Hybrid Procedural Content Generation: A Proposal

Michael Cook and Simon Colton[1]

Abstract. Procedural content generation in games tends to target content that is abstract, dry and devoid of connection with the game's meaning. This paper proposes merging user-driven content generation approaches with procedural content generation to create a new paradigm which we call *Hybrid Procedural Content Generation*. By replacing aspects of existing procedural generation techniques with humans, we can give rise to new kinds of game experiences.

1 Introduction

Procedural content generation and user-generated content (PCG and UGC respectively) are two concepts which are familiar to anyone who has played or made games in the past decade. The idea that content for a game can be created after it has shipped enables many new kinds of game experience, as well as engaging players in new kinds of activities, including creative involvement in the game. They also provide interesting research platforms to ask new questions and build intelligent systems to help shape these new ideas about games.

In this paper we introduce the concept of *Hybrid* Procedural Content Generation (HPCG), a fusion of user-generated and procedurally-generated content that similarly offers new kinds of game design and also new opportunities for artificial intelligence in games. By incorporating players into procedural content generation systems we can produce hybrid systems that are much stronger than standard procedural or user-driven generative approaches.

We illustrate the concept of HPCG by giving three examples of prototype games which incorporate some kind of HPCG system into their game design. *Murder* is an assassination game set in a Cluedo-esque mansion at a dinner party, in which the player must perform several narrative actions and then kill another character at the party. *Mystery* is a Poirot-style detective game in which the player must solve a murder using deduction and exploration. *The Book Of A Thousand Tales* is a roleplaying game in which the player leads a band of heroes through a branching narrative.

The remainder of the paper is organised as follows: in *Background* we discuss both PCG and UGC and their relative weaknesses. In *Hybrid PCG* we briefly introduce the concept of HPCG, its motivating factors and how we see it being used within games. We then describe two simple game designs that comprise a HPCG system. Finally in *Opportunities for Computational Intelligence* we talk about the longer-term impact of such approaches and the potential for new research directions HPCG could give rise to. We then sum up our proposal in *Conclusions*

2 Background

According to [6], most PCG systems can be categorised as either *constructive* or *generate-and-test* systems. In the former, content is gradually built up out of successive passes at generation, and each layer of generation is "*guaranteed to never produce broken content*" [6]. Spelunky [2] is a good example of this style of generation, where dungeon levels are built out of several different layers of content which are hand-crafted to some extent to guard against failure [7]. Generate-and-test approaches employ a generative step that produces content, and then an evaluative step which assess what was generated and either triggers further generation/alteration (such as an evolutionary system which will run many times to evolve a result, as in [2]) or simply reject the generated content and begin again from scratch. *Dwarf Fortress* [3] employs a generate-and-test approach during its world generation.

PCG has been applied very effectively to many kinds of content generation, particularly level design [3] and general game content such as item generation in roleplaying games. However, many types of content are hard to generate using either of the above approaches. In particular, content which requires an understanding of context of the real world is hard to generate, such as game narratives or replicating human-like qualities in NPC actions such as deception or fallibility. These dynamic kinds of content rely on an understanding of the real-world, from cultural knowledge (like understanding symbolism when constructing a narrative) to common-sense reasoning (when deciding how a character should react to a particular situation, for instance). As a result, most content generation focuses on abstract data that is detached from the game's setting and theme (the levels in Spelunky are simply arrays of numbers, for instance – the system does not need to understand what a cave looks like or what an explorer does).

User-generated content (UGC) is also a common feature in many modern games. Allowing the player to create content for a game both increases the amount of content available at no extra cost to the developer, and gives players a sense of engagement and investment in the game world by allowing them to contribute to it. *Spore* [4] is a prominent example of user-generated content – players designed animal species for inclusion in their games using an assortment of body parts and customisations. These animal species propagated not only throughout the player's world but also to their friends' worlds via cloud sharing online.

UGC is one of the biggest recent trends in the mainstream industry thanks to the enormous success of *Minecraft* [5], which merged user-generated content with the core mechanics of the game. In Minecraft, generating content is how one plays the game: building structures, artworks and shaping the world as the player sees fit. UGC has drawbacks, however. In the case of generators like Spore's, which present themselves as tasks outside of gameplay, the user is consciously

[1] Computational Creativity Group, Goldsmiths, University of London

[2] Mossmouth Games, 2009
[3] 2006, Bay Twelve Games
[4] Maxis, 2008
[5] Mojang, 2011

aware that they are generating content. As a result they are thinking about how the content will be perceived by others, which has an impact on how and what they create. This can be seen somewhat in the comedic nature of many of Spore's creatures – players know they are creating things which will amuse or confuse other people. While this may be seen as a positive for some tasks (in Spore's case the objective is specific content generation) because the player is consciously considering the design of their content, for other tasks it may be less good – particularly those that take place in a fictional context. For example, in Minecraft it is possible to construct floating houses, which may break the suspension of disbelief for other players. It is preferable here that all players construct buildings in a similar way, so that they can maintain the narrative fiction for everyone equally.

