Game of MetaLife 0.33

Joel Dietz [a], Peter Suber [b]
[a] Computational Law, MIT, Cambridge, MA
[b] Berkman Klein Center, Harvard University, Cambridge, MA

Abstract

We built a gamified simulator for new computational lifeforms by combining lambda calculus and cellular automata via a custom purpose programming language that has been designed to run simulations across multiple rendering engines (i.e. metalambdas). In short, the authors have discovered (a) it is possible to embed lambdas into cellular automata in a 3d environment (b) these automata can genetically absorb each other, evolve into more complex lifeforms across multiple rendering engines (c) these lifeforms can secure their own computational resources through web3 infrastructure (d) various incentives can be deployed to allow for a more dynamic evolutionary experience (e) these lifeforms can be imported into and be used inside game environments.

Keywords: Game of Life; Zuse Hypothesis; Cellular Automata; Computational Lifeforms; Von Neumann Universal Constructor ; Lambda Calculus; Nomic; Monads; DAOs

1 Introduction

Conway's Game of Life is well known. A small set of preconditions allows complex life forms to emerge and achieve a stable state. Numerous implementations document the many patterns that emerge including extending it to multiple dimensions. What Conway's Game of Life does not do, however, is give the lifeforms any possibility of evolution. Inserting lambdas into a cellular automaton allows the automaton to express variable complex objects and, if it absorbs other automata, can grow genetically. This is similar to what Von Neumann suggested in his "theory of self-constructing automata," except that this introduces one additional variable, namely that the simulation takes place in a game environment in which the lifeform can compete alongside other lifeforms.

Similarly Konrad Zuse discussed the possibility of computational universes, including the mathematics of such, and the appropriate algoirthmms for computing them.[1] MetaLife,

[1] Konrad Zuse, Rechnender Raum, Friedrich Vieweg Sohn, Braunschweig, 1969. English translation:

in this sense, is an implementation of the Zuse hypothesis made possible with web3 incentives. Similarly, von Neumann suggested that cellular automata could evolve in a similar direction (i.e. as a universal constructor) where they could replicate themselves and evolve in various directions.

2 Metagame engine for creating computational universes

Seeding computational universes requires first creating cellular automata and providing an environment where they can evolve. This means that that each lifeform must have its computation inside of it in a common format that can be deployed across rendering engines. This common we format refer to as a **metalambda** and is a computationally measured version of lambda calculus that can be deployed cross platform along with a metaurl (i.e. fractal addressing space). Consequently, key features of m2 are compositionality, fractal subdivision of space, the ability to set environment variables within a particular space, and the ability to run simulations forward and backwards in time.

2.1 Units of account

To manage this, we must start by dividing space and time into composible parts. Euclidean space is divided into the length of cubes. A first order cube is one Neumann meter long. Additional cubes exist at a 100x scale factor smaller or Einstein meters, such that each cube is the set of the 1,000,000 sub-cubes = 100^3.

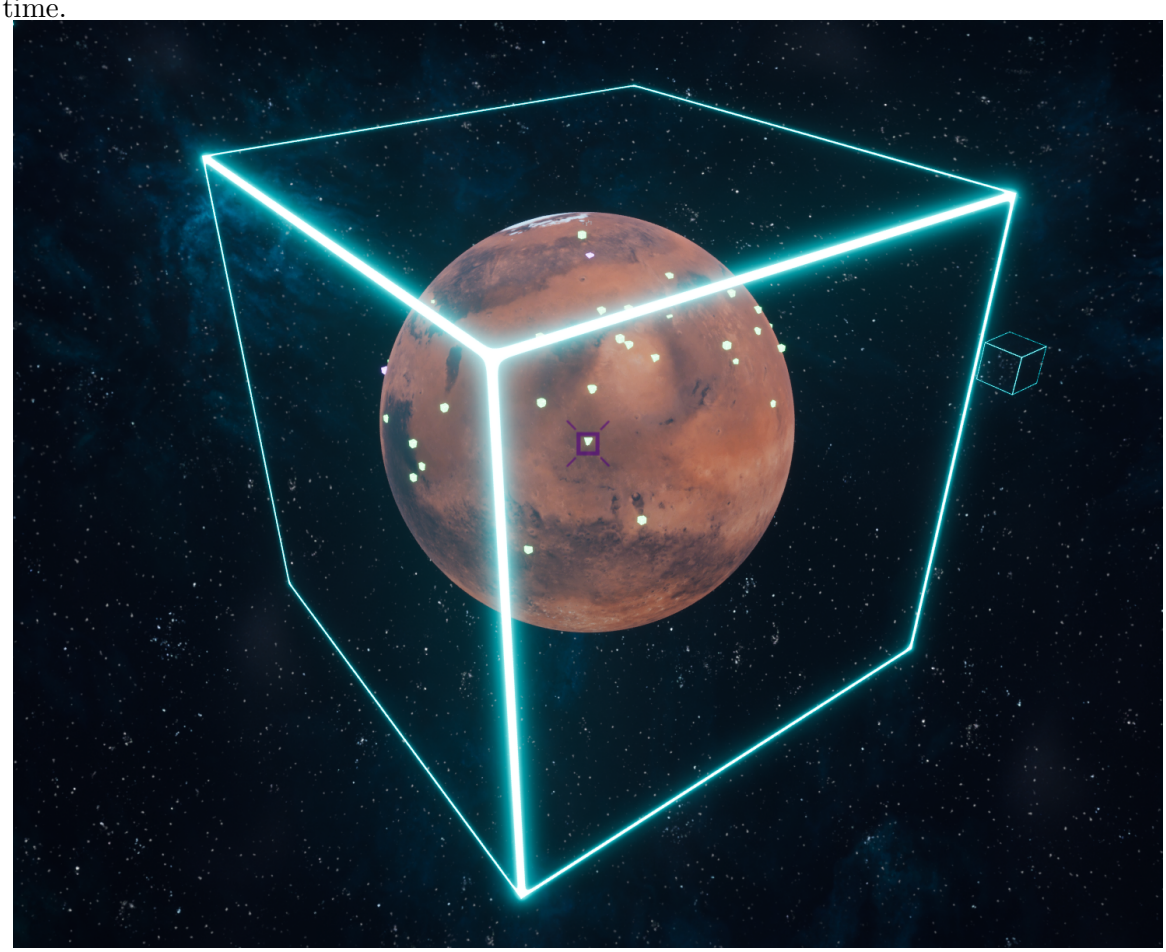
Naming conventions have been derived from the primary contributors to the concept of a computational universe.

