Appendix A
Game-designer interviews

To complement the technical content of the chapters, all written by academics (though some of the chapter authors also design and develop games), we performed five interviews with the creators of well-known PCG-heavy games. We selected the interviewees mostly because the games they had been part of had either introduced new interesting PCG techniques, or because they had integrated them in game design in some novel way. The interviews were performed in 2013 and 2014 over email, and are reproduced in their entirety here (except for corrected typos). We asked most of the interviewees the same set of questions, focusing on the role of PCG in game design and the limits of generative methods.

The interviewees are:

- **Andrew Doull**, creator of *UnAngband* and *UnBrogue*, founder of RogueLikeRadio.
- **Ed Key**, creator of *Proteus*.
- **Michael Toy**, co-creator of *Rogue*.
- **Richard Evans**, AI lead programmer on *The Sims 3*, co-creator of *Versu*.
- **Tarn Adams**, creator of *Dwarf Fortress*.

A.1 Andrew Doull

Was there anything you wanted to do in a game you worked on that you could not do because of algorithmic or computational limitations?

My thinking about game design has changed significantly over the years, since writing the original Death of the Level Designer series of articles. One of the key changes—guided a lot by games like Michael Brough's 868-HACK—is that a game should embrace limitations, rather than attempt to design around them. So rather than creating procedural systems which can model everything à la Dwarf Fortress,

there are real advantages in keeping games as limited as possible in order that the significance of individual procedural elements is emphasized, rather than smoothed over by a melange of inputs. My personal limitations are very much around algorithm implementation rather than design: for instance, while I have a good understanding of what a Voronoi diagram looks like and where it could be used, I'm unlikely to ever successfully reimplement anything but a brute force approach for calculating cell membership.

What new design questions has PCG posed for some game you worked on?

I've written extensively about this—refer to the writing on my blog post on Unangband's dungeon generation and algorithmic monster placement for specific discoveries. Since then, UnBrogue includes very little original procedural content: I mostly plug new values into the well written framework that Brian Walker has developed for Brogue's "machine" rooms. These days I try to steer clear of actually designing procedural systems: my experience is that you can achieve a lot using a very simple set of algorithms, provided you choose your content carefully (see Darius Kazemi's essay on Spelunky's level generation for a great example of this).

What is the most impressive example of procedural content generation you have seen since your own work?

I'd be hard pressed to ignore Miguel Cepero of Voxel Farm, who I'm sure a lot of people you interview will mention. While the above ground system looks great, it was his cave designs that won me over, after being a doubter. What really impresses me though is he's developing all this while being the father of twins—I'm in the same position and I can never find the time...

What do you think of the fact that roguelikes have become a genre of their own? Is PCG in your opinion an essential part of what a roguelike is?

I'm going to quote Edmund McMillen here, since he made the definitive statement on why you should write a roguelike:

"The roguelike formula is an amazing design plan that isn't used much, mostly because its traditional designs rely on alienatingly complicated user interfaces. Once you crack the roguelike formula, however, it becomes an increasingly beautiful,

deep, and everlasting design that allows you to generate a seemingly dynamic experience for players, so that each time they play your game they're getting a totally new adventure."[1]

PCG is obviously an important part of this process, but it isn't independent from the other roguelike genre features like permadeath.

My hunch is that there are other "design plans" out there that are waiting to be found that will feature PCG—in fact the majority will do, but we don't necessarily know what they look like, or have the maturity of the medium (of PCG) to be able to discover them.

What are your tips for designing games that use PCG?

Keep your algorithms simple and choose your content carefully.

Do you have any interesting stories about PCG failures?

Not personally, since I've taken such a conservative approach to PCG algorithms.

In general, is there anything in a game you think could never be procedurally generated?

The specific quirks of the real world. I'm not saying that PCG can't create something like the real world: I expect the depth required to make a world "completely convincing" is actually more shallow than most PCG "haters" realize—but ultimately, when it comes to simulation, the fact that we exist at a specific time and place in a continuum of choices and random events is something that PCG can only hold a mirror up to. Hand placed design will always be needed if you want to model history—PCG will overtake hand placed design for "fantasy worlds" in the not too distant future.

Why is PCG not used more?