The second drawback is that players tend not to be designers, and UGC systems rarely have any kind of feedback mechanism or assistive aspect to them. Content is either used wholesale or not used at all, and frequently even this decision is made by players rather than an intelligent software system. Creatures in Spore are uploaded and shared online, structures in a Minecraft world exist for all players in that world and can't be edited or changed by the game. UGC is all-or-nothing and thus lives or dies on the skill and appreciation of the players using these systems. In some cases this can be worked around – ratings systems in games such as LittleBigPlanet [6] simply filter the best creations and downplay the rest. In this case, however, UGC simply becomes a means by which to discover talented people and get them to produce content, rather than allowing everyone to contribute equally.

3 Hybrid PCG

We propose that PCG and UGC approaches can be combined in a single approach that solves some of the problems mentioned in the previous section while opening up new challenges and research questions for computational intelligence research to tackle. We call this combined approach *Hybrid PCG* because it synthesises software-driven content generation with player activity. The underlying premise is to replace generative systems or parts of systems with playable games, resulting in new ways of generating, evaluating and filtering content, not just for single games by potentially for many different games at once.

To illustrate this approach, we will describe in this section two in-development game prototypes, *Murder* and *Mystery*, which utilise a HPCG approach to generate a large corpus of content and filter it. These games not only supply content to one another: by generating content that is transferred between games, they also produce a corpus that can be used by other games or intelligent systems. After describing the games we will discuss the new affordances such a setup offers and then lead into a discussion of the opportunities for computational intelligence they represent.

3.1 Illustrative Example - Murder/Mystery

In *Murder* the player takes on the role of a character attending a dinner party at a mansion, as either a guest, a family member, or an employee of the host. Like most of the people present they have a motive to kill the host, and must do so at some point during the evening. In addition, they must also complete one or more objectives relating to their motive (such as confronting the host in an argument, or breaking into a room and stealing something). The game operates in a 'sandbox' style, where the player can explore the house freely and approach their objectives in many different ways. However, the game simulates player action carefully and records things like fingerprints left on surfaces, sightings by other people in the house, and so on.

At the end of the game, once their tasks are completed, the player can choose to 'discover' the body themselves or wait for it to be discovered by someone else. They are then asked to provide an account of their whereabouts for the evening by being shown their actual movements and then editing them to change their version of events – for example, by claiming they were never in a particular room at a certain time, and so on. The game then assesses how quietly and quickly they completed the game, as well as how well their alibi compares to the evidence they left behind, and gives them a rating.

In *Mystery* the player takes on the role of a detective tasked with solving a murder at a dinner party. They play a point-and-click adventure in which they can examine the alibis and backgrounds of the characters present, ask for accounts of events, and walk around the house looking for clues or analysing parts of the crime scene. The case files are built from case descriptions produced by *Murder*, potentially converted using an automated system that can filter the case to make it harder or easier (by making certain evidence more or less conclusive or adjusting the memories of other characters, for example) or simply presented to players unaltered – we discuss this further in section 4.

There is a time and resource limit on solving a case - if the player takes too long or uses up all of their investigative resources (such as sending objects for fingerprinting) the case remains unsolved. Whatever the result, the case file data gets sent back to a central server which both affects the value of a case (repeatedly unsolved cases rise in value to detectives) and the reputation of the player who created the case file in *Murder*.

3.2 HPCG in Murder/Mystery

Both *Murder* and *Mystery* are standalone games that are effectively separate from one another. If the data format for case files is open, anyone could design a game which retrieved case files produced by *Murder* players and use them in their game. Similarly, several games might produce case files with the right format that could be used by *Mystery* as game content for the player to investigate and solve. The games are not intrinsically linked except through the exchange of information about the case files and whether or not they are solvable by players.

In the language of PCG, players of *Murder* are acting as a generator of case files, in the first step of a *generate-and-test* system. There are two important consequences of this. Firstly, unlike UGC approaches, the players are *engaged in a game* while generating content, pursuing objectives in whatever way they see fit. We argue that this leads to more natural behaviour by players and therefore a more human-like kind of content generated than if players had been asked to manually design case files as authors. Secondly, the content being generated is complex - it involves creative problem-solving and asks the player to respond to social situations (such as confronting someone about a personal relationship, or making small-talk at a dinner party). Such content is difficult to generate automatically without a lot of involvement from a designer, and even with such involvement the content is likely to be lacking in variety over a long period of play. By using players to generate it, we make this difficult generative task easier.

