Non-Arbitrary Spaces: Algorithmic world generation from blockchain data.

ChainState

www.chainstate.xyz

contact@chainstate.xyz

Abstract. The central data structure of the Bitcoin network[1] is the blockchain; an ever-growing store of data which provides an immutable ledger of all on-chain Bitcoin transactions. Although the notional purpose of this data is to track transactions, it can also be repurposed.

This paper demonstrates that useful worlds can be created by organizing blockchain blocks into novel arrangements and using the data within blocks as function inputs.

1. Introduction

This document presents a concept to automatically generate a complex landscape which we will call a *Non-Arbitrary Space*, derived purely from on-chain data present on the Bitcoin network. It builds on the published work of the Blockrunner[2] team, which conceptualizes the use of Digital Matter Theory[3] (DMT) and formalizes Non-Arbitrary Tokens (NATs) derived from defined patterns in blockchain data.

2. Definitions

As many of the concepts and terms used in this paper are novel, they are defined below for clarification:

Digital Matter Theory The Blockrunner team summarized the concept as "Digital Matter Theory proposes that it is possible to create a form of digital substance by leveraging the inherent patterns present in data." The concept of a Non-Arbitrary Space proposed in this paper falls within this theory.

Non-Arbitrary Tokens Non-Arbitrary Tokens or NATs are a new type of cryptographic token that are associated with a user-defined pattern called an element.

Non-Arbitrary Spaces As the Bitcoin blockchain grows, it adds new blocks to the chain. The concept of Non-Arbitrary Spaces (NAS) proposes to arrange these blocks in a non-arbitrary manner to create a usable 'world' for applications such as gaming.

Element These represent the building blocks of digital matter formation, where any piece of block data can identify a pattern on Bitcoin's blockchain.

The image below shows an example relationship between NASs, BITMAPs, NATs, and elements.

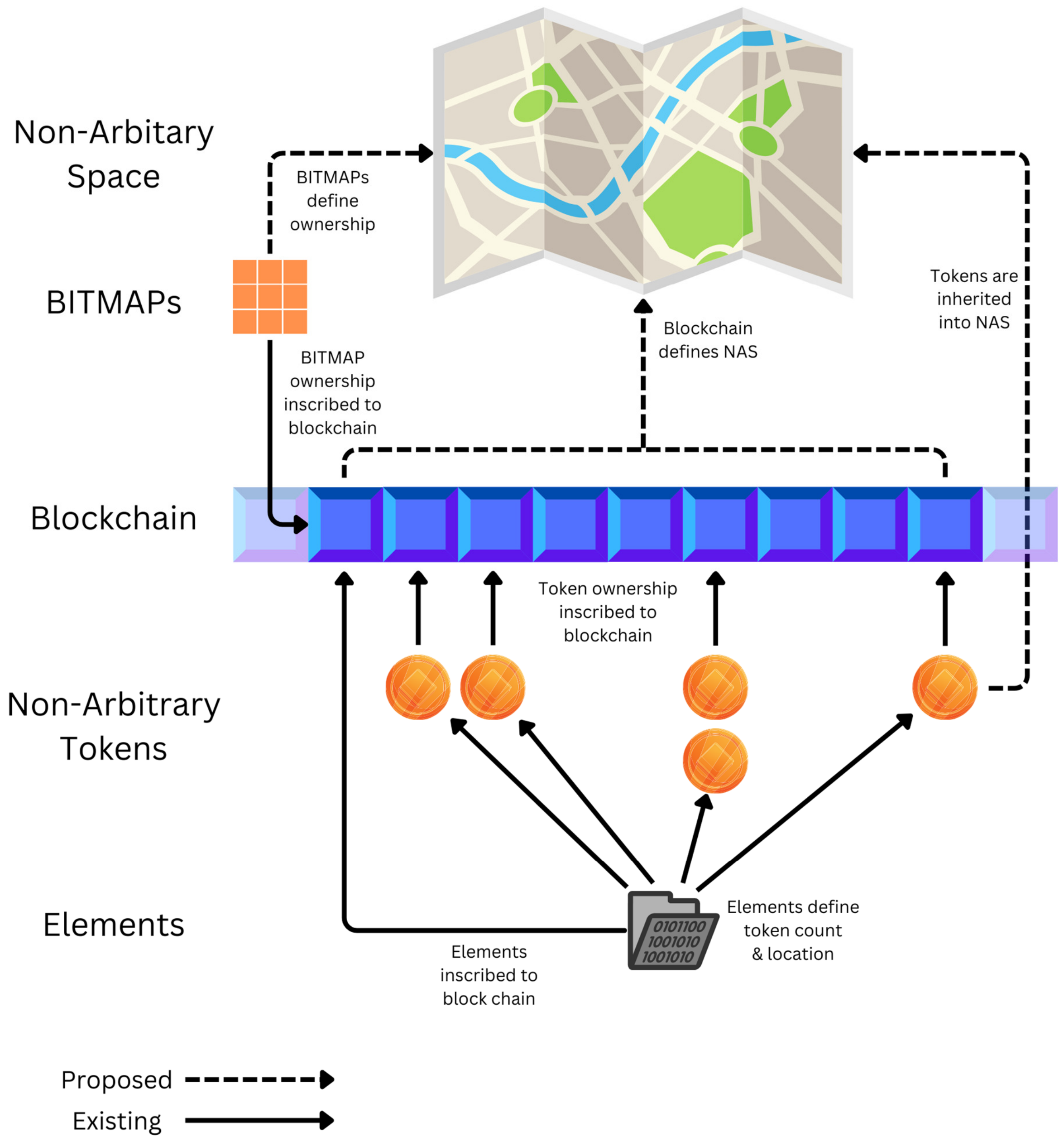
Non-Arbitary Space
BITMAPs define ownership
BITMAPs
BITMAP ownership inscribed to blockchain
Blockchain defines NAS
Tokens are inherited into NAS
Blockchain
Token ownership inscribed to blockchain
Non-Arbitrary Tokens
Elements
0101100 1001010 1001010
Elements define token count & location
Elements inscribed to block chain
Proposed
Existing

