Efficiency, Realism, and Representation in Generated Content

A Case Study using Family Tree Generation

Peter Mawhorter
Pomona College
Computer Science Department
Claremont, CA 91711, USA
Massachusetts Institute of Technology
Computer Science and Artificial Intelligence Laboratory
Cambridge, MA 02139, USA
pmawhorter@gmail.com

ABSTRACT

Work on procedural content generation often centers game mechanics and visual/audio aesthetics, whereas the generation of social structures has not received the same attention, despite potentially enabling new forms of gameplay. When the generation of social and/or cultural content is attempted, tensions naturally arise between algorithmic efficiency, realism, and representation. A specific algorithm for generating family trees is used as a case study for these issues, with the hope that this can stimulate broader discussion in the community.

CCS CONCEPTS

•Applied computing → Media arts; •Theory of computation → Generating random combinatorial structures; *Pseudorandomness and derandomization;* Random network models;

KEYWORDS

procedural content generation, representation, relationships

ACM Reference format:
Peter Mawhorter. 2017. Efficiency, Realism, and Representation in Generated Content. In *Proceedings of FDG'17, Hyannis, MA, USA, August 14-17, 2017,* 4 pages.
DOI: 10.1145/3102071.3110570

1 INTRODUCTION

Generative methods have been applied to the creation of digital game worlds since practically the advent of digital games (see e.g., *Rogue* [22]). The techniques applied have focused in large part on physical play spaces and combat-centric enemy distributions and statistics, however (in games such as *Minecraft* and *Left4Dead* [14, 20]). In contrast to these techniques, a few games have applied generative methods to things like narrative development, interpersonal relationships, or even entire cultures and mythologies (e.g., *Façade*, *The Sims*, *Dwarf Fortress*, and *Ultima Ratio Regum* [1, 6, 7, 10–12]). Despite the potential these games have shown for introducing new modes of gameplay and engaging new audiences, academic research on procedural content generation (PCG) for in-game cultures and societies is quite rare relative to techniques for generating physical environments.

One reason for this may be that when algorithms move from physical spaces to social or cultural spaces, they take on additional cultural implications that may be divisive or exclusionary, ultimately becoming flashpoints for pushback. Generating social/cultural spaces puts in conflict the ideals of efficiency (of algorithms and/or development), realism (or believability), and representation (broadly: creating work that has positive social impact, especially with respect to marginalized identities). The game *RimWorld* [21] serves as a recent example of such conflicts. The relationship code in that game (a colonization/management game which uses PCG for its starting scenarios) came under fire after it was revealed to proscribe gender roles in a way that created problematic in-game dynamics (unfortunately, as one might expect, the out-of-game dynamics that resulted were also fraught) [9]. This kind of incident shows how complicated the ever-present spectre of "bad content" can get when social/cultural content is produced. Ignoring for a moment the author's intentions, *RimWorld* illustrates what could be called a tension between efficiency (in this case of developer time) and representation, where stereotype-propagating in-game dynamics are the result of a quickly-put-together system. Tensions can also arise between realism and representation: for example, a realistic generated medieval society would include many harmful dynamics (e.g., child slavery) that would require careful framing to avoid sending the wrong messages. This careful framing is complicated by the fact that some dynamics arise indirectly from procedural rules, requiring intensive analysis beforehand to recognize and then frame appropriately. Unfortunately, the most common solution is to simply filter out such fraught dynamics, but this can not only lead to efficiency problems, it also impacts realism in a way that can amount to erasure. Many users will of course not notice this erasure, but that's precisely why it's harmful.

The procedural generation of social and cultural relationships thus presents unique challenges in terms of representation, and to elaborate on these further, a simple case study is useful. In the next section, I discuss a system that generates nuclear family relationships (mothers, fathers, and their children) in a stateless/implicit

FDG'17, Hyannis, MA, USA

978-1-4503-5319-9/17/08…$15.00
DOI: 10.1145/3102071.3110570

manner, making it randomly accessible. Being randomly accessible is a useful property for several reasons, including simulation scaling, but presents a unique set of challenges that complicate realism and representation (for more discussion of explicit vs. implicit generation and some of the benefits of both approaches, see [23], which discusses this problem in the domain of texture synthesis).