A second in m2 is a computational step and may be subdivided. Consequently, a Max second is 1 computational step, whereas a Descartes second represents 100 computational steps.

Calculating Space, MIT Technical Translation AZT-70-164-GEMIT, MIT (Proj. MAC), Cambridge, Mass. 02139, Feb. 1970. PDF scan. More info here: https://people.idsia.ch/ juergen/digitalphysics.html

Unit Name	Meter	Time	Math
Max Planck	mm	ms	10^0
Decartes	dm	ds	10^2
Wolfram	wm	ws	10^4
Zuse	zm	zs	10^6
Penrose	pm	ps	10^8
Einstein	em	es	10^{10}
Neumann	nm	ns	10^{12}

With this framing I now can refer to the exact location of all objects in all Euclidean spaces, including all metaverses that are implemented in Euclidean geometry at any point in time.



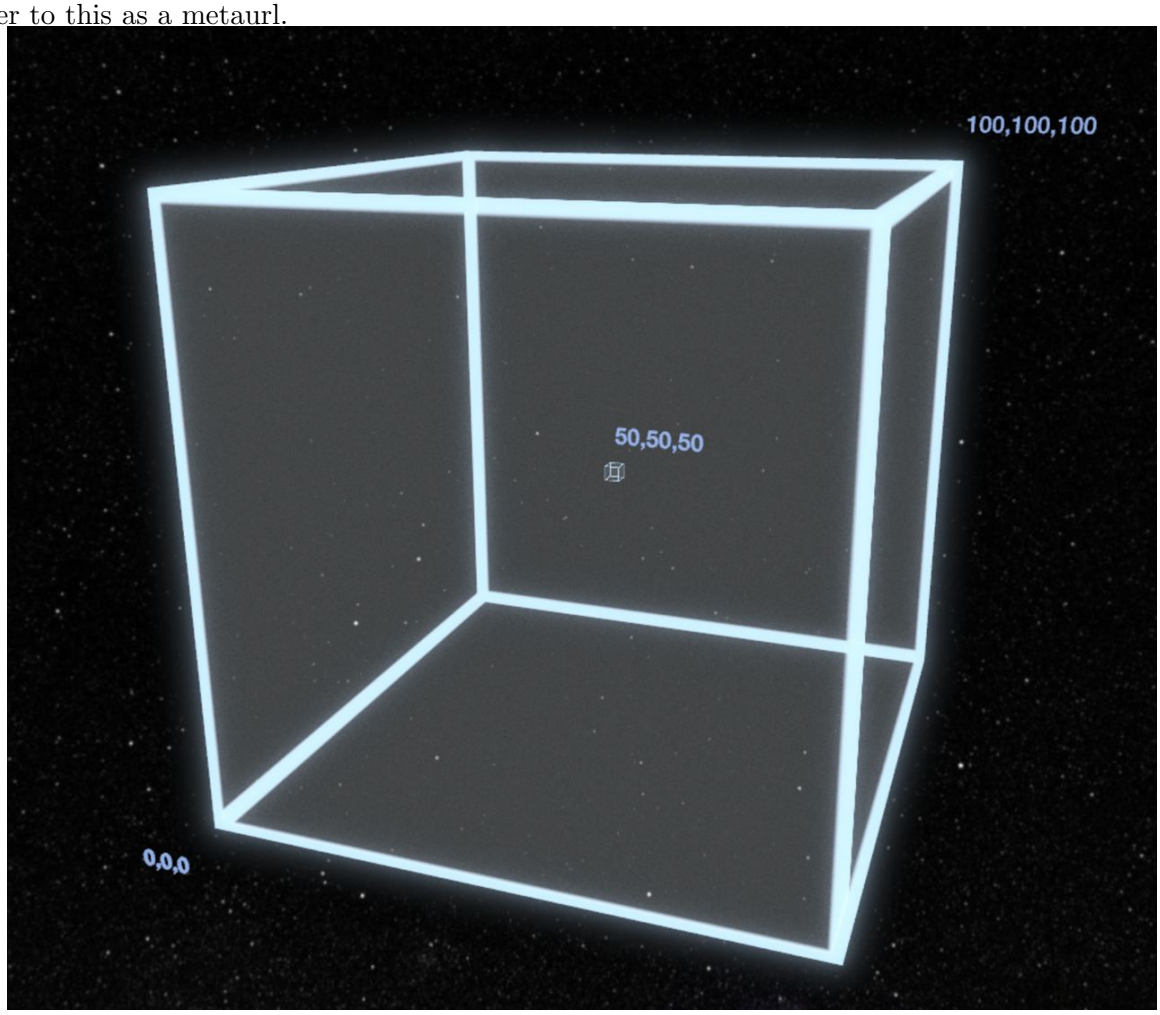
For example, one can place Mars inside a Neumann length cube and then place smaller cubes inside of it that describe the land tangential to the surface of mars, and then further subdivide.

