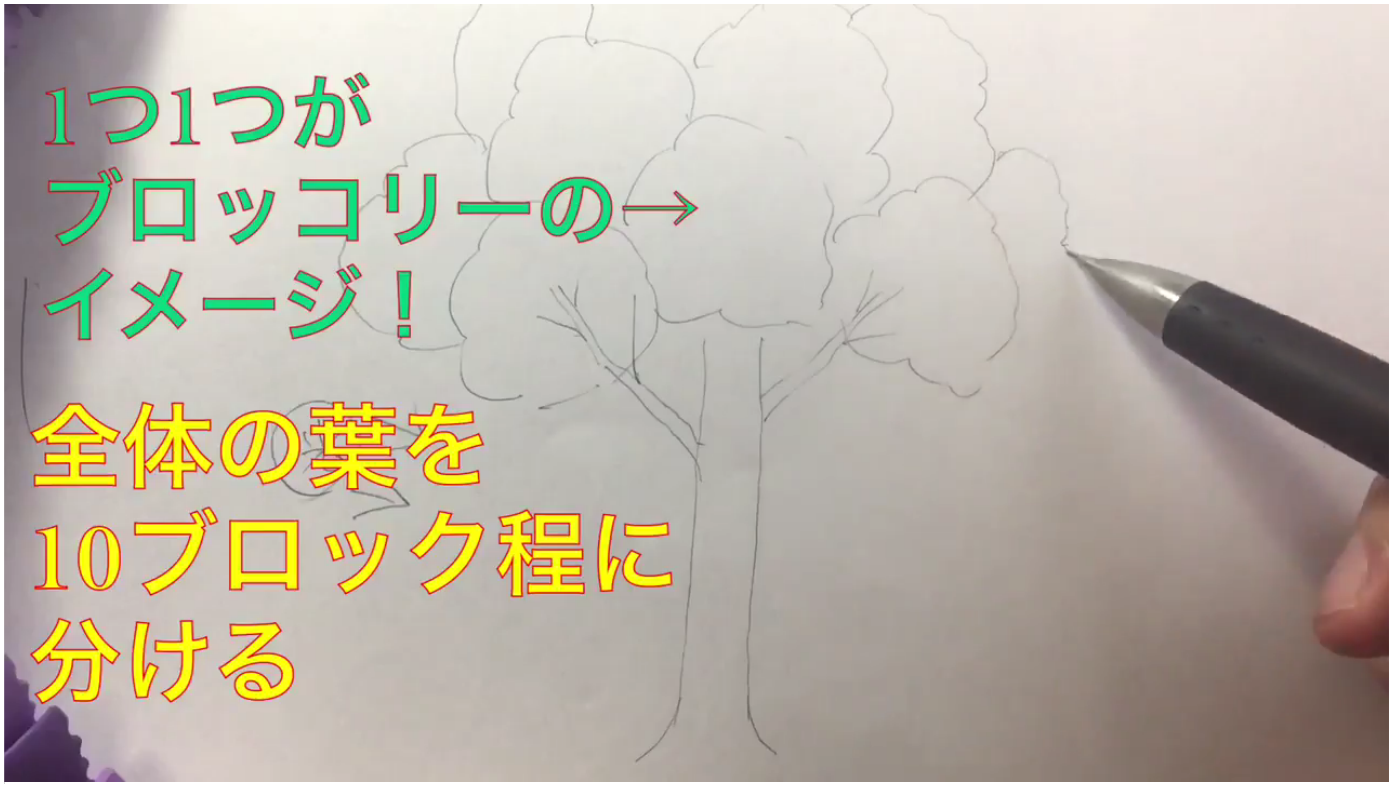


吉村拓也
@hanari0716

0:08 / 2:20

That's great but we'll leave that for the recursive procgen Doodle Insights. For now, **we're going to be considering linear-focused steps.**

Here's a good example:

```
define trunk -> define branches -> draw dark leaves -> draw light leaves
 -> lighten up the border of the tree on one side
```

This the best tree generation I've ever done, and I've done a bunch.