2 RANDOM-ACCESS FAMILY TREES

In the game *Noctis* [4], players explore a procedurally generated galaxy about twice the size of our own, including billions of stars and orbiting planets. The design takes advantage of a simple fact central to many PCG approaches: although the information content of this galaxy is ridiculous, in a given play session, a single player will only explore a very tiny fraction of it, and all of that information can be coherently generated using randomly-accessible (sometimes also called "implicit") noise functions such as Perlin noise [17] (see also simplex noise [5, 15]). However, the planets of *Noctis* are almost exclusively devoid of life, and there is no meaningful in-game social or cultural interaction. What would it take to add intelligent, social beings to *Noctis*'s billions of planets? It would require a similarly randomly-accessible generation function for societies and cultures.

Games such as *Dwarf Fortress* and *Ultima Ratio Regum* have approached this problem using mixed generation approaches, where some attributes are randomly accessible (and can thus be abstracted and then re-generated to save space) while others are generated once and then stored as permanent properties of the world (see e.g. [7], although not much detail is provided). More academic games such as *PromWeek* and *Bad News* have also taken on the challenge of generating detailed social relationships, but have either used carefully hand-crafted starting scenarios or ad-hoc generate-and-store approaches [13, 18]; a fully random-access approach has not been attempted. For a family tree, a random-access approach would be comprised of a set of functions which given a person could find their biological parents, their children, and which partner(s) they had each child by.

Before presenting such functions, several grounding assumptions are useful: first, we can identify each "person" as an integer, so we are looking for functions that accept and return these integer IDs. Second (and we will revisit this), we can assume that half of our population is male and half is female, and without loss of generality let even-numbered IDs be female and odd-numbered IDs be male. Now if we generate a random integer, we can find a "female version" by subtracting one if it's odd, and a "male version" by adding one if it's even. Finally, we will assign each ID a birth-date in sequential order, with a parameter that controls how many births happen in the world per day (another assumption to revisit, as it means that population neither grows nor shrinks over time).

Now the computation of one's biological mother is simplest, as each child has exactly one mother (perhaps our only realistic-enough assumption). We can "just" pick an appropriate age range of children and shuffle[1] our ID within that many children (which we'll call a "cohort"), then subtract an appropriate number in ID-space corresponding to the birth-rate-per-day times the desired-age-difference-in-days to find a mother ID. If each child uses the same shuffle seed and the same cohort boundaries, there will be no overlaps after shuffling, and each mother will have exactly two children (we assign "children" of male IDs to the corresponding female ID). In order to have a variable number of children per mother, we can first use the above technique to map small cohorts of mothers and children together (perhaps groups of 32) and then randomly subdivide the children while uniformly subdividing the parents into two groups, recursively assigning the first and second parent groups to the first and second child groups. This subdivision algorithm can end up with empty child groups (the split point among children is random), in which case those parents have zero children, but since we started with 32 parents and 32 children, this will be compensated for by another parent in the same group having more children. This process gives us two functions: one for finding a parent ID given a child ID, and its inverse, for finding a child ID given a parent ID. The inverse is computable because all of the individual operations (shuffling, subtraction, and subdivision) are reversible (both parents and children need to use the same seed for the shuffling and random subdivisions, of course). The inverse function results in a few contiguous children, which can be ordered arbitrarily. In order to avoid the situation where all children have the same/contiguous birthdays, the child cohort can be pre-shuffled once (and then un-shuffled for the inverse function) so that contiguous results become scattered.

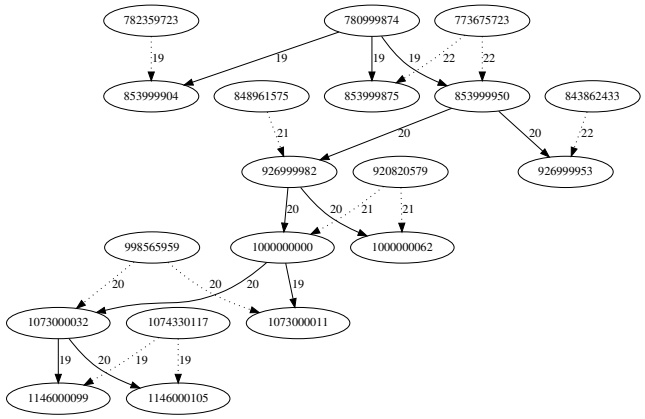

Figure 1: Graph of family relationships. Dotted lines connect fathers to their children, while solid lines connect mothers. Edge labels are the age-at-birth of the parent.