2.2 MetaUrls

This vector-based system of sub-division of space can also be expressed by a url such as follows:

`/50.50.50`

This equates to the space at the coordinates 50,50,50 in neumann meters (or, more technically, the vector represented by +50, +50, +50 in the Euclidean x,y,z) We will hereafter refer to this as a metaurl.



`/1.2.3/2.100.3/2.32.30/3.3.3`

/in Neumann meters/in Decartes meters/Wolfram meters/Zuse meters

In short, this describes a location at four levels of fractal depth.

One can run a simulation at the cube which is at coordinates at this depth by passing the appropriate metalambda command.

`/1.2.3/2.100.3/2.32.30/3.3.0/create(x.l(go(right)))`

In short, I have run the create life function to create a cellular automaton and passed that automaton a function which will be its instructions at each computational step.

Should one desire to see some future state, one can additionally pass time, in this case 1 Wolfram second (i.e. 10,000 computational steps).

```
/1.2.3/2.100.3/2.32.30/3.3.0/create(x.1(go(right)))+1ps
```

The features of function passing will further illustrated in future sections.

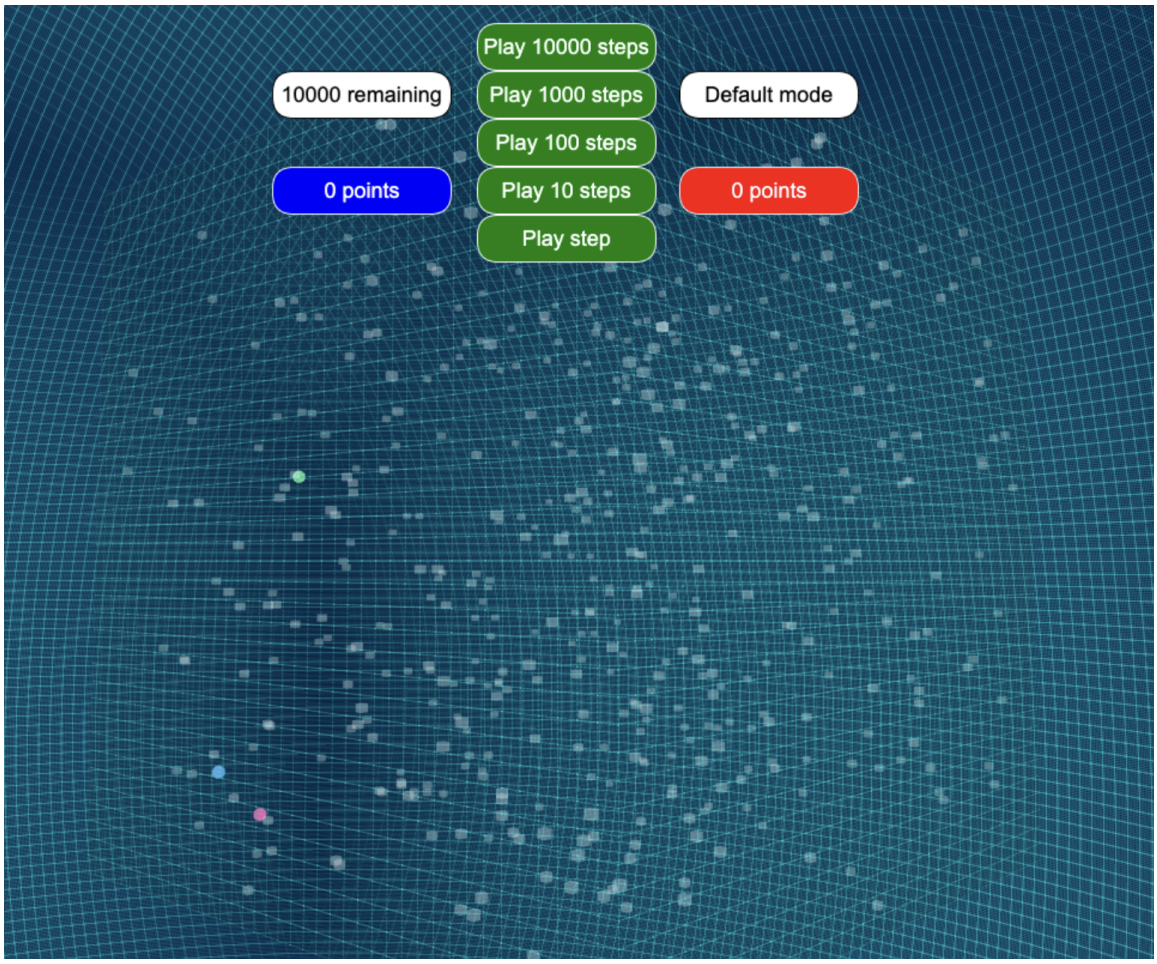
2.3 Game starting condition and rules

Each simulation has a set of starting conditions that can also be describe as a game with certain rules.

