The Death of the Level Designer

Andrew Doull

January 2008
http://pcg.wikidot.com/the-death-of-the-level-designer

Procedural Content Generation in Games

Procedural content generation is yet to set the game industry on fire. It has featured in one of the greatest games of all time, Diablo[1] and it's successor, who directly trace their roots to roguelike games such as Angband. But the recent implementation of random level generation in Hellgate: London[2] did little to inspire people that this method works well for game level design. But bubbling under the surface of the industry, and very much evident in future games like Spore is a methodology that like most technologies has been underwhelming in the short term but in the long term will have profound consequences for how games are designed.

An announcement late last year should have set the hearts of anyone working in the games industry a tremor. No, it wasn't Activision merging with Vivendi. Gearbox Software previewing use of procedural content generation in Borderlands[3] was met with moderate skepticism, but was generally well received. It enabled me to remind[4] at least one other person of the greatness of the line 'Guns. We need lots of fucking guns.', predating Neo's similar but censored sentence by seven years (The movie in question was Split Second[5]). But more importantly, it points the way forward to a time where the current role of the level designer will be as obsolete as punch cards for programming computer games.

This article will survey the current scope of procedural content generation in order to highlight where Procedural content generation (PCG) is making inroads into the traditional level designer roles, and how various enabling technologies are making procedural content easier to build and deploy within other game technologies. The survey will point in a number of directions, not necessarily positive, that I anticipate procedural content generation taking game play, and conclude with some suggestions what the next five years will bring, and a few options for anyone interested in taking the ideas herein further.

PCG is the programmatic generation of game content using a random or pseudo-random process that results in an unpredictable range of possible game play spaces. I prefer the term procedural content generation to procedural generation: the wikipedia definition[6] of procedural generation includes using dynamic as opposed to precomputed light maps, and procedurally generated textures, which while procedural in scope, do not affect game play in a meaningful way. The concept of randomness is also key: PCG should ensure that from a few parameters, a large number of possible types of content can be generated.

Procedural content generation can exist in games in a number of ways.

1. Runtime random level generation

This is the 'prototypical' example of procedural content generation, which is what most of you would have thought of when you starting reading this article. The classic example is of Rogue's random placement of rooms linked by corridors, which makes the game infinitely replayable and softens the penalty of permadeath. I would suggest that dynamic random level generation in two dimensions is still an interesting area of research: I've written extensively about it elsewhere[7], but there are still algorithms to be discovered and techniques to be used. The problem is however mostly solved in the sense that there is a robust variety of interesting algorithms from maze generators to room placers to choose from and considerable literature covering the implementation methods.

In contrast, procedural content generation in 3 dimensions is still a relatively undeveloped field. Hellgate: London is an example of how not to do three-dimensional design – the cost of generating sufficiently interesting 3D objects and textures ensures that there is a great deal of repetition in the resulting random levels. It is possible to extend many of the two dimensional concepts into the 3rd dimension: maze generation is mostly extendable depending on which algorithm you choose, and the recent releases of Dwarf Fortress[8] shows that it is possible to extend the classic roguelike design into three dimensions.

Procedural generation of height fields in 3D is the only other area that has had considerable effort devoted to it. Probably the best in-game example is Tribal Trouble[9] which uses a simulation of erosion[10] in order to create a play field resulting an infinite number of multi player maps. There are a large number of fractal-based world generators which use similar techniques to create vast and elaborate world spaces, which to a greater or lesser degree use simulation of

[1] http://en.wikipedia.org/wiki/Diablo_(video_game)
[2] http://en.wikipedia.org/wiki/Hellgate:_London
[3] http://en.wikipedia.org/wiki/Borderlands_(video_game)
[4] http://www.rockpapershotgun.com/?p=724
[5] http://en.wikipedia.org/wiki/Split_Second_(1992_film)
[6] http://en.wikipedia.org/wiki/Procedural_content_generation
[7] http://roguelikedeveloper.blogspot.com/2007/11/unangband-dungeon-generation-part-one.html
[8] http://www.bay12games.com/dwarves/
[9] http://tribaltrouble.com/
[10] http://oddlabs.com/download/terrain_generation.pdf

geological and meteorological processes to design the map height-fields and populate them with content.

What is missing is any serious attempt to create fully three dimensional world spaces, including bridges, archways, towers and other highly interconnected topologies. This is in a big part due to one of the key constraints of dynamic level generation which is to ensure complete connectivity between all parts of the map. Even in a two dimensional map, this can be a complex problem. In 3d, any game which simulates gravity must ensure that units that fall into lower parts have a way of accessing back to the higher parts unless deliberate disconnections form a part of the game play. In addition, constraints on slope steepness, minimum unit size and so on can complicate level connectedness considerably.