Given these parent and child functions, we can also add a partner function, which determines the partner by which each mother had each child. This function works similarly to the mother-selection function: shuffle the men in cohorts, then match them with women (of about the same age). An extra step here is required to allow multiple partners: first, each woman is assigned a random number of partners (perhaps with a bias, again more about this in the next section) between one and the number of children she has. Next, different seeds are used to shuffle the men, so that each woman has multiple partner-candidates (up to 32, if that's the maximum number of children per mother). Then based on the mother's number-of-partners, she picks partner candidates starting at a random index as her actual partners. To invert this and find a

[1]The field of cryptography has extensive literature on reversible permutation algorithms, see e.g. [19].

child-bearing partner from a male ID, you need to check 32 potential partner IDs to see which of them (if any) actually chose you, but while expensive this process is still local and constant-time (it depends only on the max-children-per-mother parameter).

These invertible parent/child and mother/partner functions can be used to generate an arbitrary region of a nearly infinite (limited by integer bit-widths) family tree. One such region, composed of six generations starting with the maternal great-grandparent of ID 1,000,000,000 is shown in Figure 1. Note that this figure used limited child shuffling, so child ages are extremely close together, but this could be fixed as described above.

3 LIMITATIONS & TRADEOFFS

As pointed out repeatedly in the previous section, the algorithms discussed there are shot through with problematic assumptions. These algorithms were developed to quickly achieve a semi-realistic result (and even then age gaps are a serious problem), but that process resulted in poor realism and representation. Note in particular that the desired scope of generation (entire galaxies full of people) means that representation issues are realism issues too: if the algorithm ignores "statistically rare" occurrences, that will be noticeable on the galaxy scale (of course, at smaller scales where it's "not noticeable," it's really only "not noticeable" to uncritical audiences from majority groups, so this is still a representation issue). To highlight several issues:

(1) **Male:female ratio**: The male-female ratio at birth is not 1:1 in the real world (see e.g. [8]). Beyond that problem, there are two more: first, even if we assume this applies to biological sex only, humans have a rich range of possible sexual expression, and perhaps 1.7% of people are not correctly described by either "male" or "female" (see [2]). Second is the issue of gender: again a substantial percentage of the population is not well-defined by the terms "male" or "female" [3], and this should have an impact on family structures. The algorithms described above thus have both realism and representation problems in terms of both sex and gender.
(2) **Births-per-day**: The assumption that the number of births per day is constant means that the population does not grow over time. This has essentially never been the case throughout history, and could lead to a number of representation problems beyond the obvious realism ones: if the population is constant, then it will presumably experience resource competition in a very different way from real populations, leading to potentially misleading procedural messages. Unfortunately, although births-per-day could be defined as a function of strict-ID order to create a correctly exponential population, this leads to all sorts of problems with the shuffling and matching algorithms that are supposed to create our family structures. The births-per-day constraint also distorts family structures, creating more people who never have kids than is realistic. Although this may seem like an unimportant change, small changes in social structures can have far-reaching effects (see e.g. [16]), and this gives the algorithm above more potential to represent a distorted view of society.
(3) **Independent/Conflicting Births**: The algorithm above assumes that each child has one mother, but this selection is independent of the other children of this mother. This could lead, for example, to two children being born a few days or weeks apart, which isn't physically possible. This also means that there are essentially no patterns to child birthdays for a given mother, which is not at all realistic. Ideally, an exponential distribution of child birthdays relative to mother birthdays could be achieved, but this is technically difficult.
(4) **Partners-per-mother**: The assignment of partners-per-mother is obviously a sensitive one, but the algorithm introduces further problems because the birth-ordering of the children is arbitrary, so there's no easy way to express a bias towards consecutive births by the same partner. This is both a realism problem, and without careful framing, potentially a representation problem.
(5) **Other problems**: Although I've pointed out some of the most obvious hang-ups with this simple algorithm, there are doubtless more problems that could arise (for example, how prevalent is incest using this algorithm). Because the system is procedural, some may not be discovered until the system is in production and exercised fully. By acknowledging the problems we *can* find up-front, however, we can hope to improve both realism and representation.

