Techniques To Improve **Any** Generativity:

So You Have 18,446,744,073,709,551,616 Planets…

You can generate many… things. They are all mathematically unique, **but they aren't perceived as unique** You may have the **10,000 bowls of oatmeal problem**: all your content is perceived as unmemorable (despite being *technically* different)

Is this a problem?

(Did you say the Really Big Number? In your marketing? Because it got you a lot of media coverage? Then its a problem. Never listen to the Really Big Number.)

Sometimes content can be oatmealish, and be just fine, but sometimes it needs to have character. Know your oatmealishness levels:

- **Background/In-fill (just don't be empty!)**
- **Perceptual differentiation**
- **Perceptual uniqueness**
- **Characterful (test: would you write fanfic for this generated item?** Have a few pieces of characterful content, sprinkled with perceptually unique content, and maybe some infill in the background.

Ownership: MSG For PCG

- Allow users to name content
- Showing off content with their name attached, to a large audience (the "victorian explorers club" model, DIY anti colonialist reading here)
- Let players take credit for your generativity: let them become **creators, curators, retellers**

Data Structures

- A/B test generators
- Release new generative content safely
- Create editors and run user-made generators safely (Tracery/Cheap bots done quick)
- Visualize your generators to spot problems

@galaxykate's

ENCYCLOPEDIA OF GENERATIVITY

You can build a generator! Yes you! Even without code, or with a little code, or with sophisticated math-filled code! Lots of generative content uses extremely sophisticated and brilliant AI and fails anyway. Some of the best generative content is **simple!** The hardest part of procedural content is **DESIGN**: what you make and how you use it.

❤️ **YOU CAN DO THIS!** ❤️

Thinking About Generators

Generators can be BIG or SMALL, SIMPLE or COMPLEX, in the SPOTLIGHT, or in the BACKGROUND

A generator might let a small team (or a single person) make lots more content than they could make alone, or it might help you make weird and surprising content, or it might help you make impossible-for-humans-to-make content.

You might make generators that make textures for games, textures for fabric, impossible spaces in VR, humorous stories, serious stories, cocktail recipes, music, light shows, poetry, 3D printed teapots, a chatbot on Twitter, dialog, soap opera stories, game rules, or SO MUCH MORE. Generate lots of small stuff before you generate something big, there is much you will learn!

Building An Artist-In-Box

Find an expert in the domain to talk to, or read up on the domain space. Try to **understand** how an expert would build one of these. When do they use intuition or formal procedure? What steps do they take? Even fields like cocktail recipes and renaissance architecture have their own strange rules!

You're making an artist-in-a-box, so you need to teach the algorithm everything a good artist would know.

- Write all the good things this domain needs
- Write all the bad things that can go wrong
    - put a star next to the ones that *break*) it, not just bad (an ugly teapot vs one with a hole in the bottom, a boring game vs an unwinnable one)
- Write how you know when one is good (a "heuristic")

Possibility Space And Expressive Range

Your generator will make many things, some good and some bad. The total number of things it can generate is the **possibility space**, like the number of six-letter words is "191102976". The **expressive range** is the *kind* of stuff in your generative space, like how most of those six-letter words are nonsense, but some are gerunds and many are nouns.

For More Info About Constraints

- Mike Cook's Danesh project
- So you Want to build a Generator - Kate Compton
- Expressive Range: Evaluating and Comparing Generative Systems - Gillian Smith

Subtractive methods

Generate and test: If you can write an algorithm to judge "quality"…generate until you get some good content.

Throwaway vs ranking/prioritization: test for brokenness, but rank by quality, its better to return crappy content than wait forever for "good enough" Beware of false functions (there is no "fun equation")

Seeded random number generation

- Seeded random numbers: Same seed? Same generation! (if nothing is framerate or input dependent!)
- Whitelist a catalog of known good content
- It's faster to verify questionable content than to build a testing function

Computationally exploring the possibility space (aka, "search"): Brute force search, Hill-climbing, Genetic algorithms (mostly works with parametric generation)

Constraint solving: You can describe a possibility space and constraints, just find the valid parameters.

- IK-solving
- Answer set solving (Potassco Clingo)
- Brute force!
- DO. NOT. WRITE. YOUR. OWN.
- if a member of you team starts doing this, STOP THEM (brute force is ok. just pay attention to exponential growth)

If you find you want to do constraint solving, here are some helpful resources about a good tool. (https://eis-blog.soe.ucsc.edu/2011/10/map-generation-speedrun/) (https://sourceforge.net/p/potassco/mailman/message/31086395/) (https://thelazydev.net/blog/post/the-basics-of-answer-set-programming)

Fractals And More

Fractals and other math can create impossible mathematical spaces for games and VR. It's not for the faint-of-heart or scared-of-math, but can lead to spectacular results. Warning, if often does *not* work well with traditional gameplay! http://marctenbosch.com/news/category/miegakure/ http://elevr.com/hypernom/ (github and arxiv paper)

Simulations

Often simulations are used to build generative content, like the simulated particles behind Spore's texturing, or the complex simulation behind Dwarf Fortress and Bad News.