Tribal Trouble side steps these issues somewhat by generating enough maps and checking their connectedness, and then using a statistical proof to show that a sufficiently large percentage of maps will be playable. Other random level generators may use a verification method, which discards and restarts any maps that may prove unsuitable for play. Sufficient tests have to be made for degenerate cases, and fall backs to avoid infinite loops or sufficiently long delays in level generation, which even with a two dimensional maps can be a considerable complication, often due to the computational cost of random number generation. Dwarf Fortress maps, even prior to the introduction of a 3rd dimension, could take 15 minutes or more to run on a modern processor.

2. Design of level content

If the problems of random map generation may seem insoluble, then this does not stop them from being used. What frequently happens is that level content, particularly height fields, is generated in the level design tool, as opposed to the final game engine. The level designer is then responsible for verifying the correctness of the level or modifying it to ensure it is correct. Procedural content generation then becomes a mechanism for minimising the cost of content creation.

Darwinia[11] by Introversion Software is a great example of using procedurally generated levels supplemented with hand editing to leverage a small developer base. The Darwinia levels were original built using a procedural 3D height field generator as it would have been unfeasible to places all the map vertexes manually. Introversion is using the same technique for their next in-development game Subversion: Chris Delay releases frequent development updates on the in house blog that are well worth reading[12]. The Subversion city generator looks robust enough to be used as a dynamic random level generator at this stage but without knowing the final game play it is impossible to be sure.

The MegaTexture[13] technology that John Carmack developed initially for Enemy Territory: Quake Wars lends itself well to importing a procedurally generated bitmap for a map level. The technology itself uses Wikipedean 'procedural generation' to convert this bitmap into a three dimensional playing space, but the original bit map could be created using standard PCG height field algorithms and 2 dimensional PCG techniques.

At a lower level, there are already well accepted techniques for infilling level detail using 'procedural generation' and procedural content generation techniques. Chief amongst these is SpeedTree[14], a middle ware product that enables level designers not to have to worry about the placement of every in-game tree and shrub. SpeedTree is used in Oblivion to great effect from what was originally a piece of software developed to enhance computer game golf courses.

Similar techniques can be used to automate the placement of textures and decals so level designers don't have to worry about texture alignment and can instead focus on higher level design elements. This is in addition to procedural textures, which are again an example of 'procedural generation' that may incorporate random noise to lessen visible artefacts.

3. Dynamic world generation

This is one of the earliest procedural content generation techniques, where the in-game map exceeds the ability of the computer to store it either in memory or on-disk (depending on performance requirements). While storage has grown massively since the days of Elite[15], it is still not unlimited and the same techniques can be put to great use, for instance, displaying all the objects in an typical size asteroid belt or planetary ring.

The technique is to use a seed number (42 is a popular choice[16]) which remains constant, and then grow the playing field iteratively using this seed number permutated by pseudo-random number techniques. Usually growth is from

[11] http://www.darwinia.co.uk/
[12] http://www.introversion.co.uk/blog/
[13] http://en.wikipedia.org/wiki/MegaTexture
[14] http://www.speedtree.com/
[15] http://en.wikipedia.org/wiki/Elite_(computer_game)
[16] http://en.wikipedia.org/wiki/42_(number)

a top-down subdivision, which lends itself well to fractal generation techniques, as well as matching typical level of detail (LOD) implementations. The map is never held in memory except as a temporary structure to display, and is discarded as soon as components of it are no longer required: nothing procedurally generated is ever written back out to disk.

The best example of this in development is Infinity: The Quest for Earth[17]. I can only urge you to watch this video[18] (wait for the second half if you are impatient) and then read the developer's blog, in order to see the power of this technique.

It does have significant drawbacks. In particular, certain structures, typically long thin ones like roads and rivers are very hard or completely impossible to implement this way because of the subdivision technique (roads are merely hard, rivers impossible because of the dependency of having to know the height of an adjacent subdivision in order to compute the river path). And currently the process is very CPU intensive, with increasing levels of detail usually depending on a high O-notation algorithm of some kind.

Changing the algorithm will also change all of the map, as the map structures themselves are highly dependent on the algorithm and therefore the software cannot be upgraded easily without resetting the game. The procedural content generated must be verified as consistent as a part of the run-time implementation: which limits what checks can be made to ensure that the content makes sense and therefore requires robust generation procedures. Finally, any change to the map must be stored as a delta. Reloading map deltas will ultimately overwhelm the strengths of the technique, so the majority of implementations will prevent, minimise or ignore any player or designer made changes to the map structure.

