Note VI

by Chenchao Ding

Nov. 11, 2024

Procedual Generation

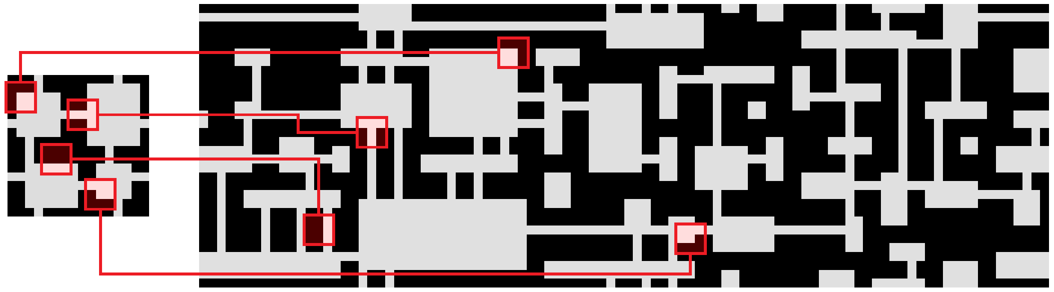
In computing, **procedural generation** (sometimes shortened as **proc-gen**) is a method of creating data algorithmically as opposed to manually, typically through a combination of human-generated content and algorithms coupled with computer-generated randomness and processing power. In computer graphics, it is commonly used to create textures and 3D models. In video games, it is used to automatically create large amounts of content in a game. Depending on the implementation, advantages of procedural generation can include smaller file sizes, larger amounts of content, and randomness for less predictable gameplay.

One example of procedural generation, here using L-systems to generate realistic looking tree models. Different models can be generated by changing both deterministic parameters and a random seed.

- human-curated content: akin to a set of constraints or rules.
- a world can be “interactively rendered/generated” instead of automatic algorithms.

A Simple but Curious Algorithm (WFC)

Among various Proc-Gen algorithms, there is a almost recent “quantum mechanics inspired” algorithm named Wave Function Collapse (WFC) which is quite minimalist:



The basic idea is:

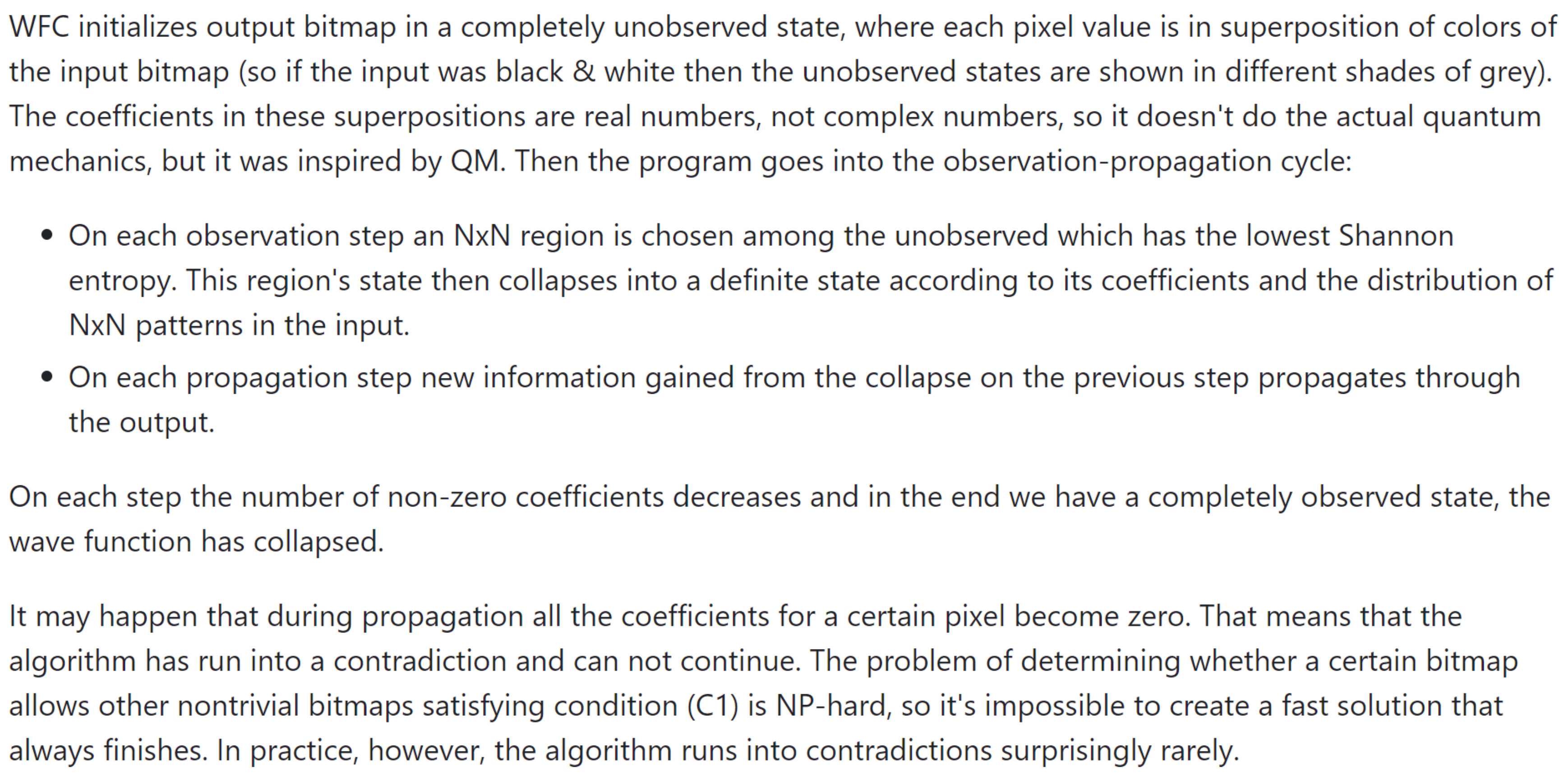
- given a very small tilemap A (e.g. 16×16).
- learn a set of constraints from A with $N \times N$ (e.g. $N = 3$) patterns.
- generate a potentially infinite map B that is “locally similar” to A.

Localy similarity:

i. B contains only those $N \times N$ patterns of pixels present in A.

ii. distribution/density of $N \times N$ patterns in B should be similar to A (if B is large enough).

A Simple but Curious Algorithm (WFC)



WFC initializes output bitmap in a completely unobserved state, where each pixel value is in superposition of colors of the input bitmap (so if the input was black & white then the unobserved states are shown in different shades of grey). The coefficients in these superpositions are real numbers, not complex numbers, so it doesn't do the actual quantum mechanics, but it was inspired by QM. Then the program goes into the observation-propagation cycle:

- On each observation step an NxN region is chosen among the unobserved which has the lowest Shannon entropy. This region's state then collapses into a definite state according to its coefficients and the distribution of NxN patterns in the input.
- On each propagation step new information gained from the collapse on the previous step propagates through the output.

On each step the number of non-zero coefficients decreases and in the end we have a completely observed state, the wave function has collapsed.

It may happen that during propagation all the coefficients for a certain pixel become zero. That means that the algorithm has run into a contradiction and can not continue. The problem of determining whether a certain bitmap allows other nontrivial bitmaps satisfying condition (C1) is NP-hard, so it's impossible to create a fast solution that always finishes. In practice, however, the algorithm runs into contradictions surprisingly rarely.

- lowest entropy + belief propagation (a bit like Minesweeper, but without pre-existing map)
- can run into contradiction (which is similar to Kochen-Specker configuration)

To Make it More “Quantum-like”

A tilemap is like a quantum system where:

- each pixel is an observable (whose outcome is non-binary).
- each small $N \times N$ tilemap A is a set of “compatible observables”.
- contextuality arises where there is no globally consistent assignment.

To make it more “quantum-like”:

- allow global contradiction (fuzzy tiles) to exist.
- allow global contradiction to “propagate”.
- the player keeps its own “consistent memory/belief” of the global map.

First-Person Perspective and QBism

The tilemap rendering model can be compatible with QBism:

- When there is only one player: easy.
- When there are 2 players: inter-subjectivity arises, no referee since no global consistency.