Figure 1: Proposed relationship between NAS and existing data structures.

3. A Simple NAS

The existing arrangement of these blocks is a linear chain, which results in a space of dimension 1 by n, where n is the number of confirmed blocks on the network. As the number of blocks is currently over 900 000 and forever increasing, this creates an extremely long and thin ‘world’ which has little application.

NASs provide a solution to this problem by arranging blocks into more useful forms. For example, we could rearrange the first 10 000 blocks into a grid of dimensions 100 x 100 to be used in a game environment.

The problem is that the choice of 10 000 is itself arbitrary; our use of base 10 or indeed any number system is arbitrary as far as the Bitcoin network is concerned. (For example, Bitcoin code uses several bases including 2, 10 and 16; the original Bitcoin address format is base 58.)

The Bitcoin difficulty retarget period is 2016 blocks. This is a non-arbitrary number for the network, as the difficulty target is static throughout this period. Difficulty is recalculated and updated at the end of each period, aiming to make the average block creation interval 10 minutes.

This number is also relatively large compared to the number of blocks currently mined. As shown in the table below, a width of 2016 blocks creates a usefully proportioned world, and won’t approach a ‘long skinny’ world for the best part of a millennia.

So a width of 2016 is a good choice since it is both non-arbitrary and useful.

With a width defined by the retarget period, each row of the grid now has one uniform difficulty. It therefore becomes an x-axis variable which is the count of *retarget periods*. Likewise, the y-axis now measures *blocks since difficulty retarget*.

This creates perhaps the simplest NAS, shown below:

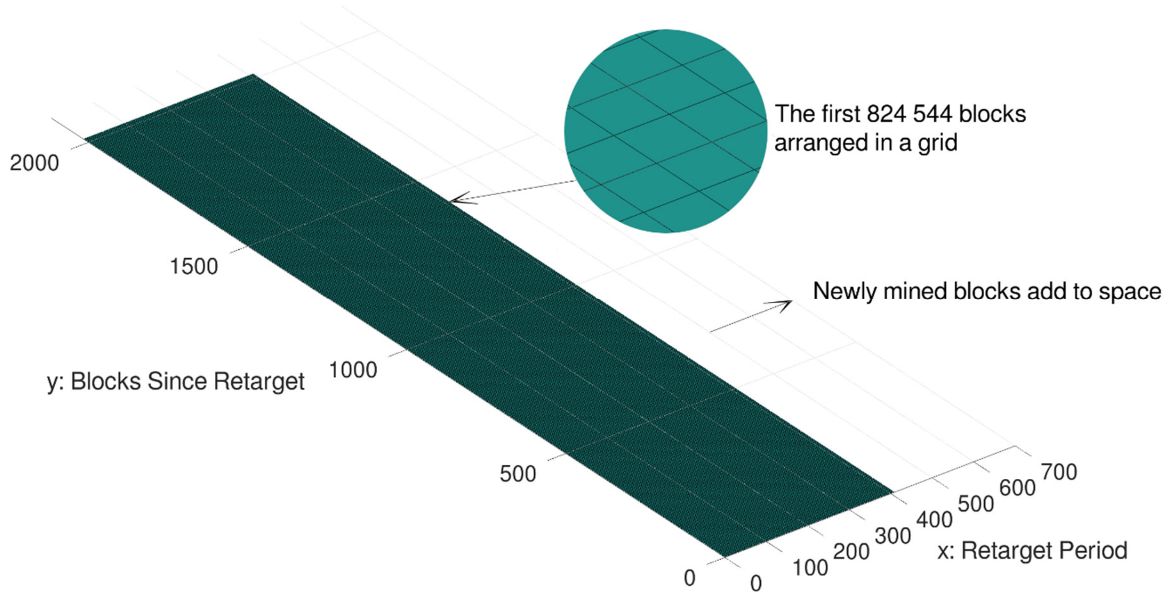

Figure 2: A simple NAS created from the first 824 544 blocks.