What instead happens is that the map lends itself well to exploration: essentially you are moving through a set of possibility spaces. The game designer is as much in the thrall of the procedural content generation algorithm as the players. He can manipulate the algorithm to try to generate different types of spaces – but the whole world depends on the algorithm, and therefore he may equally destroy something else that has been created. In fact one DOS-based game has made this exploration a central element of the game: features discovered and named by the players become a part of each further release.

4. Instancing of in-game entities

Most games hide the enemies faces. This is a natural consequence of the fact that humans naturally look at faces to distinguish individuals, and designing or video capturing a new face for each potential enemy in a game can be a long and exhausting process. Consider a game engine that is automatically capable of generating a large number of faces based on a set number of parameters. This possibility is used to great effect in various MMORPGs such as Eve Online, where it is important to distinguish players from each other. Procedural content generation would be the next logical step – where each enemies face is randomly assembled by picking a random value from the allowed range. You'd end up with good looking and ugly individuals – mostly ugly if the random face selector in Oblivion is indicative of the results.

But this is not just limited to individual faces. The Massive program used by Weta in Lord of the Rings does the same for whole orcs, elves and other races. Again, individuals are selected from a base design template that is then varied from one to the next. When will we see this technology in-game? Well if Project Offset[19] is anything to go by, soon. The first sneak peek video released showed a large number of instanced goblinoids of various shapes and sizes getting shot by a crossbow and turned to ice within real time in the game engine (The fact this video is now no longer available on their site does not bode well for this feature).

Any why stop at goblins? Borderlands has instanced guns – apparently close to 500,000 different types of them, as well as other equipment types. By combining instanced guns and enemies, you ensure a virtually unique stream of opponents for the player to encounter. This can also be used to cut down on the total art assets required for the game. If you have different in-game statistics for small, medium and large individuals, you can increase the number of potential encounter types – much like World of Warcraft does with it's cookie cutter but with different paint mobs.

5. User mediated content

You can go one step further than SpeedTree, and instance all the game vegetation, so you'll never run into the same tree twice. This particular example is useful for two reasons: firstly, it's often used in first year comp sci courses to demonstrate fractals and procedural techniques, but also because Stanford are now offering a user guided procedural

[17] http://www.infinity-universe.com/Infinity/
[18] http://www.fl-tw.com/InfinityForums/viewtopic.php?t=5556
[19] http://www.projectoffset.com/

content generation system called Dryad[20] to assist people in exploring the possibilities of virtual worlds. Dryad allows you not just to create trees, but it serves as a template for all possible trees, and guides you towards 'high-quality' trees within that space, based on other user choices.

This mechanic is very similar in concept to that used in Spore. Rather than directly exploring the procedural content options by using random selections, both Dryad and Spore provide the same procedural frameworks of a PCG to allow user generated content to be rapidly created without deep technical know how. The face composite selection mentioned earlier that MMORPGs use is a similar mechanic. These rely on the user's perception of beauty or interest to guide procedural selections.

This process is extremely powerful. Google uses the ESP Game[21] to label images for search using this kind of distributed intelligence. The concept of "Games with A Purpose"[22] is the ongoing academic project based on the ideas of Louis Van Ahn[23], who noticed that games can be a great motivating force for otherwise repetitive and uninteresting activities. The concept of user generated content when combined with PCG allows the created content to be tuned by intelligent feedback, compensating for the main weakness of PCG in that human intelligence is only indirectly in control of what is produced. The players in effect become the censors of what are appropriate and inappropriate choices (The issue of censorship and procedurally generated content vis-a-vis a procedural hot coffee is another great topic – one I've briefly touched[24] on before).

6. Dynamic systems

S.T.A.L.K.E.R.: Shadow of Chernobyl[25] provides the best modern example of the variety that a dynamic AI system can provide in a static game environment. It's A-life system procedurally creates enemies in previously vacated areas, and ensures that no two encounters, even after a reload of a saved game, will ever proceed in quite the same fashion. Oblivion promised a similar revolution with it's Radiant AI system, but the majority of the dynamism was scaled back to prevent a rapid depopulation of Cyrodiil – a problem that suited the atmosphere of S.T.A.L.K.E.R. more than a fantasy world.

To call dynamic AI a procedural content generation system may feel slightly revisionist – but the elements of unpredictability through a range of possible options fit the scope of what procedural content generation delivers. More importantly, it imparts the same benefits and problems. A dynamic AI system allows the developer to cut down on time required to develop step by step AI scripting and allows them to focus more on the high level goals. But at the same time the system must be robust enough to cope with the game-space and handle the same contingencies that plague PCG based systems. A S.T.A.L.K.E.R. developer's diary[26] highlights the issues they experienced tuning the A-life system.