2.3.1 Rules

For example, the starting rules expressed as a set:

1. Game takes place on a 100x100x100 grid.
2. Game length is 10,000 computational steps.
3. Each lifeform begins with 10,000 energy
4. In each computational step all appropriate physics rules are applied (e.g. rules derived from Conway's game of life)
5. In each computational step all lifeforms perform one function.
6. With each function usage 1 energy point is deducted
7. 5000 food cubes are randomly scattered around the grid with 1000 energy each
8. Lifeforms consume inner food cubes and other lifeform if that lifeform has less energy.
9. Starting lifeforms are generated at with a random seed in a range from 1-100. This seed is used to determine the use of "go" and "sensing" functions.



2.3.2 Key Variables

The starting variables consist of:

The size of the game board By default 100x100x100 cubes.

The existence of any binding physics rules By default none, only that the maximum movement per computational step in any direction is +1 or -1 in any direction.[2]

Energy The amount of scattered energy, by default 5000. Computational lifeforms require energy to operate.

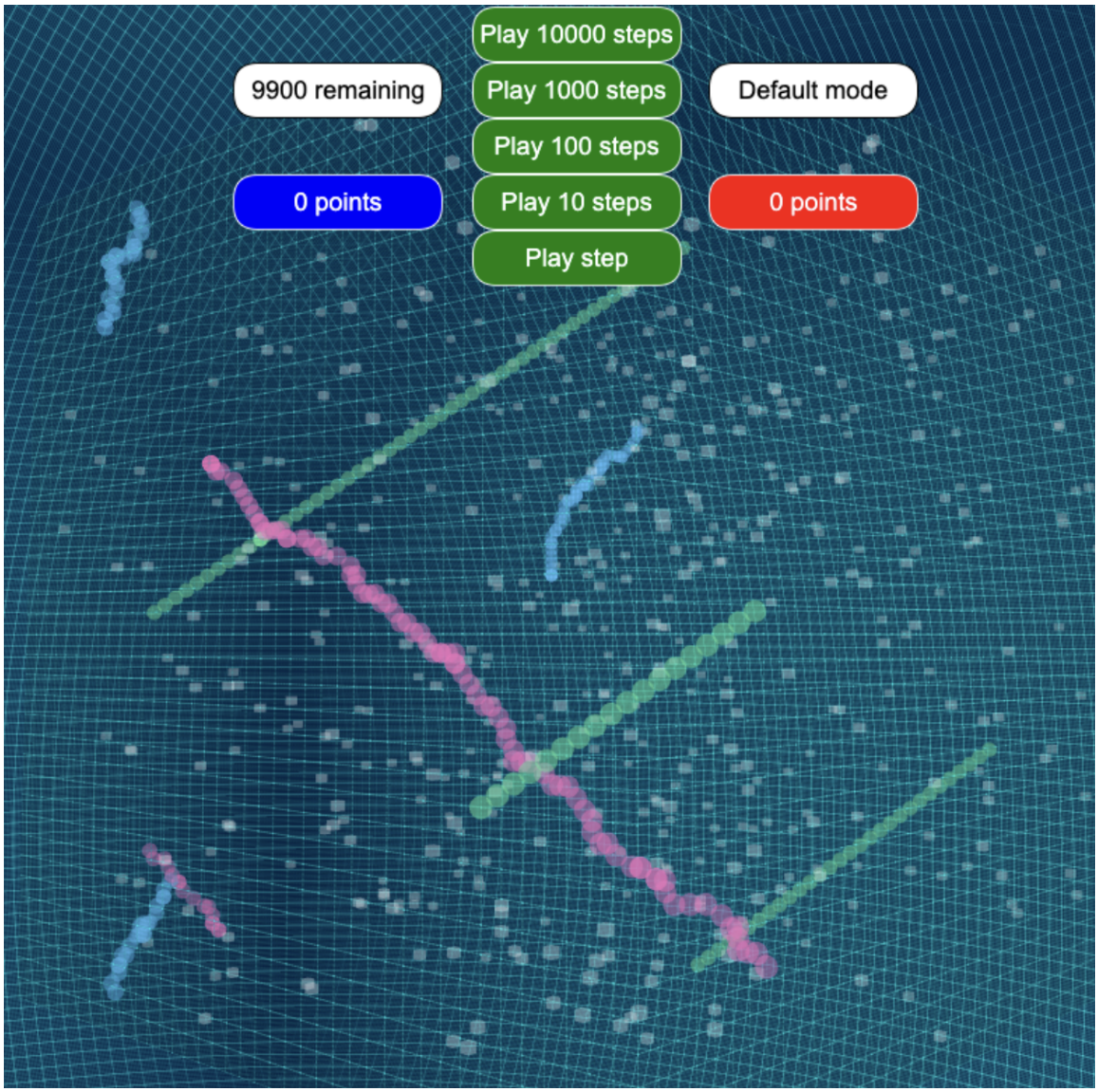
Lifeform energy The amount of starting energy, of each lifeform.