Although the problems above seem difficult, there are actually a variety of remedies available, some of which synergize with each other. Some example solutions:

(1) **Re-assigning childless people**: This idea addresses both problems 1 and 2 above. Instead of trying to add random variables which control gender/sex expression and then preempt or alter family formation, a sample of parents who did not have biological children under the current system can be assigned appropriate identities (e.g. intersex, genderqueer, asexual/aromatic, etc.). Of course, a (probably smaller) percentage of people who do have biological children also express these identities, but an appropriate balance could be achieved by assigning these identities post-hoc instead of before family formation. The inflated percentage of childless people in the constant-population system is an asset here, as it ensures identities that are anti-correlated with bearing children can be represented at realistic levels.
(2) **Sex & gender fluidity**: Although it's an irrevocable fact of this system that each child has exactly one biological father and one biological mother, those people don't need to be exactly male and female, either in sex or in gender. A more accurate pair of terms is "child-bearer" and "partner," and so the even/odd assignment described above actually uses these terms. Regardless of sex and gender identity, there's still one person who bears each child[2], and again, post-hoc sex and gender assignment can work from this requirement to a variety of desired demographic distributions.

[2]A counter-point here would be the existence of conjoined twins, which is quite difficult to deal with, but could perhaps be handled using a shared-ID model.

(3) **Exponential Populations**: As mentioned above, the technical barriers to exponentially growing populations are significant, but if these were overcome, such techniques could most likely also be used to distribute child ages, and a more realistic exponentially growing population could be used as a backdrop for stories that addressed problems like colonialism.
(4) **Birth Merging**: Although some statistical balancing might be needed, the issue of conflicting births can be addressed by assigning actual birth-dates in addition to nominal birth-dates. If the birth-date mentioned above is one's nominal birthday, then all children of each mother can inspect each other to find any conflicts, and these can be resolved by setting conflicting birth-dates equal to the minimum of the group, creating twins, triplets, etc. as necessary. This adds a dimension of realism/representation (the existence of twins and triplets) while also addressing the problem of conflicting births.
(5) **Re-Framing**: The issue of multiple partners per mother is readily controllable, and thus can be made realistic and tackled by framing it appropriately. To reiterate an earlier point, although such framing is not easy, the alternative of erasing complicated identities is generally more harmful because of how it reinforces dominant stereotypes.

The lists above are hardly exhaustive, but I hope that they can inspire conversation around these topics. As a particular example of a generative algorithm over a social space, this randomly-accessible family-tree-generator has both unique technical constraints and its own set of challenges in terms of realism and representation. Sometimes the technical constraints are the source of problematic defaults (like assigning half of IDs to be child-bearers or non-child-bearers in order to produce efficient pairings), but at other times they actually give space for diverse identities to be expressed (for example, a fixed-size population assumption leaves plenty of childless people who can represent parts of the gender spectrum which don't usually have children).

4 CONCLUSION

Ultimately, I view the technical challenges associated with improving representation and realism as inspirational. Having been forced to think technically about how to model mothers who bear children with multiple partners, I am inspired to assign them a diverse set of explanations for and reactions to that identity. That diversity in turn could create a much more thought-provoking world for players to explore, and could even be the basis of social game mechanics specific to those people. In the end, the resulting game world is much richer for the diversity included, even, or perhaps especially, if it is rare. These technical mechanisms for representing diverse identities can also be used for intentional representational choices. A generative model that can represent genderfluid people out of necessity is also one that can construct imaginary societies of great gender fluidity in service of an author's expressive goals.

The larger point about generative methods to be made here is that within generated game worlds, the algorithm is a powerful arbiter of normalcy, and thus a natural vehicle for the expression of stereotypes. This is a huge responsibility in terms of the design of these algorithms, because perpetuating the biased and broken stereotypes that are expressed so often in popular culture is sometimes easier than subverting them. But PCG systems are a powerful tool for subverting stereotypes as well, because they by their very nature enable diversity. And when players encounter something "outside the norm" in a game, that moment has the ability to teach as well as surprise and excite.