Other dynamic systems in game, such as weather and music are also being explored. Brian Eno is producing procedurally generated in-game music[27] for Spore, and other games have used dynamic changes in music based on the player circumstances to great effect. Dynamic weather and a day/night cycle are starting to feature in games, such as Crysis, which again perturb the possible in-game states by providing a significant unpredictability to any encounter.

Façade[28] is an experimental game which provides another dynamic system which is procedurally controlled – that of conversation. The individual snippets of conversation are not procedurally generated, but the ordering and topic transitions are controlled; and human language is flexible and context dependent enough that listeners will often fill in additional meaning to randomly adjacent phrases, particularly when cued by previous attitudes. Facade occurs within a limited space, and an emotionally charged scene, to try to emphasize these cues, and has yet to see a wider adoption of the same processes.

Speech synthesis and natural language generation are nearing levels of sophistication to be used in computer games. Note that natural language comprehension is more difficult – the best that can be delivered at the moment are heavily context dependent deciphering, or as contextual cues. But a game is a controlled enough a domain where both of these techniques may be useful.

[20] http://dryad.stanford.edu/
[21] http://www.espgame.org/
[22] http://www.cs.cmu.edu/~biglou/ieee-gwap.pdf
[23] http://www.cs.cmu.edu/~biglou/
[24] http://roguelikedeveloper.blogspot.com/2007/01/you-find-scroll-of-genocide-titled-pol.html
[25] http://en.wikipedia.org/wiki/S.T.A.L.K.E.R.:_Shadow_of_Chernobyl
[26] http://www.oblivion-lost.net/index.php?zone=design2&lang=en&page=static&static_page=life-simulation
[27] http://www.eurogamer.net/article.php?article_id=71836
[28] http://en.wikipedia.org/wiki/Façade_(interactive_story)

7. Procedural puzzles and plot-generation

Many game plots are little more than dependency graphs, which a sequence of actions must be performed in a particular order. At the simplest level, procedural content generation can be used to change the door codes and other individual puzzle elements to prevent a user from getting the information off of a game FAQ website or other source of information. And as Raph Koster points out in a article delicately titled "You Are All Cheaters"[29], the use of these out of game information repositories are game-breaking in that they stop the game being played in the way it was intended. Similarly, puzzles can be extended by making multiple parts of the dependency graph randomly placed (e.g. moving the key that opens the door to a random accessible location).

But procedural generation can do more than just make puzzles harder to solve. It can potentially give an infinite number of ways of solving a puzzle. This is an important point. In one way, failure within a game can be construed as consuming a certain amount of game content (a point I'll discuss in more detail in another article). Procedural generation of content ensures that there is an infinite amount of content to be consumed. In most games, "if at first you don't succeed, you fail", as GlaDoS puts it succinctly. But games like Rogue, which have the harshest penalty for failure – restarting the game – the fact that no two games play the same way makes the real cost for failure virtually zero. If you die in Rogue, you don't have to repeat the identical game elements in the way that you would if you fell off of a platform near the end of a level in Mario, or just before a check point in Halo.

This is even more important for games that have a fixed narrative. Narrative is consumed in a game much the same way as content. Having different multiple endings is a way of extending the narrative so that it is worthwhile playing at least twice. But why have multiple endings? Why not have multiple beginnings? Why not make every single plot point significantly different from game to game? Procedural generation of plots is a way of extending the life-time of a game significantly beyond that of a single play through. It sounds like this complexity of plot branching[30] features in Mass Effect, but in a non-procedural manner.

By making plots dynamic, choices will take on real significance. Suddenly no individual (except the player) is indispensable. Conversation trees will take on real meaning, where it is truly possible to offend or impress anyone. The threat of death will hang over any NPC – and the necessity of resetting the game in certain non-failure states will hopefully be lessened (although the negative reviews[31] of Fire Emblem suggest otherwise. Maybe it is a natural human need to not want to lose anything of value you have invested time in). The dynamic AI director in Left4Dead[32] suggests that pacing, another important game element, is amenable to procedural techniques.

Every so often, someone rediscovers procedural content generation and for a moment sees it as the solution to their game development worries. The latest last crisis was in the increasing cost in delivering triple A games on console platforms, and procedural generation was going to magic this cost away by making content free. Googling for 'procedural content generation' is like looking through a slightly forlorn graveyard - it's sunny and the textured head stones look beautiful in the evening light, but ultimately it's not as full an after-life as you anticipated.

Why has PCG failed to deliver on the promised hype?

Firstly, procedural content generation algorithms are hard. Not hard as in you need to get a smarter programmer. Hard as in the travelling salesman ate my NP-complete halting state hard.

You are no longer looking at simple test cases in resolving issues with PCG. The problem domain becomes a complex web of interacting cases, so that you instead need to start coding automated players to explore it. And these automated players, because they are powered with less than human intelligence are good at finding crashes and infinite loops, but less than perfect at spotting anomalous events, such as water running up hill. And there are never any guarantees in this exploration that you cover the whole problem domain, because this domain is infinite in size. You are left evaluating statistically whether you have enough coverage before releasing the game.