The dimensions of this NAS will grow as the blockchain does. The table below shows some milestones for this.

Block Height	**Event**	**Proportions**	**Real-world area (km^2)***	**Date**
823 785	Last block mined in 2023.	1:4.3	8 238	Dec 31 2023
4 064 256	‘Square’ world.	1:1	4 064	~2086
6 929 999	Last mining reward issued.	1.7:1	6 930	~2140
40 642 560	‘Long skinny’ world.	10:1	40 642	~2781

Table 1: Future NAS dimensions

*Assuming each block is equivalent to a city block of approximately 100 x 100 m.

4. More Useful NASs

Although choosing a width of 2016 blocks produces a usable area, it is still currently flat, which isn't too useful.

To fix this, we need to use some non-arbitrary input to determine the elevation of each block.

As we're trying to make elevation changes larger in scale than individual blocks, transaction count is a good choice. This number is defined per block, however it is typically consistent across multiple blocks, changing with slower moving variables such as network adoption and market sentiment.

To define elevation, we can use data on the blockchain as an input. This should be chosen to create some realistic terrain. We could use the number of transactions in a block to determine the elevation of that block. This results in the following NAS:

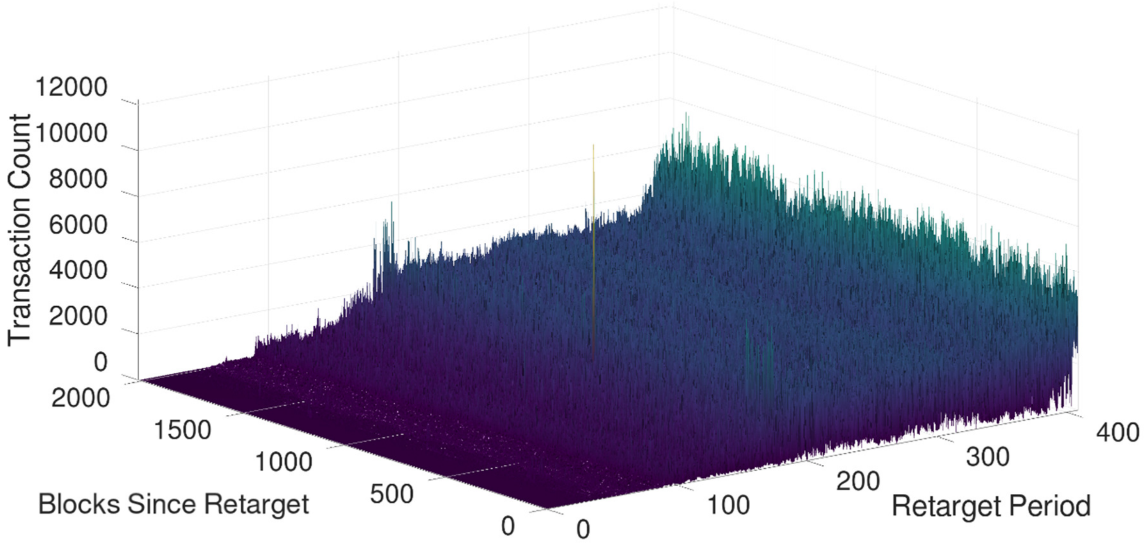

Figure 3: A NAS where elevation (z-axis) is determined by the number of transactions in each block.

Already, this has some semblance of a landscape, however the data is noisy, and the colors are generic.

This can be fixed by averaging the data and applying a more natural color gradient. The figure below uses the same data from Figure 3, while adding a 5x5 block linear average, and a color gradient determined by elevation.