By publishing this analysis of how efficiency, realism, and representation intersect in generated content, I hope to provoke a discussion that goes beyond the specific algorithm I used as an example here. What technical limitations have hindered realism and/or representation in your work, and how have those led to generating more diverse spaces? Are there specific state-of-the-art techniques that could be improved upon in terms of representation (broadly), and how might such improvements lead to other changes in player experience and/or creative control?

REFERENCES

[1] Tarn Adams. Slaves to armok, god of blood chapter II: Dwarf fortress. Bay 12 Games, 2006. Microsoft Windows, Mac OS X, Linux.
[2] Melanie Blackless, Anthony Charuvastra, Amanda Derryck, Anne Fausto-Sterling, Karl Lauzanne, and Ellen Lee. How sexually dimorphic are we? review and synthesis. *American Journal of Human Biology*, 12(2):151–166, 2000.
[3] Gary J Gates. How many people are lesbian, gay, bisexual and transgender? *Technical report, The Williams Institute, UCLA School of Law*, 2011.
[4] Alessandro Ghignola. Noctis. Self-published, 2000. Microsoft Windows.
[5] Stefan Gustavson. Simplex noise demystified. *Linköping University, Linköping, Sweden, Research Report*, 2005. URL http://webstaff.itn.liu.se/~stegu/simplexnoise/simplexnoise.pdf.
[6] Mark Johnson. Ultima ratio regum. Self-published, 2012. Microsoft Windows.
[7] Mark R Johnson. Modelling cultural, religious and political affiliation in artificial intelligence decision-making. In *Proc. AI & Games Symposium*, 2015.
[8] Piet Hein Jongbloet, Gerhard A Zielhuis, HM Groenewoud, and Pieternel C Pasker-De Jong. The secular trends in male: female ratio at birth in postwar industrialized countries. *Environmental Health Perspectives*, 109(7):749, 2001.
[9] Claudia Lo. How rimworld's code defines strict gender roles, 2016. URL https://www.rockpapershotgun.com/2016/11/02/rimworld-code-analysis/.
[10] Michael Mateas and Andrew Stern. *Architecture, Authorial Idioms and Early Observations of the Interactive Drama Façade.* School of Computer Science, Carnegie Mellon University, 2002.
[11] Michael Mateas and Andrew Stern. Façade. Self-published, 2005. Microsoft Windows, Mac OS.
[12] Maxis. The sims. Electronic Arts, 2000. Microsoft Windows; various.
[13] Josh McCoy, Mike Treanor, Ben Samuel, Michael Mateas, and Noah Wardrip-Fruin. Prom week: social physics as gameplay. In *Proc. of the 6th International Conference on Foundations of Digital Games*, pages 319–321. ACM, 2011.
[14] Mojang. Minecraft. Mojang; Microsoft Studios; Sony Computer Entertainment, 2011. Various platforms.
[15] Mark Olano, JC Hart, W Heidrich, B Mark, and K Perlin. Real-time shading languages. *Course Notes. ACM SIGGRAPH*, 99, 2002.
[16] Chai Bin Park and Nam-Hoon Cho. Consequences of son preference in a low-fertility society: Imbalance of the sex ratio at birth in korea. *Population and Development Review*, 21(1):59–84, 1995. ISSN 00987921, 17284457. URL http://www.jstor.org/stable/2137413.
[17] Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3): 287–296, 1985.
[18] James Owen Ryan, Adam Summerville, Michael Mateas, and Noah Wardrip-Fruin. Toward characters who observe, tell, misremember, and lie. *Proc. Experimental AI in Games*, 2, 2015.
[19] Babak Sadeghiyan and Josef Pieprzyk. A construction for super pseudorandom permutations from a single pseudorandom function. In *Advances in CryptologyfiEUROCRYPTfi92*, pages 267–284. Springer, 1993.
[20] Valve South. Left 4 dead. Valve Corporation, 2008. Microsoft Windows, XBox 360, OS X.
[21] Ludeon Studios. Rimworld. Ludeon Studios, 2013. Microsoft Windows, Mac OS X, Linux.
[22] Michael Toy, Glenn Wichman, Ken Arnold, and Jon Lane. Rogue. Epyx, 1980. Various platforms.
[23] Li-Yi Wei and Marc Levoy. Order-independent texture synthesis. Technical Report TR-2002-01, Stanford University Computer Science Department, April 2002.