Consider that dynamic AI is a sub-set of PCG, and the promises of AI are consistently five years away on delivering, if not longer. But dynamic AI is starting to become an acceptable part of game development, and even if crippled, starting to feature more in major game titles such as Oblivion.

But I have suggested that the pieces are falling into place, albeit slowly. Even random plot generation, although handled poorly in S.T.A.L.K.E.R. (see GearHead[33] for a good example) is moving in the right direction. Outside of

[29] http://www.raphkoster.com/2007/12/28/you-are-all-cheaters/
[30] http://feeds.feedburner.com/~r/gamesetwatch/~3/216815607/opinion_the_state_of_character.php
[31] http://www.destructoid.com/destructoid-review-fire-emblem-radiant-dawn-56004.phtml
[32] http://www.l4d.com/
[33] http://gearhead.roguelikedevelopment.org/

dynamic world generation, I don't believe anything I have shown you is a compelling argument for procedural content generation - as compared to the majority modern game titles. I'm not surprised if you disagree with my assessment that the level designer is under any kind of threat. But you're wrong.

Let me suggest where procedural content generation will move in the next five years and under what guises.

1. Traditional level design tools will adopt more and more procedural content generation functions. You will be able to paint a section of landscape and farm buildings, a village or town will pop up into existence. Similarly, more game engines will support dynamic lighting and weather, requiring less pre-rendering components.

2. Triple A titles will incorporate more PCG elements in controlled conditions (as will MMORPGs). You'll find much greater use of instancing to enhance visual appeal. Games that feature a central PCG mechanic will do much better in the marketplace – whereas Hellgate: London mostly missed the mark, Left4Dead and Borderlands will be more successful.

3. Spore will be released, possibly delayed again, and be a resounding success – cannibalizing the game playing audience much as World of Warcraft has done for PC games, but on a multi-platform basis. This will put EA 3-5 years ahead of the market, with only Sony, with LittleBigPlanet, being able to compete (Will Wright is arguably one of the world's best game designers, and he has ninjas working for him). As a result, a round of inferior Spore knock-offs will appear the following year and put back the cause of PCG by a year or two (Much like the current round of MMORPG cancellations).

4. PCG will continue to eat away at the bottom end, with various independent developers coming up with better game designs using these techniques. This will have the perverse effect of making independent development harder as the best indie games take the audience away from both other indies and 2nd tier developers.

5. At some point middle ware developers will get on board with PCG. This is already happening with dynamic AI, but I'd expect to see it with 3D level design systems and more generic 'game-in-a-box' systems. What timing this occurs around depends on whether they're able to achieve it ahead of or behind the Spore release fallout.

It's this last point that I want to expand on and suggest some ways of moving the state-of-the-art forward for procedural content generation. You may wish to stop reading at this point unless you're passionate about procedural content generation, or a venture capitalist.

Firstly, I think there's a great place in the market at the moment for a fast-moving company to deliver 3D arena like first person shooter levels using a subscription model. Call it Map of the Day or similar (that particular URL is cyber squatted, but it's the point I'm trying to make).

Users sign up for a subscription based on their favourite shooter and get a single player map every day generated by the company's random map generator. It's important that it's a single player map - multi-player maps would be a lot harder to balance and are re-usable, whereas no-one is catering well for single-player episodic gaming on a frequent enough a basis. It's not critical to start with that the map designs are exceptional - good enough is fine. Obviously you want to DRM lock the content if possible, but it's not strictly necessary. Maybe partner with Valve to deliver through Steam, although you're potentially competing with them at the same time.

Secondly is the 'game-in-a-box' middle ware model. This is targeted for companies looking to take their media intellectual property and turn it into a game, without having to pay either huge development costs or leave consumers with a badly designed game. The game in the box takes the minimum character and art assets required and uses PCG techniques to turn those elements into an entire game at a fraction of the cost of normal game development. While the game itself does not have to have PCG elements, for extensive replayability you may as well include them.

The consumers benefit because they get a well-designed game as opposed to the licensed rubbish that is often churned out at the moment. The 'game-in-a-box' company also becomes a well known brand in it's own right, so purchases would also be made on strength of the franchise in addition to the media property. The media company at least gets the moral satisfaction of having licensed a well-produced product, as opposed to having to bury it in a land-fill in the New Mexico desert[34].

An unlikely prospect? Well, as you've probably guessed, one company is already doing it. Chunsoft's Fushigi no Dungeon (Mysterious Dungeon)[35] is heavily indebted to the procedural content generation techniques pioneered by Rogue, and has produced Mysterious Dungeon games under license for Pokemon, Dragon Quest and Mobile Suit Gundam as well as their own series of independently produced games.