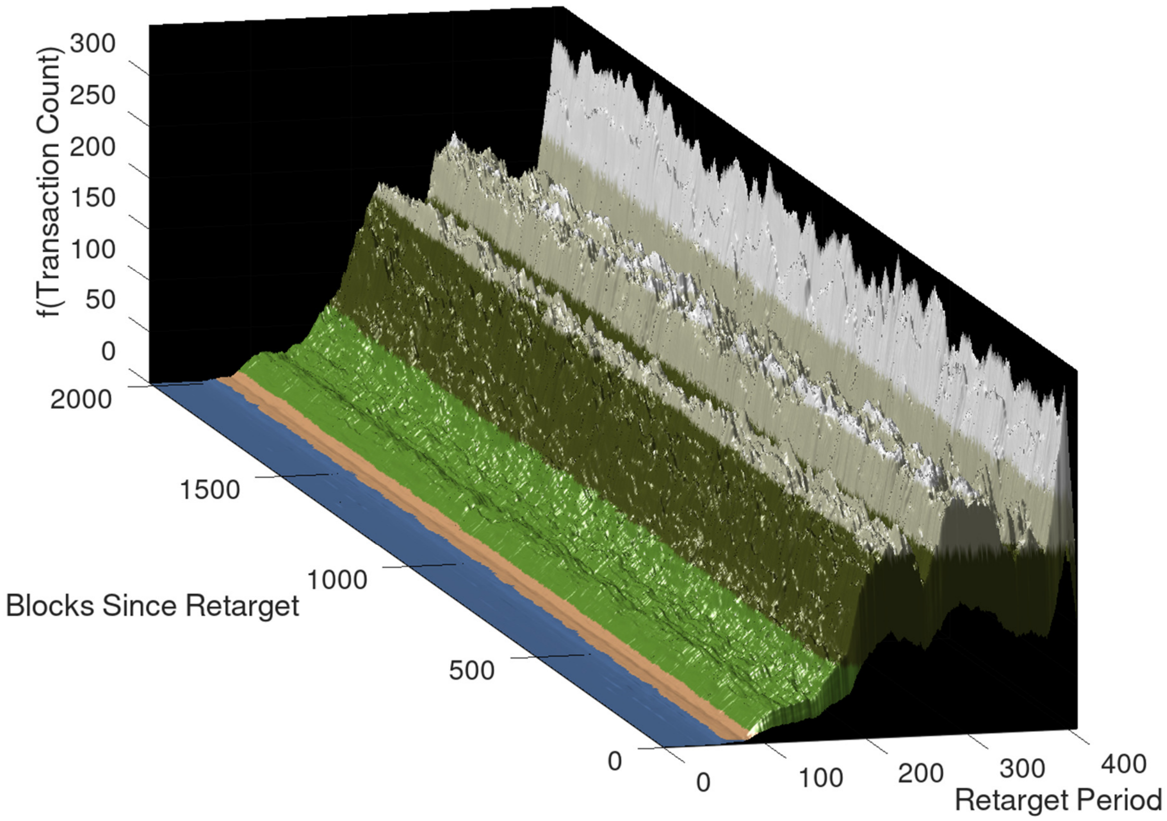

Figure 4: A NAS where elevation (z-axis) is determined by the linear average of a 5x5 array of blocks.

This still could be improved; the land is quite consistent in the y direction, and ends abruptly on each side at the retarget blocks. As the blockchain grows, it will print a long rectangle of quite homogenous land that may become monotonous.

We can add variations in each direction by weighting the transaction count of each block according to the coordinates in the space.

To continue with the non-arbitrary theme, we can use *blocks since retarget* and *blocks since halving* as the weighting references. To create a more organic world, a bell curve (as shown in Figure 5 and Figure 6 below) can be used, so that if a block is further from these two events, the transaction count will be weighted more, resulting in a higher elevation.

Figure 7 show the NAS that this creates. Although there are some arbitrary choices here, the world is still a function of blockchain data only.

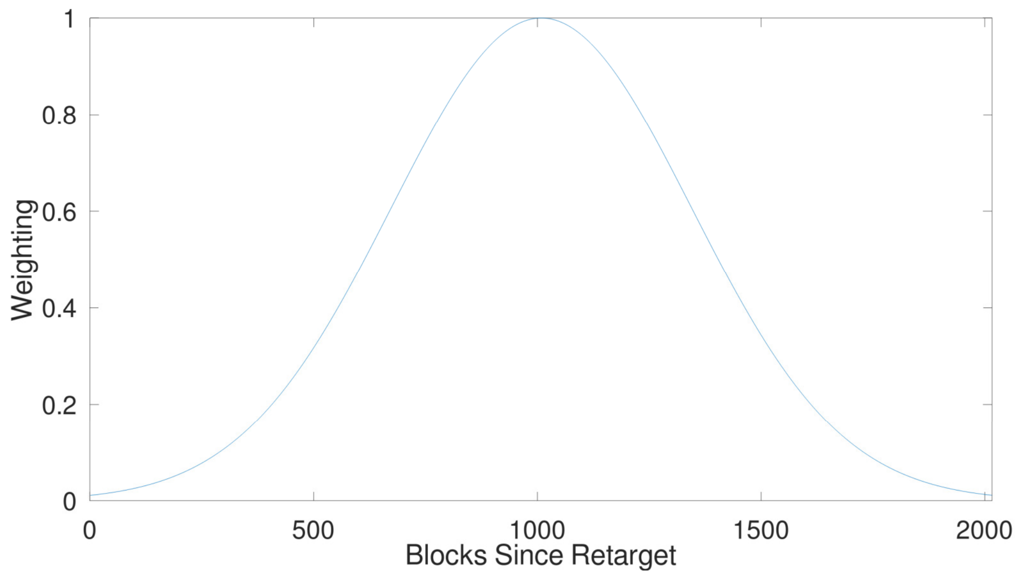

Figure 5: A normal distribution, mapped from 0 to 1 across the y axis.