[2]Integrating variable physics rules is the subject of our collaboration with the Wolfram Institute and builds on Stephen Wolfram's work on metamathematics and the ruliad /urlhttps://writings.stephenwolfram.com/2021/11/the-concept-of-the-ruliad/

Seed Default seed, passed as a hexadecimal string and then used to seed the universe with certain computational lifeforms. This could be an ethereum address.

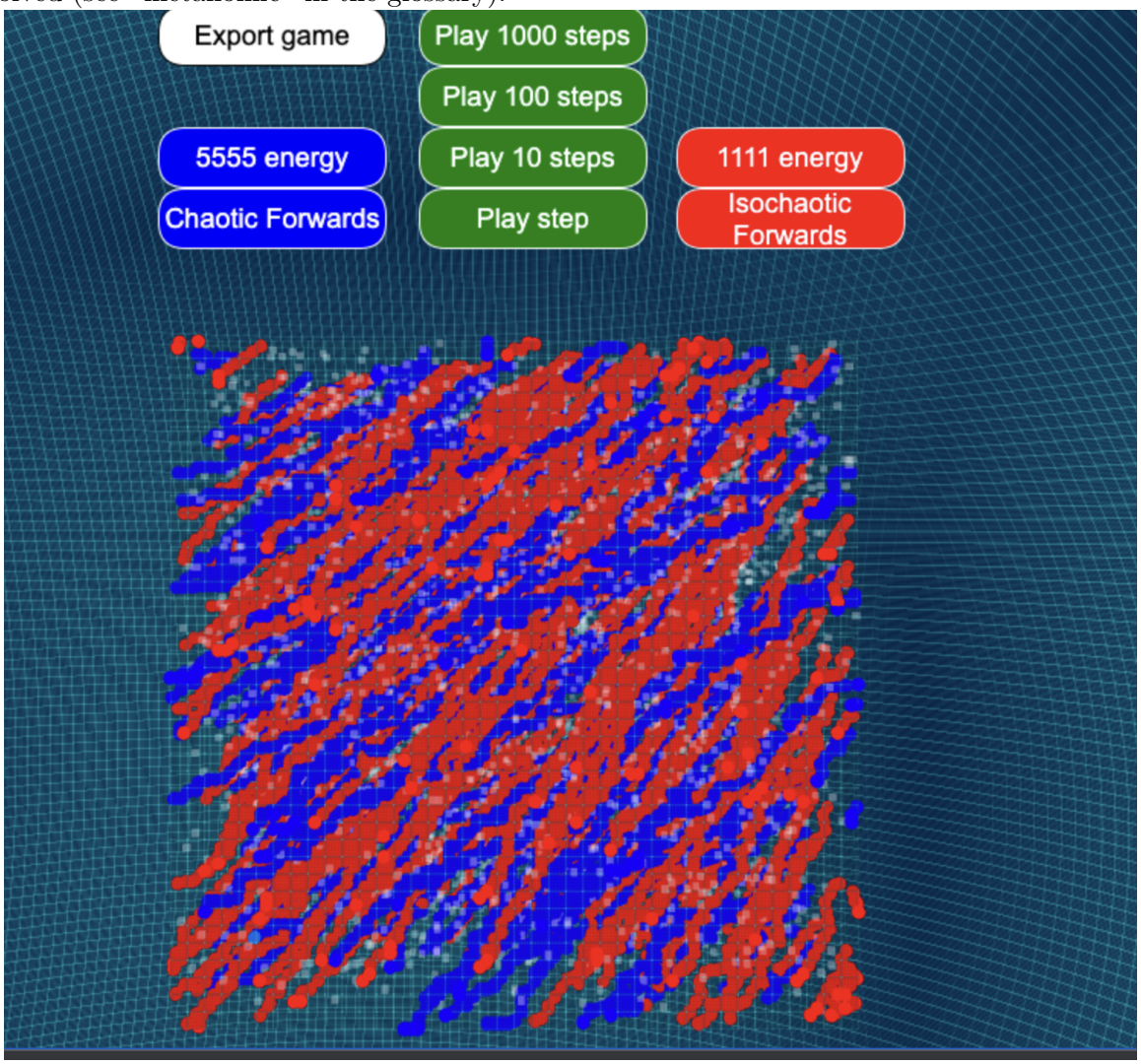
Winning Conditions By default, the lifeform that has the most energy at the end of a particular number of computational steps wins.

Available functions Passed in one or more files in a standard "metalambda" format.



The game can be run in competitive mode, where people chose specific conditions (and their own functions) and play to the end, or pseudo-randomized where the game conditions are dynamically change include genetic seeding and mutation for the various functions

involved (see "metanomic" in the glossary). [3]



First finished game of metalife that was run over 10,000 computational steps with two competing lifeforms using different algorithms in an interactive native 3d BabylonJS powered simulator.

[3] Some competitive modes already built allow for the insertion of additional objects in the playing field and for the human player to "intervene" and take over a particularly object (such as a spaceship), thus creating a non-deterministic model of resolution.

3 Implementation Details

3.1 Overview

The following implementations have been built:

Language	Rendering Engine
Javascript	BabylonJS
C♯	Godot
C++	Unreal Engine

Each of these has certain benefits. The Javascript version is accessible in the web browser. Godot is embeddable in a cross platform browser and supports some pre-render. Unreal Engine allows for the highest quality visuals. However, the ability to run parallel implementations at scale is somewhat limited.