[34] http://en.wikipedia.org/wiki/E.T._the_Extra-Terrestrial_(Atari_2600)
[35] http://en.wikipedia.org/wiki/Mysterious_Dungeon

A match made in procedural heaven.

It's January 2013. The Christmas games embargo that the Clinton administration put in place to ensure you saw friends and family over the holiday season has been lifted, and you excitedly unwrap the plastic from your lastest purchase: Dream Park[36] 2.0. You've already heard about the great mods created by international players working on behalf of Gamers Now United (Stallman's reorganisation of the former Open Source movement once he finally succumbed to playing World of Warcraft), and you download a pre-recommended pack from your Facebook game reputation system.

The game fires up, and you click through the menus making a few choices and letting the RNG choose the rest. Medieval. Clockpunk[37] instead of magic. Moderate size continent. You see a few of the random choices flash in front of you: the game has generated four civilisations, distinguished by Ionic, Doric, Corithinian and Tuscan architecture, and the first mod kicks in. FastGen reduces the elabourate character generation options down to the choice of three key phrases: you choose Rugged, Handsome and Prankster on a whim, and end up with a boyishly good looking peasant standing at the edge of a mud-hovelled village amidst a crowd of on-lookers.

The king of the Ionic civilisaton (Hrepathous) is visiting the local barony and come down to the village to demonstrate his gift of healing and attend to the villagers' ailments. As you grow bored of the Baron's speeches and begin to look elsewhere, your lover whispers 'the pigsty' in your ear and with a shy smile slips out of your grasp and towards the romantic rendevous. You follow at a short distance, but come up abruptly against the squat figure of her father.

'Where are you going?' he asks threatening. You stammer something and are forced to divert your course to around the back of the building the king sits in.

In the shadows, stand two assassins, in Doric dress, cradling weapons you have never seen before. They've cut a hole in the adobe and are preparing to slip through.

One sees you, and raises his weapon. Just then your lover runs out, between you and your attacker. She is cut down in a hail of projectiles, and your heart beats and time seems to slow.

Do you run or fight? Your decision is made for you, as the screams of the villagers signal that more armed men have appeared in the fields around you. You run.

The hectic pursuit, between huts and over fences, watching your fellow villagers gunned down and slaughtered, comes to a climax. Trapped in a dead end, amidst the rutting pigs where you had intended an altogether different purpose, you are caught by your pursuers. One, cruel looking and with a milky blindness in one eye, and a clockwork hand, steps forward. He raises his mechanical arm and laughs at you. It is the last thing you see.

The You Only Live Once[38] mod kicks in, working in conjunction with the AI plot director. Instead of restarting from an earlier checkpoint, you awaken again, this time as a Corinthian merchant in a larger town. Rumours of war between the civilisations are spreading...

There's two kinds of hard – I'll call them depth and complexity.

Depth is similar to what I think Dwarf Fortress does for it's random world generation – it requires that you come up with e.g. a list of every possible type of mineral, multiple 'layers' of level generation such as height, moisture, temperature and combine these all. The building blocks however are relatively simple – some kind of fractal subdivision, working out which combination of height, vegetation, temperature etc result in plains, swamps and so on.

Complexity includes problems however that there is no possible solution for, or the solution cost rapidly increases out of bounds. Things like the Travelling Salesman problem[39], placing rooms and corridors in some kind of ordering, etc. Placing rivers falls in this scheme – you have to keep trying and abort if the river doesn't reach the sea in most naive implementation of rivers for instance.

To compare the two, most naive AI routines are complex rather than deep. For instance, in chess, it is complex to check for every possible move as the combination of moves grows so quickly, so deep solutions such as having a 'book' of moves are required.

[36] http://en.wikipedia.org/wiki/Dream_Park
[37] http://davinciautomata.wordpress.com/2007/03/03/introducing-clockpunk/
[38] http://www.zincland.com/7drl/liveonce/
[39] http://en.wikipedia.org/wiki/Travelling_salesman_problem

Dwarf Fortress achieves the depth problem probably better than anyone else. But I can't think of anyone coming up with complex solutions for procedural content generation. For instance, in AI, the 'complex' solutions are always 5 years away, as I mentioned, but they seem to be getting closer, and AI becoming a little smarter.

Listening to Tarn Adams' recent interview with Geek Nights[40], he's anticipating that plot like game behaviour will emerge from the lower level rules in Dwarf Fortress. I suspect this will, because he has enough depth in the game to support it. But I also suspect that very few other people, particularly commercial developers are going to want to spend the time coming up with as deep games. (S.T.A.L.K.E.R. is on the outer edges of 'deep' game behaviour, and it took six years to develop). So the 'short cut' is to come up with a procedural content generation system for game plots / game narrative to try to get replayability to another 'level'.