And here is **the inspiration for it:**

As you can see, I did **simplify** the steps by using tricks I could use on Pico-8. The thing you can find back the most in the doodle is lighting. In this **how-to-draw-trees** video, you can see how the trunk is lightened on one side and how the leaves are divided in a lot of balls of leaves, lightened **separately**. Even though I have very few balls of leaves in comparison, lighting them differently gives the tree a ton of charm. The lighting of just one side of the tree simulates the lighting effect from the video on the trunk and also adds some more lighting to the leaves. *After the generation is done, the leaves are animated with **procedural dithering** to make it look more lively.*

My point here is that **step-by-step videos are the best thing to try and translate into linear procgen.** It is sincerely **a ton of fun to do** and the result is always top notch if you do it well. While you could just follow the steps presented to you, it can also be more clever to **break them down and rearrange them** in a way that makes more sense with your tools, like I did with this tree generator.

It doesn't have to be drawing videos either, I would be seriously be interested into **a similar process applied on 3D sculptures or IKEA furniture manuals.** *(I'm not sponsored by IKEA in any way)*

But let's get more original! As written somewhere above, linear procgen is super underrated. Not only does it let you make **really cool-looking content based on other people's stuff**, it's also a great **solution to procgen problems**! If you're programmer, you encounter problems all the time and you know that in every cases there's a few solutions ranging from **efficient and clever** to **idiotic and terrible**. But surely you also know that some solutions are **clever but terrible** and some others are **efficient but idiotic**. **Linear procgen makes for great efficient-but-idiotic solutions.**

All you need to do is **break down your problem** in smaller parts and then get onto **solving each part one after the other**. Idiotic. But efficient.

Let's say I want to do **a cave generation algorithm**, making caves like the ones you would find in **Terraria** or **Starbound**, or **Minecraft** if it was in 2D. **Plenty of recursive-procgen approaches exist** and you can read all about them on the internet. The problem I have with them is that they are **complex** and, if you're not too interested in them, **really boring**. **Here comes the**

linear approach!

What do we want in our cave generation?

- We want **an uneven surface**, with at least hills.
- We want **an interesting distibution of dirt and rock.**
- We want **caves**, preferably crossing one-another.
- We want **water and lava.**
- We want **some greenery.**
- We want **ores.**

We have our feature wishlist, **let's cross them off, one by one.**

For the uneven surface, we're using a spline going through points of random height. **The distribution of dirt and rock** is done by drawing that spline lower and lower until it's totally under the screen, each time with a random color (dirt brown or rock light/dark grey) depending on the current height.

The caves are also splines with points of randomized heights, set at different levels. The black splines are also drawn again and again with random height offsets to make them wider.

The water and lava are both added using an algorithm looking for suitable spaces that can hold them and drawing them there.

The greenery is cellular automata. **The ores** are another cellular automata.

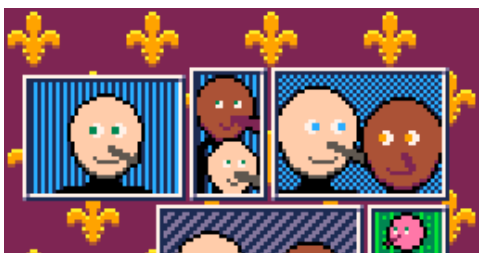
And like that, **here we have our cave generation!**

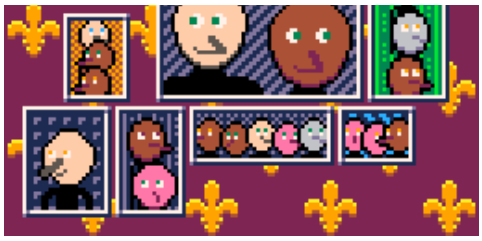
*Again, after the generation is done, I'm using **cellular automata** on the liquids and **procedural dithering** on the ores to inject some life into the result.*

This will have taken a few hours to do but it was really easy all along with barely any problem at all (apart maybe from the liquid placement).

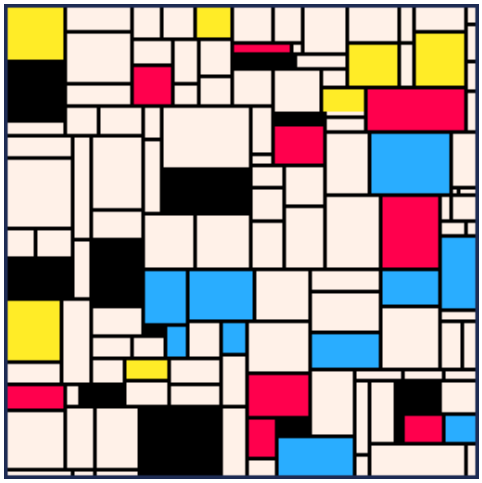
Here's yet another positive side of this type of generation: **each of your steps can be isolated and potentially reused in a different generator!**