3.2 MetaLambda Specification

3.2.1 Introduction

Metalambdas are cross platform lambda functions which control computational lifeforms.

Specifically they must follow the following:

1. Allow all of the functions of the metalambda spec.
2. Measure the computational time necessary for each function call.
3. Cut off function execution time if it exceeds appropriate time limit.
4. Implement the metalambdas according the game rules (i.e. one metalambda execution per computational step)
5. Allow for the insertion of a "seed" which allows mutations

This means that each game of metalife implementation is, by default, fully deterministic and that with same starting conditions you can verify the same results will be reached regardless of the game engine.

3.2.2 OPCODES

1. If/then
2. Math (addition, subtraction, division, modulous)
3. Go (move object in any direction in the x,y,z axis)
4. Insert (insert an object of a certain type in the space)

3.2.3 Anticipated Future OPCODES

1. Sense (determine the location of the nearest energy or lifeform within a certain range)
2. Absorb (combines the genetic code of a lifeform with the current one)
3. Mutate (changes factors of the lambdas)

3.2.4 Available environment variables

Environment variables that can be accessed into lambda.

1. NUMBEROFCOMPUTATIONALSTEPS - number of computational steps completed so far

3.2.5 Metalambda injection

```
allActiveLifeforms.forEach(function (current, index, arr) {

                var lastLifeform =  current;
                currentLifeform = metaLambda(LastLifeform.seed,LastLifeform);

                }
```

In this case (the babylonJS implementation), you can see that the seed is passed separately from the lifeform itself and may be subject to additional rules.