To continue the PCG metaphor, players of *Mystery* act as evaluators of the content generated by *Murder* players. Let us assume that

[6] Media Molecule, 2008

Mystery either does not edit the case files at all, or at most edits them in order to ensure that they can be solved by some process of deduction (by ensuring that at least one piece of incriminating evidence exists, for instance). Players solving, or attempting to solve, cases are providing data about how easy a case is to solve. The routes players took, the order in which they examined evidence or questioned people, and their ultimate success at solving the murder can all be recorded as additional metadata attached to the original case file. In the same way that people can be used to generate content that requires complex understanding of the real world, people can also be used to provide evaluation metrics that would be difficult to encode into a system by hand (and too subjective to source from a single designer).

3.3 Desirable Properties of HPCG Scenarios

While this remains a preliminary proposal for HPCG, and the idea still needs much exploration, we posit that certain game designs or scenarios are better suited for the application of HPCG. We discuss them briefly here, and hope to clarify this in future work after more experimentation and prototype development.

3.3.1 Asynchronous Activity

The most important property for employing HPCG is that the games involved deal with asynchronous activity. Murder/Mystery work well because the two game phases are chronologically non-overlapping: one player commits a crime, then after they are finished the second player can arrive and solve it. This means that no player is left waiting for action to be completed in real-time, which could affect the experience of either player and slow down gameplay, and it also means that any PCG systems have complete information from the other game or games when they begin generating content.

3.3.2 Well-Defined And Decoupled Interfaces

Keeping the interfaces between games as simple as possible is a good feature if the designer intends for other systems to feed data into the HPCG besides their own. For Murder/Mystery we noted that in theory it is possible for other games to generate crimes for Mystery to solve, or to design other games which use Murder case files as input content. In order to enable this, it's important that the interfaces between the games are very well-defined and public so that other developers can take advantage of them. Making sure the games can export data as well (such as putting Murder's case files in external text documents) also makes this easier.

3.3.3 Guided Player Activity

Depending on the kind of content being generated or the roles the players are taking on in the larger HPCG system, it may be desirable for the gameplay to be very directed or guided. The reason for this is that the HPCG system is making assumptions that the data they collect represents a certain kind of behaviour from the player - for example, committing a crime, not wanting to leave evidence behind, acting in order to blend in. It's important to be able to encourage and motivate the player to work towards certain objectives so that these assumptions carry through into the data they generate, and can then be relied upon to generate good quality content in other areas of the HPCG system. If a player begins acting differently, or isn't sufficiently motivated to play properly, the HPCG system will still proceed with the data and this can generate undesirable outcomes in other games.

4 Opportunities for Computational Intelligence

On the surface, HPCG appears to replace software-driven PCG systems with players that perform the same tasks, therefore resulting in systems that involve *less* computational intelligence, rather than more. However, HPCG systems open up new research questions that demand answers, and also create opportunities to build even more complex generative software. In this section we discuss several possibilities in brief.

4.1 Learning From Human Generators

One possible outcome from HPCG systems is that they eventually transition back into being PCG systems which use a player's in-game activity as a source of training data. In [4] Orkin and Roy describe *The Restaurant Game* (TRG), an experiment in which participants played through an interactive scenario in pairs and their behaviour was then recorded and later analysed using machine learning to build behaviour models of characters in those situations. TRG suffers from some of the same problems that we mentioned in the context of UGC earlier in the sense that players are aware they are generating content as they play. Nevertheless, the authors' argument is that automatic content generation (in this case speech and behaviour patterns) can be mined from large-scale data corpora [5].

By employing HPCG to tackle complex generative tasks, like the generation of creative behaviour in *Murder*, such systems produce special cases of the kinds of corpora Orkin and Roy present with The Restaurant Game. They are special cases in the sense that they are obtained through observing players at a time when their primary concern is *completing* a game rather than performing for another observer (whether that observer is a human or a data-mining program). The player is not participating in an experiment, nor is their ultimate goal to provide good data. Instead, they are focused on achieving objectives and are immersed in a ludic task. As a result, we argue that their behaviour is more natural and as a result more valuable, resulting in useful corpora of data that can be mined, as with TRG, to obtain behaviour. In the case of games such as *Murder*, the available information is particularly valuable because the player is providing information that an ordinary PCG system would not have access to - such as solving problems in creative or innovative ways, as well as failing at tasks in a natural, humanlike way.