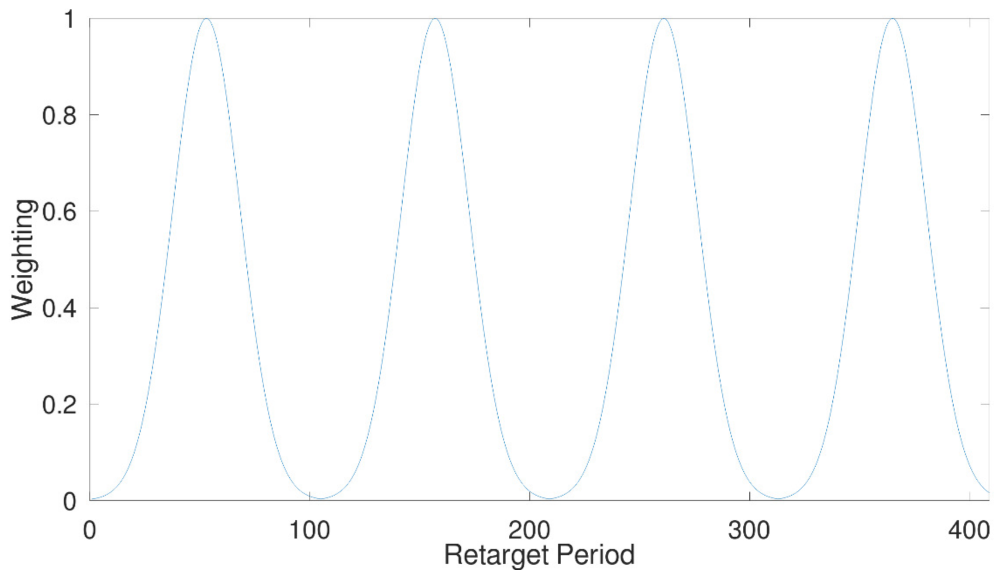

Figure 6: A normal distribution, mapped from 0 to 1 across the y axis, with period of 'blocks from halving'.

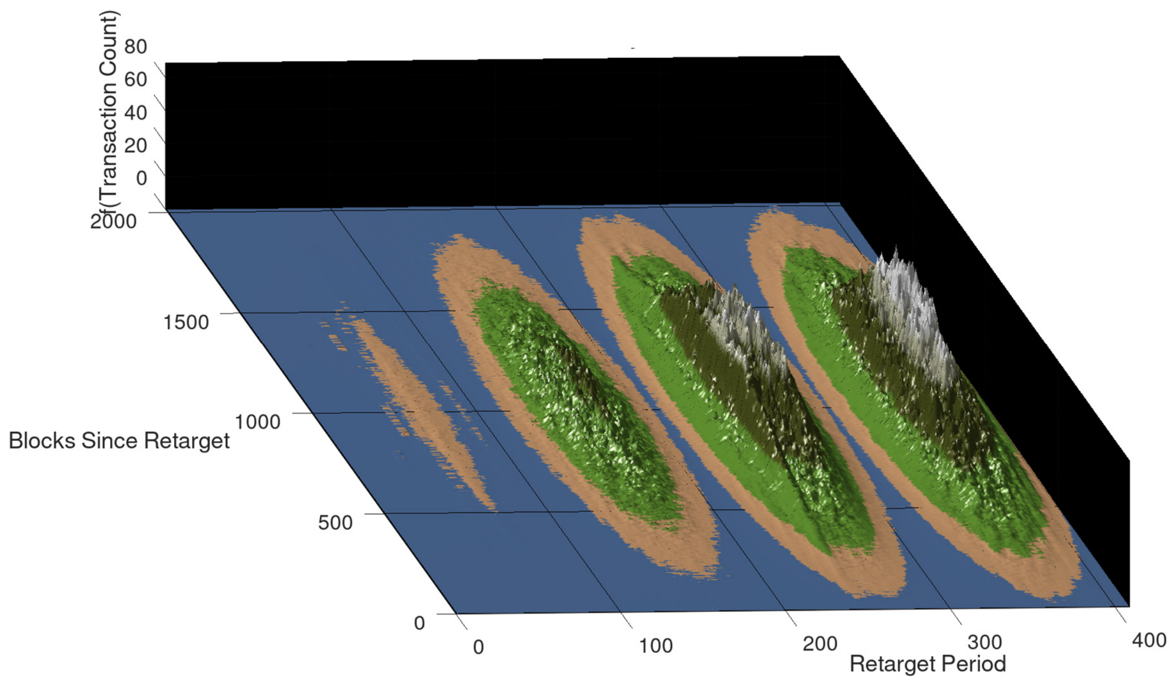

Figure 7: A NAS where elevation (z-axis) is determined by the linear average of a 5x5 array of blocks, weighted to distance from both a retarget block and a halving block.