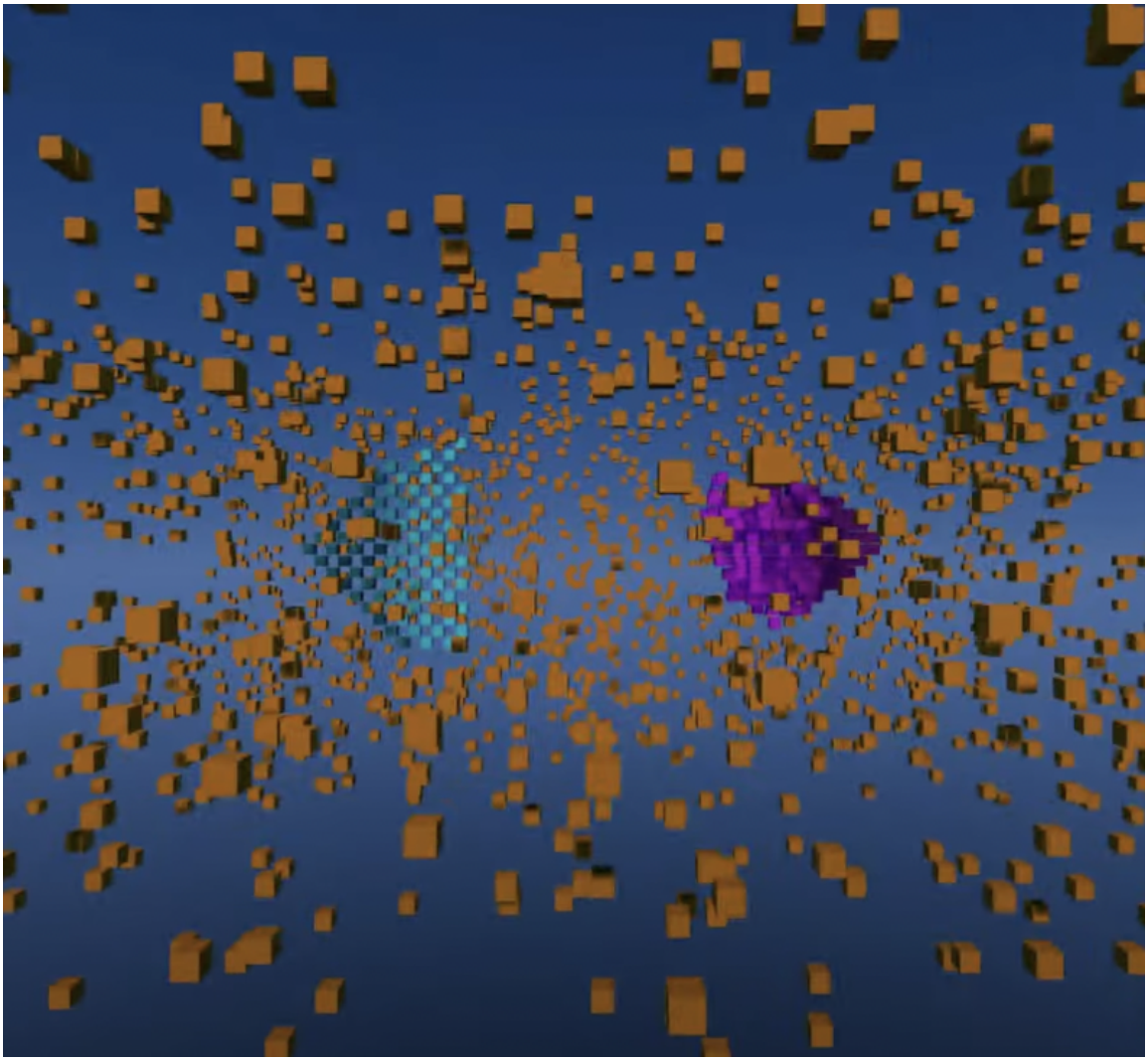
3.2.6 Example code

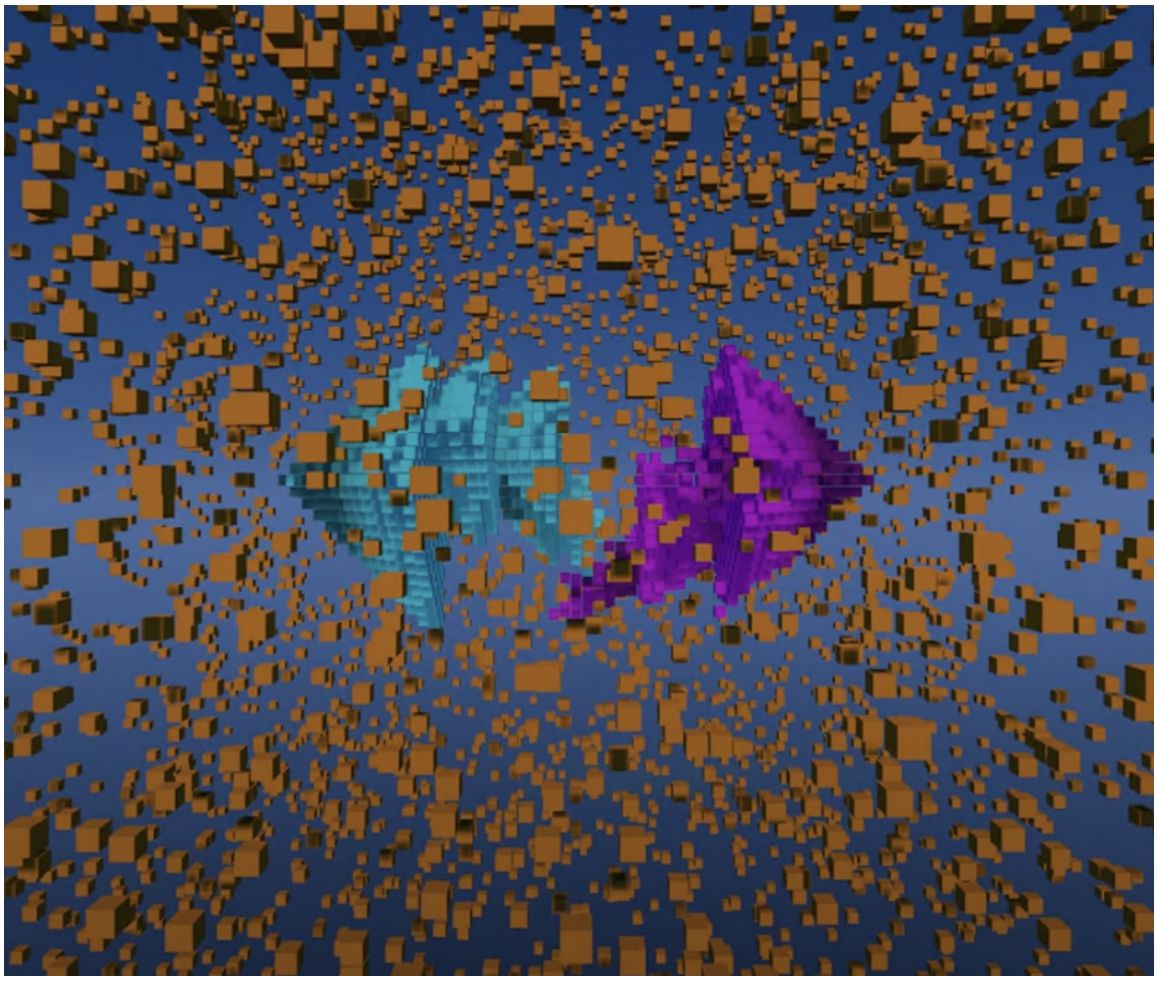
This is the implementation of a MetaLambda in the C♯ programming language. In short, it consists of the location of the lifeform in a vector based 3d spatial environment and a lambda that are the instruction of that lifeform of how to operate at each computational step. Additionally it contains an "absorb" function to describe what should happen should this lifeform encounter another one.

```
public MetaCell(MetaCoordinate coordinates, MetaLambda lambda, int complexity)
    => (Coordinates, Lambda, ComputationComplexity, State) = (coordinates, lambda,


public void Absorb(MetaCell otherCell)
{
    // merge or aggregate the lambdas
    this.Lambda = cell => otherCell.Lambda(this.Lambda(cell));
    // increase the complexity
    this.ComputationComplexity += 1;
}
```