4.2 The Computer As Curator

The game *Murder* can be seen as a generator of content for the game *Mystery*, but raw generated case files from the game may not be interesting, fun to solve or, indeed, solvable at all. Building *Murder* as a HPCG system provides us with a wealth of generated murder cases for players to solve, but it doesn't guarantee their quality or difficulty level. If a player plays a perfect game, it will be fairly unsatisfying for players of *Mystery* to repeatedly fail to solve. Similarly, the player may make an obvious mistake that renders a case trivial. This poses an interesting problem: how can software curate, tweak and improve raw HPCG output to ensure consistently entertaining content for another player?

There are many factors to tweak in a case file produced by *Murder* – both the actions of the players and the other characters, the evidence left behind, the ordering of events. Altering this information requires

an understanding of how people's behaviour is interpreted by others, to assess whether a change will make a case easier or harder to solve for a player detective. HPCG systems leverage human players to solve creative, complex problems that are hard to solve using generative software alone. It follows, therefore, that curating and improving the results of a HPCG problem requires an understanding of how these players reason about problems and act in certain situations. The task of curating complex creative content sourced from humans may have parallels with the problem of curating and evaluating in Computational Creativity [1].

Recall that in section 2 we discussed the problems with existing UGC and PCG paradigms. One problem with UGC approaches is that players are not designers, and expecting them to be able to produce quality game content, either knowingly or not, is unreasonable and often results in a large volume of low-quality content that no-one wants to use. HPCG offers an opportunity to leverage the output of users and improve it using computational intelligence, obtaining content that has its foundations in the creativity of real players, but has been curated and refined by software to be of higher quality.

5 Acknowledgements

The authors would like to thank the reviewers who provided helpful feedback which improved this paper. This work was sponsored in part by EPSRC grant EP/L00206X.

6 Conclusions

In this paper we briefly outlined a proposal for *Hybrid Procedural Content Generation* or HPCG, a synthesis of user-generated content and procedural content generation where subsystems in a content generation pipeline are replaced with players playing games achieving similar tasks. We illustrated the idea with two connected games – *Murder* and *Mystery* – in which players of the former acted as a generator of content which was then filtered and evaluated by players of the latter. We discussed what new avenues of research such an approach might offer and how it solves some of the problems that procedural content generation and user-generated content can have.

This paper is an early proposal for such games and systems to be designed, but we hope that it will spark discussion and potentially lead to interesting new kinds of games and intelligent software. We believe that working with game developers may be of essence here, to leverage good game design alongside new kinds of computational intelligence. Collaboration is difficult, but we believe this is a promising avenue to explore.

REFERENCES

[1] Simon Colton, Michael Cook, Rose Hepworth, and Alison Pease, 'On acid drops and teardrops: Observer issues in computational creativity', in *Proceedings of the 7th AISB Symposium on Computing and Philosophy*, (2014).

[2] Erin J Hastings, Ratan K Guha, and Kenneth O Stanley, 'Evolving content in the galactic arms race video game', in *IEEE Symposium on Computational Intelligence and Games*, (2009).

[3] Britton Horn, Steve Dahlskog, Noor Shaker, Gillian Smith, and Julian Togelius, 'A comparative evaluation of procedural level generators in the mario ai framework', (2014).

[4] Jeff Orkin and Deb Roy, 'The restaurant game: Learning social behavior and language from thousands of players online', *Journal of Game Development*, (2007).

[5] Jeff Orkin and Deb K. Roy, 'Understanding speech in interactive narratives with crowdsourced data.', in *AIIDE*. The AAAI Press, (2012).

[6] Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne, 'Search-based procedural content generation: A taxonomy and survey', *IEEE Transactions on Computational Intelligence and AI in Games*, (2011).

[7] Derek Yu. The full spelunky on spelunky, 2012.

Procedural content generation â€“ PCG can be described as a se-rie of techniques that enable automatic content generation, random textures, 3d models and even audio can be created using PCG. The Programmable Graphics Pipeline â€“ PGP is the way to manipulate how a 3d scene is going to be rendered by coding each step of the graphics pipeline through shaders, shaders in turn are pieces of code than run directly into the GPU [16].Â We presented our proposal for a multi-resolution technique terrain for games. We aim to make it possible combining two strategies with CPU large scale terrain rendering, and with GPU precise low scale terrain rendering. We are going to use PCG to both gener-ate those terrains and to add details on the highest possible LOD. A textbook about procedural content generation in games, fully readable online. Texturing & modeling: a procedural approach. A classic reference on the subject by pioneers and creators of the featured methods and algorithms (Ken Perlin, Steven Worley, F. Kenton Musgrave, David S. Ebert and Darwyn Peachey). Generative Art: A Practical Guide Using Processing. The book presents multiple examples of generative arts and explains the techniques used in some of them in the form of short tutorials (in Processing). Scratchapixel 2.0. Free online ebook on computer graphics featuring articles on valu