PCG is a language that requires a level of literacy to understand. We're not effectively teaching this language yet, but we're not effectively teaching the language of game design in general either. Also, it is often more expensive than hand placed con-

[1] http://www.gamasutra.com/view/feature/182380/

tent, because a PCG algorithm which is only 90% complete can not create anything useful, whereas 90% accurate hand placed content is clearly 9/10ths done. It's hard to describe working with PCG this way, but there's almost a phase change between when a PCG algorithm just creates junk, and when it starts producing beautiful results, and it can be very hard to tune it to reach this state.

What do you see as current directions for PCG that are worth investigating?

There's a lot of interesting stuff happening on the academic side—getting this to percolate over to game development is going to be the real challenge.

A.2 Ed Key

Was there anything you wanted to do in a game you worked on that you could not do because of algorithmic or computational limitations?

At first there was: At one point Proteus was going to be some kind of sandbox RPG with generated towns and quests. Once I started talking to David about music, we reshaped the game as being all about music and exploration, and also at this point started to find and work with the "grain" (as in carving wood) of what we had, shaping what we wanted to do to the medium we were working in.

You can extend this to Proteus being an island rather than an infinitely streaming world. The latter would have been technically much harder but also not really desirable once we decided to focus on a finite space that allowed you to get a little lost but was also bounded and so allowed some familiarity and revisiting of locations.

Do you have any tips for designing games that use procedural content?

Think about framing, structure and pacing. Consider how in the classical example of Rogue, the procedural generation is incredibly simple and designed as a kind of "lumpy canvas" for the authored elements (creature, potions, etc.) to interact. I think the most successful PCG applications understand how the procedural content is framed and given context by authored content, or sometimes by human curation. Think of procedural generation as poetry or music and make use of the player's

imagination and faith rather than trying to create results that withstand point-by-point examination.

One strength of PCG is to create "wildness"—either mimicking or evoking nature or in glitch aesthetics. On the other hand, formal disciplines like architecture provide patterns that PG systems can use, but I think you still need something in the "fiction" of the game to make freakish "wrong" results something appealing rather than bugs that break immersion.

Something I'm really keen on in game design is how to create "substances" or things that "feel substantial". I think a lot of this is about establishing scope and language early on and sticking to these as a contract with the player. Of course, you can subvert those expectations, but first you need to establish them.

In general, is there anything in a game you think could never be procedurally generated?

Stuff like human behaviour is always going to be hard. My solution to this is to have the PCG operate at a level of hints and suggestions instead of trying to generate fully detailed characters, behaviour and artefacts. If the player is invested enough to fill in the gaps with their imagination this will be a much richer experience than if they are just given all the details and their mind unengaged and free to pick holes in those.

There's a deeper issue in that "meaning" can never be created by a computer system, in my opinion. "Meaning" arises in the mind of a conscious being and is about how the player reads and interacts with the game. On the other side it's about what you as the architect of the PG system put into it—values, aesthetics, etc. Humanity is paradoxically extremely important in this domain.

What is the most impressive example of procedural content generation you have seen since your own work?

My friend Alex May is doing some beautiful stuff with procedural trees[2]. For something that's released and generating a full gameworld, maybe this PG stuff by Tom Betts[3]. No Man's Sky is also extremely enticing but hard to separate hype and expectation from the actual product at the current time (Jan 2013).

[2] http://blog.starboretum.com/

[3] http://www.big-robot.com/tag/sir-you-are-being-hunted/

What do you think of the fact that roguelikes have become a genre of their own? Is PCG in your opinion an essential part of what a roguelike is?

Well, roguelike when I first knew it was permadeath and proc-gen ascii dungeons. Now we have a whole spectrum of roguelike-likes including FTL, Don't Starve, etc. I think the genre already existed but has become broader, whilst at the same time procedural techniques are spreading and growing in all genres from FPSes to interaction fiction. I would say that yes, PCG is essential to a roguelike, but it's always interesting to take that as a challenge. Maybe the great PCG-free roguelike is Dark Souls? No-one calls that a roguelike, and I think it wouldn't work if it was procedurally generated, but it seems to share a lot of the flavour of "punishing exploration". It's interesting to think about how Dark Souls would be *worse* if it was proc-gen. Places in the world would have less resonance, and players wouldn't be able to share stories or advice in the same way.