- Cellular automata
- Agent-based simulations (The Sims and Bad News)
- Physics simulation
- Boids and braitenberg vehicles

Here's a paper I wrote about doing dance animation with steering. https://dl.dropboxusercontent.com/u/3116524/teuhana-iccc2015.pdf

Conferences And Jams:
ICCC International Conference on Computational Creativity, Foundations of Digital Games, Nanogenmo, ProcJam
Online Resources
http://galaxykate0.tumblr.com/post/139774965871/so-you-want-to-build-a-generator | http://cheapbotsdonequick.com/ http://tracery.io/ http://natureofcode.com/ http://www.contextfreeart.org/ http://ncase.me/simulating/ http://pcgbook.com/

Methods

Additive And Subtractive

Additive methods **build up** your possibility space and **expand** your expressive range. Subtractive methods **prune away** bad parts of the possibility space, and **search** for good artifacts in the space.

Here are some kinds of additive methods * **Tiles** * **Grammars** * **Distributions** * **Parametric** * **Interpretive** * **Simulations**. There are probably more (like deep learning and fractals!)

Distribution

You can put stuff down randomly, but it looks unreal and awkward. “Real” distributions are hierarchical and clustered, but also maintain spacing. Three properties help distributions look good:

- **Barnacling** surrounds large objects with some smaller objects and many tinier objects
- **Footing** adds texture or difference where two things intersect
- **Greebling** covers large flat areas in interesting (but meaningless shapes)

Algorithms to do distribution Fast Object Distribution, Andrew Willmott on Spore Voronoi pattern with easing: https://www.wired.com/2012/04/whats-voronoi-dr-evil-mad-scientist/

Parametric

Imagine you have N different sliders. Each of which controls some aspect of generative content, like the width and color of leaves or bushiness of a plant. Each slider can go from 0 to 1. So each plant in your possibility space can be represented as the position of these sliders [0.0, 0.94, 0.12, ... 0.45].

This has some neat side effects

- saving the vector saves the content
- modellable as points in an N-dimensional cube
- **nearby** points in the space make **similar** content
- any position is a valid artifact!
- You can do genetic algorithms or user directed walks through the space
- or “regionize" the space

For more info about user-controlled parametric design

“Petalz: Search-based Procedural Content Generation for the Casual Gamer”

Interpretive

Often you want to turn a *simple* input into a complex one. Interpretive methods interpret or complexify or transform content. This is especially good for user-provided content like mouse gestures or shadow-, body- or hand-tracking!

Perlin and simplex noise Perlin noise (and its later faster cousin, Simplex noise) map input values to a continuous ever-changing output value. Note that Simplex noise itself is patented, so use the similar OpenSimple noise to avoid issues.

Voronoi and Delaunay and other geometry/triangulation algorithms You have some points or other geometry and you want to make a mesh around them. *Handy algorithms:* Chew's second algorithm, Quad edges, Worley Noise

Constructive solid geometry, extrusion Can you take a path and place geometry or textures along it? This is what photoshop brushes do. But Spore also used particle trails to create the patterns on planets and creatures!

Metaballs can take a set of points, and build a complex smooth mesh around them. This expensive but impressive technique was used in Spore and Oculus Medium.

Notable Examples Of Interpretive Generation

- The Treachery of Sanctuary - Chris Milk (https://creators.vice.com/en_us/article/how-it-works-chris-milks-ithe-treachery-of-sanctuaryi))
- Text Rain - Camille Utterback (1999)
- Grasshopper modelling software
- Tiltbrush: extruding interesting geometry along curves
- Oculus Medium: voxels and metaballs, (probably?)
- I/O Brush by the Tangible Media Group at MIT, stamps sampled photos and GIFs along curves (http://tangible.media.mit.edu/project/io-brush/))
- Illustrator and Photoshop brushes, here's a canonical history of the evolution of digital paintbrushes (http://excelsior.biosci.ohio-state.edu/~carlson/history/PDFs/14_paint.pdf)
- LuminAI project (Georgia Tech)
- Nervous System: parametric and interpretive generation for 3D printed fashion and design. (http://n-e-r-v-o-u-s.com/blog/)
- The Painting Fool, parsing images into regions, and coming up with brush strokes and styles to render them (http://www.thepaintingfool.com/)
- SSX's procedural snowboard track layout (http://documentslide.com/documents/ssx-procedural-slides.html) (http://www.gdcvault.com/play/1015547/Asking-the-Impossible-on-SSX))

I made cut-and-play cards to help you design your own interpretive generators!
(http://www.galaxykate.com/arttoys/arttoy-cards.pdf)