Third Year Project: Literature Review

Matthew Taylor

Registration: 100151729

1 Introduction

1.1 Aim of the project

The project aims to implement a first person shooter game which uses procedural generation to provide a novel and interesting gaming experience. There any a lot of ways procedural content generation (PCG) can be applied to games, so this literature review will explore the broad themes of the field to find specific areas to focus on.

1.2 Areas of knowledge required

There are many PCG techniques, which have been developed and used in video games since the 1970s. I will need to gain a strong understanding of this field in order to find interesting areas to focus on.

I will also need to learn the Unity platform, in order to implement my ideas.

1.3 Report structure

The report will provide a summary of the PCG field, identifying key themes and the current state of the field, including interesting "unsolved problems" or under-explored areas.

I will then focus on a few promising areas to explore the literature in detail, with a view to informing my project's aim and objectives.

2 Key themes

2.1 Types of Procedural Content Generation

There are two main types of procedural content generation, which use similar methods but achieve different goals. They are **endless content generation** and **efficient content storage**.

2.1.1 Efficient content storage

Prevalent in the early days of PCG, efficient content storage exploits procedural content generation to create large amounts of data without needing to store it - it can simply be re-generated using a deterministic algorithm every time it is required. An example of this is the 1984 game *Elite*, as documented in Spufford (2003). Here, the game developers discuss using a pseudorandom number generator with pre-defined seeds to generate large amounts of content, without needing to store any of the content itself.

A similar technique is adopted for image generation, as per Perlin (1985). Perlin describes using well-defined stochasic functions that can be parametised to produce realistic textures procedurally. This means that textures can be generated when required or when a program is initialised, rather than needing to store them.

A lot has changed in computers since the 80s - memory and storage constraints are now much less of an issue, so using PCG to efficiently store large amounts of content has been used less.

2.1.2 Endless content generation

As storage constraints have been eclipsed by the rapid development of technology, the generation of content has become more of a concern. Increasing sophistication of games and the expectations of players mean content creation is more time-consuming and costly. This has resulted in an increase in the use of PCG to generate content.

This has become a wide field, resulting in the creation of a taxonomy of PCG in Hendrikx et al. (2013). It describes six layers of PCG in games:

1. Game bits (eg. textures, buildings)
2. Game space (eg. maps, lakes)
3. Game systems (eg. roads)
4. Game scenarios (eg. puzzles, stories)
5. Game design (eg. rules)
6. Derived content (eg. leaderboards)

This paper is relatively recent (2013) which makes the summary it provides still accurate to this day. It is cited (36 times on acm.org) by others in the industry. I have done some comparison against http://pcg.wikidot.com/ which maintains a list of games using PCG, which confirms there have been no large advances or practical examples of PCG in games since this was published. This gives me confidence that the taxonomy and techniques discussed are still accurate and relevant today.

In chapter 6 of Hendrikx et al. (2013), the paper discusses "recommendations for future research". One of the recommendations details the generation of realistic indoor game spaces. I will explore this field and the literature surrounding it in more detail.

3 Procedural generation of indoor spaces

3.1 Room generation

3.1.1 Agent based approach to decoration

Germer and Schwarz (2009) describe an agent based approach to decoration, which generates the furniture placement in a room. Agents are assigned to particular objects and, when entering a search state, seek positions that meet particular constraints and goals. It is designed to be deterministic (using PRNGs) and to generate rooms at speed (to avoid storing lots of geometry), when required. These are interesting properties that I would like to explore further. The authors acknowledge that high-level control and sophistication of the generated layouts are lacking, so these are areas I could look to build on.

This paper is now relatively old (2009) and not particularly well cited (22 citations on Wiley Online Library). It uses relevant techniques (agent based search) but uses out-dated technology (VRML). The ideas in the paper are interesting and described in considerable detail, so it is still a useful paper to consider.

3.1.2 Computerised clutter

Taylor and Parberry (2010) discuss "computerised clutter" and the challenges involved in making a procedurally generated room look realistic. They list five properties that are desirable in a room generator: novelty, structure, interest, speed and controllability. The paper describes in detail how to procedurally generate "clutter" in a room which does not look computer generated, by using the concept of "anchor points". These points are adjusted with gaussian noise to provide natural looking variation around points and patterns that can be designer-adjusted, making them look human-constructed when in reality they are generated by computers.

This paper is very specific although not well cited (1 citation on researchgate.net). It is also not very recent, published in 2010. These facts mean I am not going to read too much into the paper, especially as they primarily present only one technique of generating "computerised clutter". However, their subjective summary of what makes procedural generation of clutter in a room look convincing is useful and compelling enough that I have included it. I will incorporate some of these techniques into the decoration of my room generator.