A.3 Michael Toy

Was there anything you wanted to do in Rogue that you could not do because of algorithmic or computational limitations?

The limited size of programs on the PDP 11/70 (64 kilobytes), kept us from implementing the variety of AI driven monsters that we had imagined in the design phase.

What is the most impressive example of procedural content generation you have seen since your own work?

Have to tip my hat to Dwarf Fortress. The story-telling and emergent properties are marvelous. And the game Moria was probably the closest to what we had imagined doing when we started writing Rogue. I'm sure there are more, I am not an expert in the field, but there's my answer.

What do you think of the fact that roguelikes have become a genre of their own? Is PCG in your opinion an essential part of what a roguelike is?

The word "roguelike" belongs to the community that invented the word, so I don't claim any special authority. However the initial design goal for Rogue was to produce a game that avoided two problems, and the two solutions resulting are often stated in the definition of a roguelike, PCG and permadeath.

The first problem was that having written several text adventures, it eventually (it should have been sooner, we were young and stupid) became clear that it was never going to be fun playing a game where you knew everything. So the quest became to try and make a game where even the creator of the game is involved in a quest for discovery.

Second we wanted to avoid the "Dragon's Lair" problem where winning the game is just running until you die, then backing up and doing something different, repeated endlessly. We allowed saved games so you could stop and go to class or eat, but worked hard to dis-allow people from re-playing from a save point repeatedly, not because we were trying to create permadeath precisely, but because we wanted the in-game consequences to matter. If a player decided to take a small or a large risk, we wanted that risk to be a more real risk than simply the risk that you might have to restore from the save file. This then made the rewards more meaningful also. It wasn't just permadeath, but perma-everything.

I actually see PCG and prevention of reverse time-jumps as being inseparable. If I can save the game, explore a level, restore at the save point and explore the level again only "correctly", the entire point of the PCG is missed.

What are your tips for designing games that use PCG?

I think PCG changes how you think about the world-writing for a game.

One of the surprises for us in writing Rogue is how little PCG it took to create a game which people could play for hundreds of hours. We really barely got working what we thought was the base game, and suddenly it was popular and everywhere, before we got to what we had previously thought was going to be the part which made it interesting.

In a sense a game of Rogue is a collaborative storytelling exercise. You don't know how it is going to end, though you suspect it will be a tragedy. People have imaginations and that can be a huge advantage to game designers. Rogue allowed people to write their own scripts about what was going on, and provided them all the action scenes for their story.

In general, is there anything you think could never be generated?

I always dream of PCG worlds as rich and beautiful as the best hand-modeled worlds. Not because I think the modeling is trivial, or even possible to do, but because I love the feeling of stepping into something that has never been seen before. The problem is that a world which takes your breath away is not just doors and walls, it is cultures and civilizations. Even traditional games rarely invent these things, but just re-skin the ones we already know about. Can you ever generate something like walking through a jungle and discovering ruins in a style that no human has seen before?

A.4 Richard Evans

Would you describe The Sims 3 as doing procedural narrative generation? What about Versu?

It depends, as always, on your criteria.

The Sims games create a broad range of possible permutations of behaviour. Often, the generated behaviour sequence is everyday—but sometimes the behaviour sequence seems to conform to a narrative. The richer the personality model, and the deeper the social simulation, the more likely this is to happen. Certainly, some people did create narratives just by sitting back and watching The Sims 3. For example, Robin Burkinshaw created Alice and Kev: a great blog describing the plight of a couple of homeless Sims. He set up an initial situation (a father-daughter pair, who were both homeless), and then sat back, watching and recording the events as they occurred.

Versu procedurally generates narrative. At the drama-manager level (the level of scenes), there is a moderately rigid story graph of scenes with pre- and post-conditions. But within each particular scene, the individual agents are free to choose their own actions, based on their own desires.

Interactive storytelling originates in its own separate community. As interactive stories develop more generative procedural-storytelling systems, do they become a kind of procedural-content domain?

Interactive stories often have hand-written content at the drama-manager level, but variation and procedurality within the scene. This is a limited form of procedurality existing inside a constrained hand-written framework.

