*. Jossey-Bass Inc. Publishers, 1998.
6. Boehner, K., David, S., Kaye, J., and Sengers, P. Critical technical practice as a methodology for values in design. In *Proc. CHI 2005 Workshop on Quality, Value(s) and Choice*, ACM Press (2005).
7. Caroll, J. M., Mack, R. L., and Kellogg, W. A. Interface metaphors and user interface design. In *Handbook of Human-Computer Interaction*, M. Helander, Ed. Elsevier Science Publishers, 1988, 67–85.
8. Cooke, N. J. Varieties of knowledge elicitation techniques. *International Journal of Human-Computer Studies 41*, 6 (1994), 801–849.
9. Fernandez-Vara, C., and Thomson, A. Procedural generation of narrative puzzles in adventure games: The puzzle-dice system. In *Proc. Procedural Content Generation in Games 2012*. ACM Press, 2012.
10. Friedman, B., and Kahn, Jr., P. H. Human values, ethics, and design. In *The Human-Computer Interaction Handbook*, J. A. Jacko and A. Sears, Eds. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 2003, 1177–1201.
11. Gearbox Software. Borderlands. Feral Interactive, October 2009.
12. Harrison, S., Tatar, D., and Sengers, P. The three paradigms of HCI. In *Proc. alt.chi 2007* (2007).
13. Hullett, K., and Mateas, M. Scenario generation for emergency rescue training games. In *Proc. Foundations of Digital Games 2009*, ACM Press (2009), 99–106.
14. Interactive Data Visualization, Inc. SpeedTree, 2002 – 2012. Computer software.
15. Interactive Data Visualization, Inc. Speedtree utilities. `http://www.speedtree.com/support/speedtree_utilities.htm`, September 2012.
16. Kazmi, S., and Palmer, I. J. Action recognition for support of adaptive gameplay: A case study of a first person shooter. *International Journal of Computer Games Technology 2010* (2010).

17. Khaled, R., and Ingram, G. Tales from the front lines of a large-scale serious game project. In *Proc. CHI 2012*, ACM Press (2012), 69–78.

18. Koster, R. *A Theory of Fun for Game Design*. Paraglyph Press, 2004.

19. Lakoff, G., and Johnson, M. *Metaphors We Live By*. University of Chicago Press, 1980.

20. Lawson, B., and Loke, S. M. Computers, words and pictures. *Design Studies 18*, 2 (1997), 171–183.

21. Lopes, R., and Bidarra, R. Adaptivity challenges in games and simulations: A survey. *IEEE Transactions on Computational Intelligence and AI in Games 3*, 2 (2011), 85–99.

22. Malone, T. W. What makes things fun to learn? Heuristics for designing instructional computer games. In *Proc. Small Systems 1980*, ACM Press (1980), 162–169.

23. Nelson, M. J. Encoding and generating videogame mechanics. Tutorial at the 2012 IEEE Conference on Computational Intelligence and Games. `http://www.kmjn.org/notes/generating_mechanics_bibliography.html`, 2012.

24. Nelson, M. J., and Mateas, M. Towards automated game design. In *Proc. AI*IA 2007: Artificial Intelligence and Human-Oriented Computing*, Springer (2007), 626–637.

25. Nelson, M. J., and Mateas, M. An interactive game-design assistant. In *Proc. Intelligent User Interfaces 2008*, ACM Press (2008), 90–98.

26. Niehaus, J., and Riedl, M. O. Scenario adaptation: An approach to customizing computer-based training games and simulations. In *Proc. AIED 2009 Workshop on Intelligent Educational Games* (2009), 89–98.

27. Nitsche, M., Ashmore, C., Hankinson, W., Fitzpatrick, R., Kelly, J., and Margenau, K. Designing procedural game spaces: A case study. In *Proc. FuturePlay 2006* (2006).

28. Pedersen, C., Togelius, J., and Yannakakis, G. N. Modeling player experience in Super Mario Bros. In *Proc. Computational Intelligence and Games 2009*, IEEE Press (2009), 132–139.

29. Rizzo, A., Graap, K., McLay, R. N., Perlman, K., Rothbaum, B. O., Reger, G., Parsons, T. D., Difede, J., and Pair, J. Virtual Iraq: Initial case reports from a VR exposure therapy application for combat-related post traumatic stress disorder. In *Proc. Virtual Rehabilitation 2007* (2007), 124–130.

30. Schmitt, S. World Machine. `http://www.world-machine.com/`, 2012. Computer software.

31. Smith, A. M., Andersen, E., Mateas, M., and Popović, Z. A case study of expressively constrainable level design automation tools for a puzzle game. In *Proc. Foundations of Digital Games 2012*, ACM Press (2012), 156–163.

32. Smith, A. M., and Mateas, M. Computational caricatures: Probing the game design process with AI. In *Proc. AIIDE Workshop on Artificial Intelligence in the Game Design Process*, AAAI Press (2011), 19–24.

33. Smith, G., Gan, E., Othenin-Girard, A., and Whitehead, J. PCG-based game design: Enabling new play experiences through procedural content generation. In *Proc. Procedural Content Generation in Games 2011*, ACM Press (2011).

34. Smith, G., Othenin-Girard, A., Whitehead, J., and Wardrip-Fruin, N. PCG-based game design: Creating Endless Web. In *Proc. Foundations of Digital Games 2012*, ACM Press (2012), 188–195.

35. Smith, G., Whitehead, J., and Mateas, M. Tanagra: A mixed-initiative level design tool. In *Proc. Foundations of Digital Games 2010*, ACM Press (2010), 209–216.

36. St. Amant, R., and Horton, T. E. Characterizing tool use in an interactive drawing environment. In *Proc. Smart Graphics 2002*, ACM Press (2002), 86–93.

37. Sullivan, A., Mateas, M., and Wardrip-Fruin, N. Rules of engagement: Moving beyond combat-based quests. In *Proc. Intelligent Narrative Technologies III*, ACM Press (2010).

38. Togelius, J., Kastbjerg, E., Schedl, D., and Yannakakis, G. N. What is procedural content generation? Mario on the borderline. In *Proc. Procedural Content Generation in Games 2011*, ACM Press (2011).

39. Togelius, J., Nardi, R. D., and Lucas, S. M. Towards automatic personalised content creation for racing games. In *Proc. Computational Intelligence and Games 2007*, IEEE Press (2007), 252–259.

40. Togelius, J., and Schmidhuber, J. An experiment in automatic game design. In *Proc. Computational Intelligence and Games 2008*, IEEE Press (2008), 111–118.

41. Toy, M., Wichman, G., and Arnold, K. Rogue, 1980. Computer software.

42. Treanor, M., Blackford, B., Mateas, M., and Bogost, I. Game-O-Matic: Generating videogames that represent ideas. In *Proc. Procedural Content Generation in Games 2012*, ACM Press (2012).

43. Vallgårda, A., and Redström, J. Computational composites. In *Proc. CHI 2007*, ACM Press (2007), 513–522.

44. Walls, J. Police Quest. Sierra On-Line, 1987. Computer software.

45. Wilson, D., and Sicart, M. Now it's personal: On abusive game design. In *Proc. FuturePlay 2010*, ACM Press (2010), 40–47.

46. Yannakakis, G. N., and Togelius, J. Experience-driven procedural content generation. *IEEE Transactions on Affective Computing 2*, 3 (2011), 147–161.