5. Blockchain history in a NAS

A consequence of using on-chain, non-arbitrary data to define the landscape is that the history of Bitcoin, and indeed the larger Bitcoin community, is represented in the space created. Everything from network adoption, market sentiment, regulatory conditions, mining technology, and protocol changes have affected both transaction throughput and network difficulty. Figure 8 below shows some of these features.

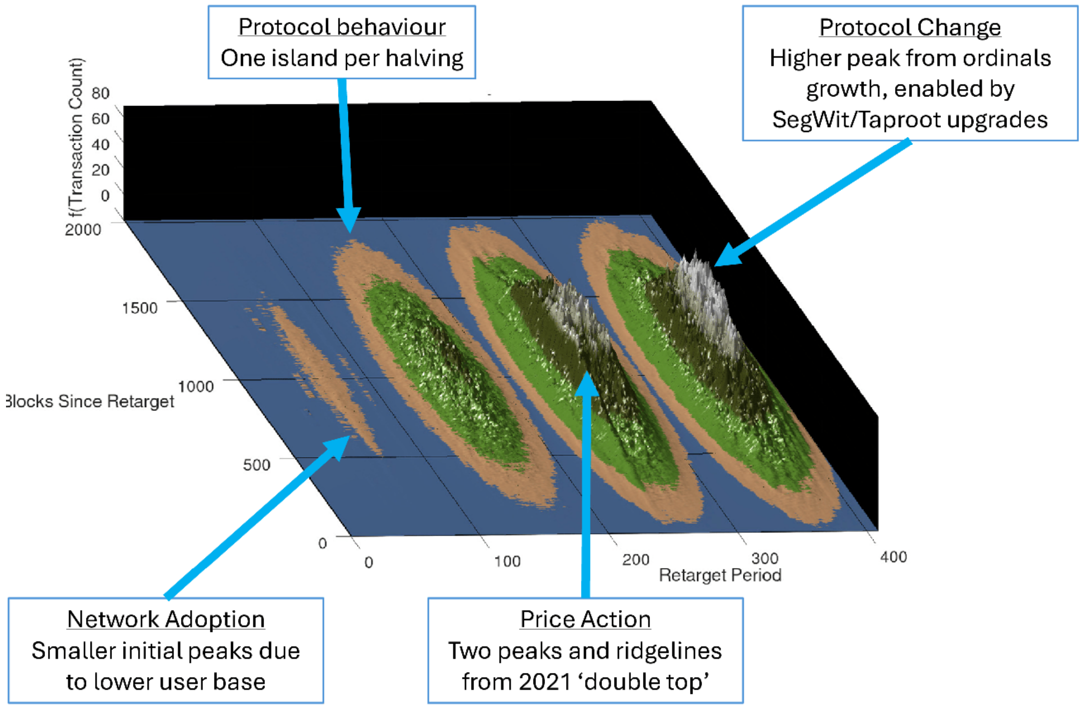

Figure 8: Effect of real-world behaviour on a NAS

6. Application of NATs to NASs

Since these two artifacts are both defined from the same data set, they work very well together.

A good example would be to use a NAT to represent a resource such as gold in the landscape. This could ensure that gold is deposited in non-arbitrary locations and is mined along with each new block should the NAT's element be present. Even the 'hybrid token' function can be used, whereby the location of each token is non-fungible, but the trade value remains fungible.

NATs could also be used to define the landscape itself, whereby a NAT increases the elevation around the block it matches with.

An example of this is in Figure 9 below. The features of the NAS are determined by the NATs in the following Table 2:

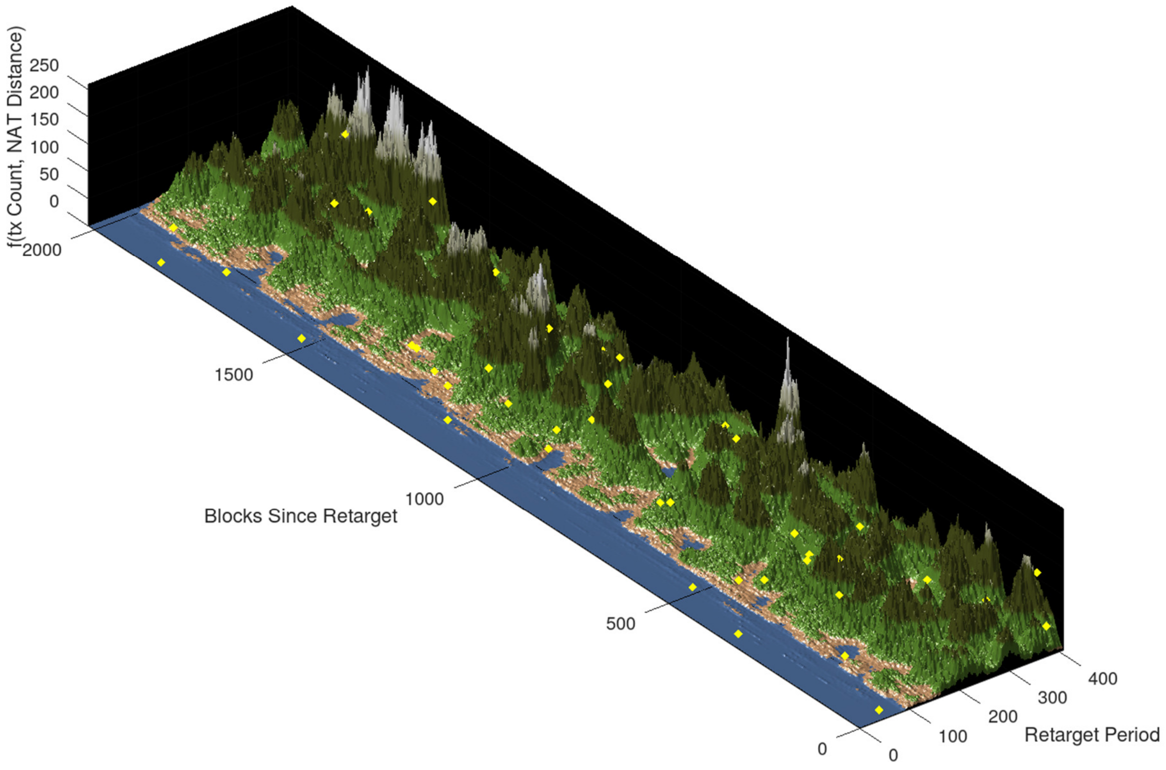

Figure 9: A NAS where elevation (z-axis) is determined by the linear average of a 5x5 array of blocks, inversely weighted to the distance from a NAT. Yellow diamonds mark the location of blocks containing the element "gold.10409.7.element"

NAS Feature	Block Field	Pattern	Element registry	Qty	Example Match (matching block #)
Elevation source	Transaction count	(none-whole field)	elevation.14.element	824544	**457** (Block 210 000)
Elevation weighting (~Mountain peak location)	Merkle Root Hash	“2009”	weight.2009.7.element	812	9339603183ff7a267eb4fa98920097979520a6c400bc771f50cb81de1b481867 (Block 405)
Gold resource location	Merkle Root Hash	“10409”	gold.10409.7.element	52	43da91cac10a1a4fb09d5a1893477c8f99104096ccfc06645a7133180b48e9d8 (Block 7878)

Table 2: NAS variable inputs

The choice of transaction counts and Merkle tree hashes for the features above creates two different data source types. Transaction counts roughly correlate to network utilization, which generally increases with height up to the bandwidth of the network. Merkle hashes however, are pseudo-random in nature.

This NAS therefore has mountain peaks in pseudo-random locations, with heights increasing as the block count increases (the first block with a transaction count of 1000 doesn’t occur until height 226804).

The gold resources are pseudo-randomly distributed throughout the space, but are fewer in number since a longer pattern is used.

7. Application examples

The image below represents a landscape defined by the NAS above, with textures procedurally generated, based on surface height and gradient. This could be used in a game environment.