Was there anything you wanted to procedurally generate in a game you worked on that you could not do because of algorithmic or computational limitations?

Yes. One thing I really wanted the computer to do was to generate a social situation that was already half-way through. So if, for example, the player turned up at a bus-stop, there might be an argument between a boy and a girl that was already almost finishing. This ability, to create social situations in media res, is not something that the Versu simulator is able to do.

What is the most impressive example of procedural content generation you have seen?

Ooo I don't know. There are so many recent exciting examples. Procedurally generated platform levels, music composition, puzzle games—it's a very exciting time.

Do you have any interesting stories about procedural-content failures?

The richer the simulation, the more possible causal pathways—and the harder it can be to understand why something is happening. During development of Versu, I had a tricky bug where half way through a murder-mystery, the doctor was being rather rude to my player character. I know that the doctor did not have an abrasive personality, and my character had never done anything rude to the doctor, so it was hard to see why the doctor was behaving this way. It turned out, after much debugging, that the reason was this: at the beginning of the game, my player character had been dismissive to one of the servants who was waiting at the table. The servant had gone back to the kitchen, and had told the others about my rude behaviour. The doctor, being a friend of the servant, had believed the servant's testimony and had formed a negative judgement about my player's character. This sort of example shows how emergence is a double-edged sword: it generates new stories, some of which are not anticipated—but can also make it harder to understand what is happening.

In general, is there anything in a game you think could never be procedurally generated?

In Versu, we generated text from templates (e.g. "[X] look[s] towards [Y obj]"), substituting proper names and pronouns for variables (generating e.g. "Jack looks

towards her"). What would be significantly harder—but also significantly more flexible—would be to generate text without templates—using e.g. a phrase-structure grammar and semantic constraints.

A.5 Tarn Adams

Was there anything you wanted to generate in Dwarf Fortress, or another game you worked on that you could not do because of algorithmic or computational limitations?

Most of the algorithms are scalable, so it's really that almost everything needs to be kept smaller than we'd like. Things like time travel are difficult to do in a proper fashion in DF since the amount of data is extreme, and even a small perturbation wouldn't be believable if it didn't have a lot of data to back it up. Fluid dynamics are difficult, and our system is pretty lame due to computational problems (and algorithmic/scientific cluelessness for anything complicated there). All of the conversation AI is very basic and will likely be held back by a lack of good ideas on my part and also my dislike of generating a lot of English sentences. The entire frontier of what we haven't done in DF is made up of our limitations along these lines, when it isn't just time constraints.

What new design questions has PCG posed for some game you worked on?

One of the interesting ones is the matter of presentation. If you generate most of the content in the game, and it doesn't hew to traditional lines, you have to be careful about how you unfurl it to the player. We're just getting started with this consideration now as we start making more generated creatures and plant-life and materials, but the game would become gray mush if we aren't mindful as we move away from pre-defined content. A very simple example is the paragraph description it pops up whenever a forgotten beast attacks your fortress—if the player were attacked without being forced to look at a description, I think it would become quite confusing as the random attacks and other properties of the creature come into play, though there's a lot of wiggle room and different methods that could be tried out. When we allow the game to replace regular wilderness creatures or the standard fantasy races (elf, goblin, etc.) or even the playable race, the roll-out of the random characteristics is going to be front and center... almost tutorial-worthy in some cases.

There's also the matter of the realized map area which has come up a lot—sometimes Dwarf Fortress has pieces of the world loaded up at five different levels of abstraction (or more?), and each of those need to mesh with each other and be

chosen so that crucial details aren't lost but also so that memory and speed are under control. This can be difficult to manage, and sometimes we have to make choices that make the game suffer—this would relate back to the first question regarding algorithm scalability I guess. You can often accomplish a lot of what you want to accomplish without doing a perfect job by just scaling back the loaded area a bit, or keeping an abstract version of a larger area loaded (whether that's a map area or some other concept).

What is the most impressive example of procedural content generation you have seen since your own work?

The cities from Subversion, maybe? Although it wasn't fully realized, it seems like it would have been cool. Seeing the game unfold in Drox Operative was neat, although if you view yourself as an RPG camera in a strategy game there, it probably doesn't stand out as an AI example. The complete package was interesting though.