3.1.3 Combining human and procedural content generation

Discussed in the papers above is the idea of content "looking" procedurally generated. Indeed, in the most recent prominent example of a game making extensive use of PCG - No Man's Sky (2016) - is criticised in many reviews, with Welsh (2016) as an example,

for being vast but samey. To quote the review: "Some disappointment is inevitable once you've visited a handful of planets and started to see the limits of the algorithm. The parameters of weirdness in No Man's Sky have been quite strictly defined, and it gets repetitive."

Rogue-like games tend to suffer less from this phenomenon. A recent example that has been explored in detail is the game Spelunky. As detailed in Brown (2016), Spelunky uses a simple path-finding algorithm, a tile-based approach to generating levels and - crucially - combines human crafted tiles with scope for procedural generation within those tiles.

This generates detailed, interesting levels that feel unique, interesting and always have a route from beginning to end. The combination of taking human-designed tiles or areas and integrating procedural elements within them, as well as arranging the tiles procedurally is not a common technique, based on my research. In addition, I cannot find any references or examples of this being employed in a 3D environment.

4 Level generation

Hendrikx et al. (2013) in Chapter 3 discuss common methods of PCG. In relation to how indoor spaces are generated, they discuss how common methods are Pseudo-random Number Generators (PRNG) and Generative Grammars (GG).

These methods are well suited to producing feasible indoor spaces required in games. Although Hendrikx et al. (2013) detail other techniques, like genetic algorithms and fractal spatial generation, these are less common in games. I suspect this is because the output of layouts they produce, despite being procedurally generated, will either follow a discernable pattern (in the case of fractal algorithms) or will be "too random" (in the case of genetic algorithms) to appeal to human players. My aim is to create a procedural generator that feels realistic in order to be compelling and interesting to the player, so I feel these approaches may not be suitable.

4.1 Generative grammars

Dormans (2010) discusses using generative grammars (GGs) to create complex, coherent, branching mission and level structures. As the article details, these generate levels more appropriate for action adventure games (as compared to levels generated by roguelikes for roleplaying games) where a pattern and structure to the level is important. The article also details the difference between mission and space.

The paper is reasonably well cited (22 citations on acm.org) and was published in 2010. The techniques it describes are certainly not dated and could concievably be used to generate content in games today.

The ideas presented are intriguing, as they specifically reference the type of game I aim to create - a 3D action-adventure style game.

5 Evaluation and analysis

I use several kinds of sources in my literature review, from academic journals to books and articles about techniques used. Here, I will evaluate and analyse my sources.

spuff This is a book that directly interviews and quotes the designers of the game Elite. Despite the age of the source (2003) it describes techniques used in the 80s - so it's age is acceptable. The fact it directly quoted developers about the algorithms they used makes this a reliable source.

perl This Perlin paper is quite old, so there are certain techniques that have superceded some of the world. Nevertheless, this was a very influential paper (cited 427 times on acm.org) and the work presented is still very relevant today. Perlin Noise is still a commonly used algorithm in procedural generation, especially in things like realistic-looking heightmaps in modern tools.

References

Brown, M. (2016). How (and why) spelunky makes its own levels | game maker's toolkit.

Dormans, J. (2010). Adventures in level design: Generating missions and spaces for action adventure games. pages 1:1–1:8.

Germer, T. and Schwarz, M. (2009). Procedural arrangement of furniture for real-time walkthroughs. *Computer Graphics Forum*, 28(8):2068–2078.

Hendrikx, M., Meijer, S., Van Der Velden, J., and Iosup, A. (2013). Procedural content generation for games: A survey. *ACM Trans. Multimedia Comput. Commun. Appl.*, 9(1):1:1–1:22.

Perlin, K. (1985). An image synthesizer. *SIGGRAPH Comput. Graph.*, 19(3):287–296.

Spufford, F. (2003). *Backroom Boys: The Secret Return of the British Boffin*. Faber and Faber.

Taylor, J. and Parberry, I. (2010). Computerized clutter: How to make a virtual room look lived-in.

Welsh, O. (2016). No man's sky review (eurogamer.net).

Literature review

Introduction: brief description of project, areas of knowledge required, roadmap	First	2.1	2.2	3	Fail
Discovery of suitable quantity and quality of material	First	2.1	2.2	3	Fail
Description of key issues and themes relevant to the project	First	2.1	2.2	3	Fail
Evaluation, analysis and critical review	First	2.1	2.2	3	Fail

Quality of writing

Clarity, structure and correctness of writing	First	2.1	2.2	3	Fail
Presentation conforms to style (criteria similar to conference paper reviews)	First	2.1	2.2	3	Fail
References correctly presented, complete adequate (but no excessive) citations	First	2.1	2.2	3	Fail

Revised Workplan (if applicable)

Measurable objectives : appropriate, realistic, timely	First	2.1	2.2	3	Fail

Comments

Supervisor: Dr Rudy Lapeer

Markers should circle the appropriate level of performance in each section. Report and evaluation sheet should be collected by the student from the supervisor.