Here is **my latest Pico-8 Doodle:**



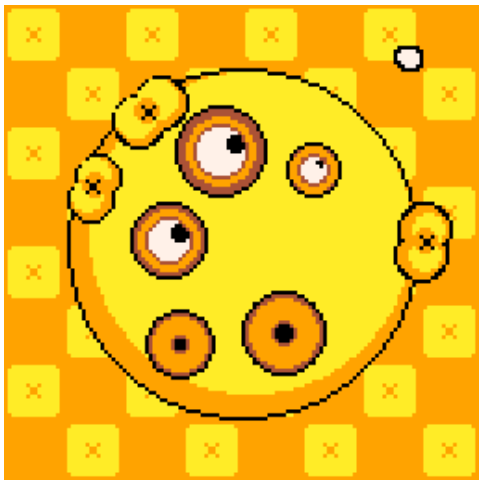


Ok it looks a bit creepy, but that first part with the moving rectangle boundaries was very interesting to do (it's done with cellular automata) and **I can reuse it for this:**



This is a procgen version of Mondrian's neoplaticist paintings. (I looked it up) All I needed to do was **extract** the moving-rectangle-boundaries step into a new cart, let it run for a second, stop it and color a few random rectangles in black, red, blue or yellow.

Finally, **we can part away from procgen a bit and apply the same ideas to a character creation module for example.** Here, each tool given to the player is one layer of your step-by-step generation. If you think about it, when you create a sim in a Sims game, choosing options in each menus and sub-menus, you're just **altering each step of the making of your final and original sim.**



While this doodle isn't nearly as complex as the making of Sims sim, **it show an exchange between procgen and player input.** The player gets a colored head and can spawn eyes, ears and mouths by pressing X which they can move around or delete by holding Z. But they can't choose which will appear. They either have to spawn a lot of parts hoping to find the one they want, or they can cope with the random and make monsters they may not have been able to imagine otherwise. This semi-recursive semi-linear design goes like this:

```
procgen head generation -> player X input -> procgen part generation
```

```
-> player Z input (move or delete generated parts)
-> player X input -> procgen part generation
-> player Z input (move or delete generated parts)
-> player X input -> ...
```

What's the catch? Well, as mentioned in the part about problem-solving, **this method tends to be pretty idiotic and will have you write a fair amount of code** which won't always look super clean. The bright side is that **the code you'll be writing won't be very complex** and you won't spend much time looking for your mistakes when things don't work, like you would with a recursive design. ***Here's a tip though***, when you code linear procgen, **you should separate your steps into functions**, so that it's easily to extract any one of them, whether it's for **testing or recycling purposes.**

Aside from that, **linear procgen is very efficient and makes for really good-looking generations.** You can also **show off each step of the generation** which people ***love*** seeing. It can be used to **translate step-by-step instructions into generators**. And it can be **translated into game design by involving the player**. And again, it is **a super underrated technique.**

I hope you enjoyed this Doodle Insights! As usual, **tell me your remarks and questions on the Patreon post**, I'd love to hear them!

These Doodle Insights are here thanks to the awesome people supporting it on Patreon! Here are their names!

Joseph White, Adam M. Smith, Ryan Malm, Matthew, Tim and Alexandra, Sasha Bilton, berkfrei, Nick Hughes, Christopher Mayfield, Jearl, Dave Hoffman, Thomas Wright, Morgan Jensen, Zach Bracken, Anne Le Clech, Flo Devaux, Emerson Smith, Cathal O'Keeffe, Dan Sanderson, Andrew Reist, vaporstack, Dzozef, Cole Smith, Jared Butowsky, Tony Sarkees and Justin TerAvest!

Writing this was really cool, as linear procgen is a technique I have a lot of fun using! Next week I'll probably write about recursive procgen which is also fun (but less imo) or maybe something else! If there's a subject you'd like me to write about, make sure to tell me!

The character creation doodle presented here can be played and downloaded there on the Pico-8 BBS and the other ones can be downloaded by 8$+ patrons on my Patreon!

Have fun with linear procgen!

TRASEVOL_DOG

Published by trasevol_dog

I make video games and graphic experiments. Everything must move. View all posts by trasevol_dog

 March 7, 2017
Doodle Insights

UP ↑