Most responses to earlier parts of this article have focused on procedural content generation being equivalent to randomly generating game levels or game worlds. I agree that this is an important area to focus on, and something that we'll see increasingly used in commercial games. But I also would argue that these techniques are mostly 'solved' and are therefore 'uninteresting'. In particular, most games which procedurally generate '3D' maps are really just generating a 2d surface e.g. a height field, and those that are 3D are mostly done through fractal subdivision, which while an interesting field of study, has been used extensively.

The one area that random map generation is missing is complex 3D topology generation: and no game is currently doing this. Dwarf Fortress doesn't generate the fortresses for you – it still relies on human intelligence to go through the process of placing tunnels and building bridges and towers. When a game is capable of procedurally generating maps of the complexity of Half-Life 2, S.T.A.L.K.E.R., Minerva or any other modern FPS, then PCG for random map generation will be completely 'solved'.

And the reason that this hasn't been done is that complex 3D topology is a complex problem as opposed to a deep problem. 3D space with gravity is immediately an ordered space, and therefore requires that map connectivity be checked as a part of the generation process. Since connectivity cannot be guaranteed, the map generation may have to throw away randomly generated maps and you immediately move into the bounds of the halting problem[41] – you can no longer prove the PCG algorithm will ever complete, for whichever algorithm you choose. (To clarify: the issue is not proving an individual map is disconnected, it's ensuring that you don't generate an infinite number of disconnected maps in succession).

In addition, complex 3D topology highlights some significant human perceptual issues – particularly the fact that we are biased against looking up. I suspect that any PCG which generates complex 3D topology will have to include an expert system that constrains the types of maps created. Constraints will include techniques to encourage the player to look up in the instance where there is a vertical element in the map, ensuring that gaps in the map that the player has to traverse are jumpable, tunnels are at least the player unit size and so on.

The expert system may act as a 'generator' rather than a 'constrainer' – these are analogous to 'wall-adder' vs. 'passage carvers' in maze generation[42], in the sense that the 'generator' may add features to the map that are consistent of a set of rules such as lily pads on a lake surface that the player can jump across.

It may well be that it is not ever possible to create a complex 3D topology procedural content generator that is good enough to be acceptable. I believe it will require extensive use of user mediated content techniques to get right, particularly for multi-player maps. Tools such as heat maps, as demonstrated in Valve's Half-Life: Episode 2[43] and Bungie's Halo 3, are already used for high-level analysis, and could be adapted for analysis with AI techniques such as neural networks and genetic algorithms. Similar AI techniques could potentially analyze every 3D map ever made and try to come up with a rule set of common features. These are not straightforward suggestions.

If we are stuck with hand made 3D maps, this only excludes 2 out of the 7 procedural content generation techniques I outlined previously. Again, S.T.A.L.K.E.R. is a great example of this – particularly in it's use of dynamic AI to ensure that the game-play within the maps keeps randomly changing. Even in Halo, where the AI is completely deterministic (after earlier experiments with random AI behaviour), there is still enough randomness from the player actions to ensure that the AI behaves differently every time encountered.

But I believe the real strength of procedural content generation will be seen in procedural generation of plot and narrative content.

---

[40] http://www.frontrowcrew.com/?p=491
[41] http://en.wikipedia.org/wiki/Halting_problem
[42] http://www.astrolog.org/labyrnth/algrithm.htm
[43] http://www.bungie.net/online/HeatMaps.aspx

The human brain has a powerful set of tools for detecting and matching useful patterns in otherwise chaotic data. This set of tools can be conditioned to override normal behaviour patterns in the presence of sufficiently powerful and randomly provided incentives: so-called operant conditioning. You will have probably seen the term 'Skinner box'[44] associated with certain types of game-play: in particular with MMORPGs and slot machines, where people become addicted to randomly provided rewards for repetitive activity.

But the same set of pattern matching tools are useful for the designer of procedural content generation systems. The semi-randomised output of PCG systems will be picked over by the player's brain, and the underlying systems hopefully understood and enjoyed. This reinforcement of play through randomisation is hopefully a more 'nutritious' game environment that simply statistical delivery of reward and one I would expect Johnathan Blow to commend rather than criticise[45].

The greatest challenge of procedural content generation will be to augment or replace human intelligence in the creation of meaningful narratives in computer games. But even in a single player game, there is a powerful creative intelligence involved in the decision process, and that is the player themselves. A procedural content generation system may be able to leverage this human intelligence in creating meaning and narrative: in part through relying in part on serendipity, happenstance and superstition. In juxtaposing two disparate game elements, the human mind will attempt to find a meaningful relationship between them, and provided that the game does not mix every game element into a brown stochastic blend, the player will build up a narrative of moderate complexity from very simple components. Such a narrative may be no more complicated than 'first I did this, then I did this, and then I did this', but it will have meaning.