What do you think of the fact that roguelikes have become a genre of their own? Is PCG in your opinion an essential part of what a roguelike is?

I don't know that genres are ever healthy, but it's cool to see more games. I don't have a definition for "roguelikes", and it's a popular subject for argument, but I can't think of a game without map-related PCG that I'd ever casually call a "roguelike" to somebody in conversation. Other people focus more on permanent death, save states, and other features, though, and for all I know Gauntlet is a "roguelike" now.

What are your tips for designing games that use PCG?

For all of these, there's the caveat that rules are meant to be broken after looking at the bigger picture, and also that the tips grew out of mistakes I still make which are evident in my games. So: Don't simulate more detail than you need to get your point across— the elements involved in the PCG should be game elements, atmospheric elements, etc.—if you don't need the molecules bumping around, invisible to the player, try to stay away from chewing up computer resources and programming time putting that in. It's fine to go one level deeper if the "phenotype" that arises from your procedural DNA turns out well, but that's a matter of happy accidents as much as planning, and you'll have difficulty refining your game if you subject

yourself to too much chaos theory (or to too much going on that just doesn't affect anything).

Keep in mind what the game is trying to accomplish overall. If you can afford it, don't substitute crappy PCG for a single, better hand-crafted asset (unless there's a really solid counter-balance, say, in replayability, then it is a matter of taste)—at least if you are polishing up your game, since experimentation is crucial at first and you might arrive at something really satisfying. PCG does not automatically increase replayability—a full play through a great pre-defined game is better than a full play through a shoddy random game that you won't touch again. I haven't always been able to do the following in a timely fashion, since it can hamper experimentation, but if parts of your game are moddable, I think it's good to keep your own internal PCG in line with the moddable format (to keep the standards uniform if anything else)—for example, PCG Dwarf Fortress creatures and materials are made by producing a text definition which is interpreted in the same way as pre-defined or modded text definitions.

I also think it is good to try your hand at your own algorithms when possible, since the output will have more character than something recognizable as Perlin noise or a Voronoi diagram etc., though just having anything you can iterate on is probably good enough, and of course existing algorithms form an important part of your PCG skills to build on. Sometimes you can get better results faster just by plowing ahead, though, rather than fishing around for something online. Most things haven't been tried yet, and there's a wide frontier of PCG in games to explore. If you are simulating something that can be related to a real-world process, keep that process in mind when you are trying to correct unacceptable defects in your output—the answer is often in some missing variable or relationship that the real-world analogy makes clear.

Do you have any interesting stories about PCG failures?

They're mostly interesting from the humor angle, since things often go terribly wrong. I'm not sure what would be interesting for the experts or people interested in making better PCG. My process is very iterative, and it's difficult for me to remember discrete instructive moments.

In general, is there anything in a game you think could never be procedurally generated?

You can view everything produced by people throughout history as procedurally generated in a larger context, so I wouldn't leave anything out in general, although there's probably a Gödel-ish proof sitting around that you can't PCG everything from computer algorithms. Some things are certainly more difficult than others.

Prose and conversations and so on can be rough, especially as it relates to AI (since that's just the Turing test more or less), and chaotic behavior that comes from many small parts (like fluids or weather) is probably not possible since you'd need to simulate the molecular behavior properly to hit upon the actual effects (though you could use a "good enough" test like the Turing test, for dynamic behaviors vs. human observation of them). So in any case, the actual limits of PCG probably aren't important yet, and I don't think supposed limitations related to being able to match pre-defined human artwork in terms of emotional impact or symbolic significance or whatever else should deter anybody from exploring what's possible.

Why is PCG not used more?

People are using it now more than I've ever seen, so I'm not sure this one is answerable from my perspective. If current PCG techniques don't measure up, it's prudent to stick with hand-crafted text and graphics and music from a financial and overall quality perspective, certainly, rather than trying to tackle everything with PCG to your satisfaction in the time you have available for your project.

What do you see as current directions for PCG that are worth investigating?

I think people are already jumping into everything, in one way or another, and nothing should be off limits there. I look at PCG as an almost universal candidate for feature implementation (since I don't have useful skills for producing non-PCG material), so I wouldn't close off or prefer any avenue. There's a good game to be made regarding any subject, and PCG can be involved in those.