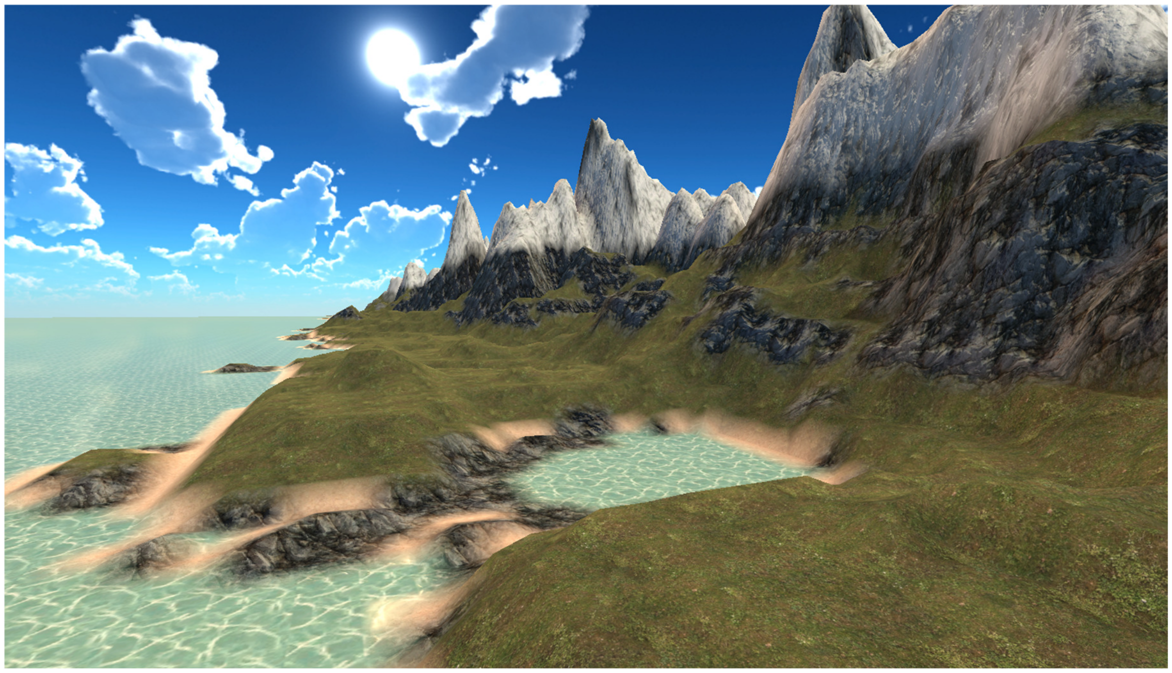

Figure 10: A rendering of a landscape defined by the blockchain.

The figure below shows a NAS wrapped into a sphere. This could be used as a complete world in a game, which grows as new blocks are mined.

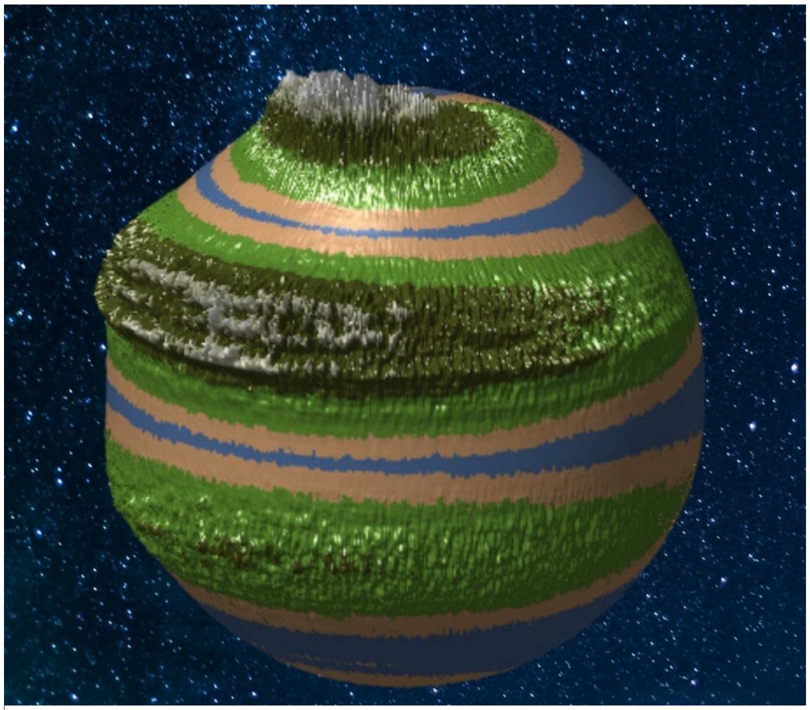

Figure 11: A spherical NAS

8. Recording of NASs on the blockchain

Either inputs to each dimension (eg, x, y and z) could be recorded, or the dimension and entire z function. Since these functions use only on-chain data as inputs, no other datasets are required.

A JSON script for the NAS shown in Figure 3 may look something like:

```
{
“name”: “<name>”,
“shape”: “<rectangle | sphere | toroid | cylinder | custom>”,
“dim1”: "retargetInterval",
“dim2”: "tx_count"
}
```

with the third dimension determined by height as a function of "dim1".

9. Conclusion

The structures defined in this paper demonstrate that spaces can be created using existing on-chain data. These are large, complex, useful worlds which can be generated procedurally for as long as Bitcoin continues to exist.

References

[1] Satoshi Nakamoto “Bitcoin: A Peer-to-Peer Electronic Cash System” https://Bitcoin.org/Bitcoin.pdf, 2009

[2] https://www.theblockrunner.com/

[3] https://digital-matter-theory.gitbook.io/digital-matter-theory/

Figure 1 created with Canva (https://www.canva.com/).
Figures 2-9, 11 created with Octave 9.2.0 (https://octave.org/).
Figure 10 created with Unity 5 (https://unity.com/)

Revision 1.0 released 1 February 2026