It is possible to go too far in procedural content generation techniques: too far being when no two games have any elements in common. This may still be an interesting game – Minesweeper being a good example – but a less interesting movie[46]. Even Minesweeper has a beginning, a middle, some tough choices where a guess might be required, and an end. Most games will have more complex choices that could be formed into a player-centric narrative. Angband for instance encourages the player to stay within a certain distance of the surface until certain resistances are acquired – encountering monsters deeper in the dungeon without these resistances would likely result in game over. A more sophisticated example of these player-centric narratives is the 'Day in the Life' posts that many games encourage – these can either be mundane or dramatic retelling of in-game events from the perspective of the player controlled avatar.

These player-centric narratives, while interesting, are probably not what you envisioned when you hear 'procedural content generation of narrative'. And while the emergent narratives of Dwarf Fortress are a sophisticated example of what is possible with a sufficiently deep game system, as mentioned, not every game developer has the time, resources and fortune to develop this type of game.

What would an explicit, as opposed to emergent, PCG narrative system look like? One way of implementing this is little more than a sophisticated puzzle generation system – essentially a string of escalating Fed Ex quests, and randomly selected Blankety Blank[47] sentences. This is still developer expensive, and at the same time not particularly rewarding for the player, as the systems underlying the puzzle generation is no more sophisticated than the statistical generation of reward probabilities in Skinner Boxes.

By increasing the complexity of the systems underlying the puzzle generation, there is a greater sense of involvement from the player, who must decode these underlying systems in order to maximise their rewards from successful puzzle solving (or questing, or whatever game equivalent process is). And the easiest way to increase this complexity is to have a behind the scenes meta-game, that the player has no direct involvement in.

Consider the plight of an individual person living in the world of a game of Civilisation IV[48]. They may be pivotal to the outcome of a single battle, if they are in the position of being a spy, or help advance the civilisation, if they are a Great Person. Each civilisation is run by a leader, whose particular traits inspire the civilisation to work better in certain areas. But no individual (except the 'hand of God' player who is playing the civilisation behind the scenes) is responsible for the actual decisions taken from turn to turn, how the civilisation advances and how the game of Civilisation IV is played.

Now, replace all the human players playing a game of Civilisation IV with computer players. And instead put the human players in the roles of individuals living in the game world. Civilisation IV has suddenly become the meta-game. Individuals will perceive the actions of the AIs controlling the game only through their indirect effects:

[44] http://en.wikipedia.org/wiki/Skinner_box
[45] http://www.braid-game.com/news/?p=129
[46] http://www.youtube.com/watch?v=LHY8NKj3RKs
[47] http://en.wikipedia.org/wiki/Blankety_Blank
[48] http://en.wikipedia.org/wiki/Civilization_IV

cities will be built, technologies gained, enemies routed and civilisations fall. But no player in the game will directly be able to control these effects – their influence will instead be felt only as warriors or spies, or great people or perhaps civilisation leaders.

The actions of the AIs in controlling the meta-game should appear to the individuals in the game to be a consistent narrative. An AI's decision to amass armies on the border to invade a neighbour will have profound in-game consequences. Whether the invasion is successful or not could be decided by one individual: a spy whose vital information enables the encirclement of enemy forces, a leader who can inspire his troops to break through the enclosure, a peasant who correctly shoed the horse of a general so he wasn't thrown from it's back. But the fact that the attack occurred, and the consequences of it, should form a consistent narrative from the player's point of view because it follows a consistent set of meta-game rules, which the player may be able to decode over the course of the game. With sufficient understanding of these rules, the player may be able to predict which side to take, when to rally his troops, when to flee the scene.

And Civilisation and other 4X games[49] are well understood problems, with sophisticated AI, that can provide a narrative backdrop to a lower level game involving player control at an individual or a squad level. There are already examples of blended RTS/FPS games such as Natural Selection[50] and Savage: The Battle for Newerth[51] that have proven successful using this kind of idea. The weakness of these games, and the reason that the Valve elected not to have a commander class in Team Fortress 2, is that the success of the game depends heavily on the competence of the battlefield commander player. Replacing the battlefield commander with an AI levels the playing field in this regard.

And hopefully makes the story behind the play.

[49] http://en.wikipedia.org/wiki/4X_game
[50] http://en.wikipedia.org/wiki/Natural_selection
[51] http://en.wikipedia.org/wiki/Savage:_The_Battle_for_Newerth