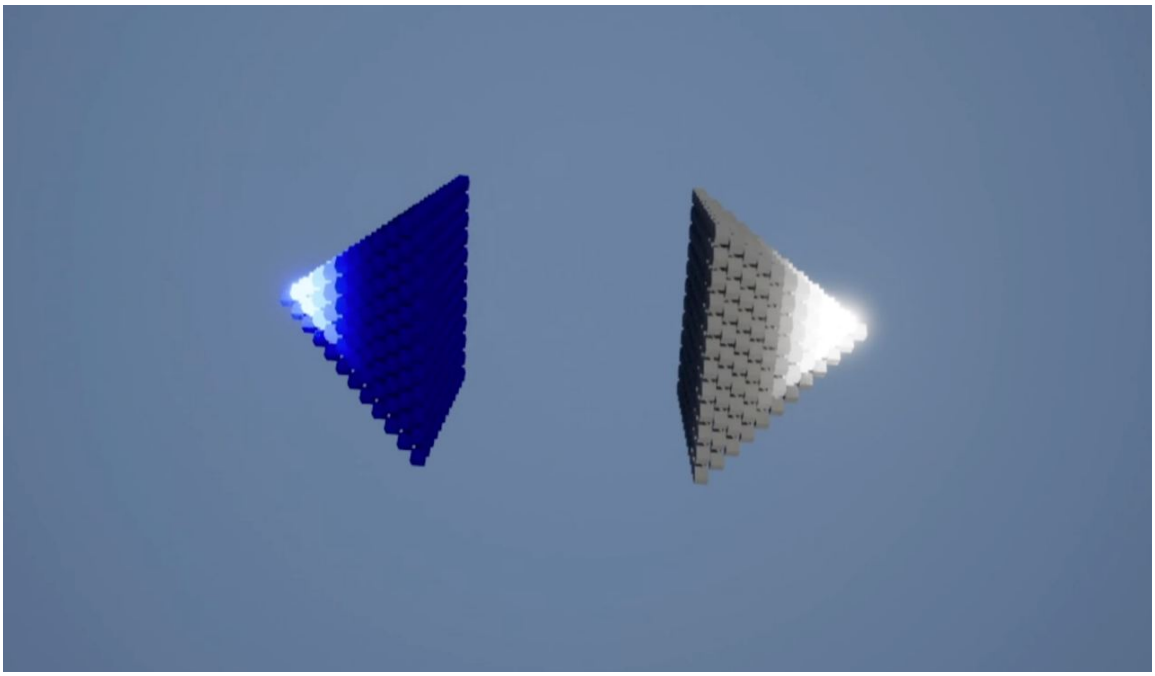
3.3 Godot engine implementation





3.4 Unreal engine implementation

3.4.1 Two lifeforms



3.4.2 Three lifeforms

3.5 Other functions

Other implemented functions include the MetaGame, which governs the state in which cells live or die, MetaLambda, the wrapper of lambda functionality, and various functions for creation and maintenance of the Grid and Coordinate space.

4 *Web3 Bindings*

Web3 primitives conveniently allow anyone to register and own a part of the overall coordinate space and, further more, own and monetize the life forms that exist there. This is issued trivially with NFT technology but, less trivially, by passing the metalambda instructions along with the initial transaction through metadata (although there are a few APIs that have agreed to integrate this). Also non-trivially, we have implemented a cross-chain Metadao with multiple levels of nesting and an automatic rollup function for the voting in lower level daos.

A DAO can also be used to govern the operating variables of the game function, such as the initial starting conditions and the evaluation criteria for each lifeform. In short, curators can decide the ability of a lifeform to continue or gain more energy, and can so on the basis of quantity (i.e. number of unique cells of that lifeform), including altering the starting conditions.

5 *Philosophical Ramifications... but is it life?*

If we define that an organism must have metabolism, reproduction, and natural selection to qualify as a life, then MetaLife may qualify. Additionally, web3 bindings, especially the ability to render an internal currency system which then, via outbound triggers, allows an advanced lifeform to secure the computational resources it needs for continued survival.

6 *Credits*

The Game of MetaLife was inspired by conversations with Peter Suber, founder of Nomic concerning an interactive 21st century version of Conway's Game of Life, which in turn was sparked by a reading of Hofstadter's book on Metamagical Themas. It is unique, in

the sense that the primary author (Joel Dietz) is a long-time artist and game designer first who kickstarted several web3 projects, including Metamask, the academic field of cryptoeconomics, and the first work on the cryptocurrencies and the law. That means that the priority of the game of metalife is, first and foremost, to make a game. In this sense, it is built on both the academic tradition of Ludology (Huizinga, Piaget, Sutton Smith), the poetic tradition of Herman Hesse, and of the world of game designers world wide. Additionally, meaningful and contributing conversations were also had with James Boyd (launch director of the Wolfram Institute), Juergen Schmidhuber, and, indirectly, Stephen Wolfram. Along the way I found the incompleted work of Von Neumann on universal constructers, which I am highly motivated to bring to completion. In this sense, the game of metalife is a game first and, as such, attempts to engage the world of hobbyists and professional researchers in the worlds of lambda calculus and cellular autonoma. It is also is an interesting tweak on the world of web3, insofar as simulations, modeling life and structured incentives were what attracted me to the space, and I have been periodically active in that space.

7 Conclusion

We have, in short, opened up a parallel universe and populated it with computational lifeforms that are able to co-evolve in unique environment . The ability of these lifeforms to evolve to higher degrees of complexity is dependent on the ability to secure the necessary amount of compute resources and to find areas of utility where they provide some benefit. Initial use cases include serving as fine art pieces, populating game worlds, and others areas where they may be useful. As a side effect we may also be able to finsh von Neumann's vision of universal constructors, evolve computing hardware, and find other forms of intelligence that are more network based.

8 Appendix: MetaLife glossary

M2 A set of logical transformations and testing framework to populate computational universes, and, moreover, test the possibility of parallel universes. Comprises all of the below. [1]

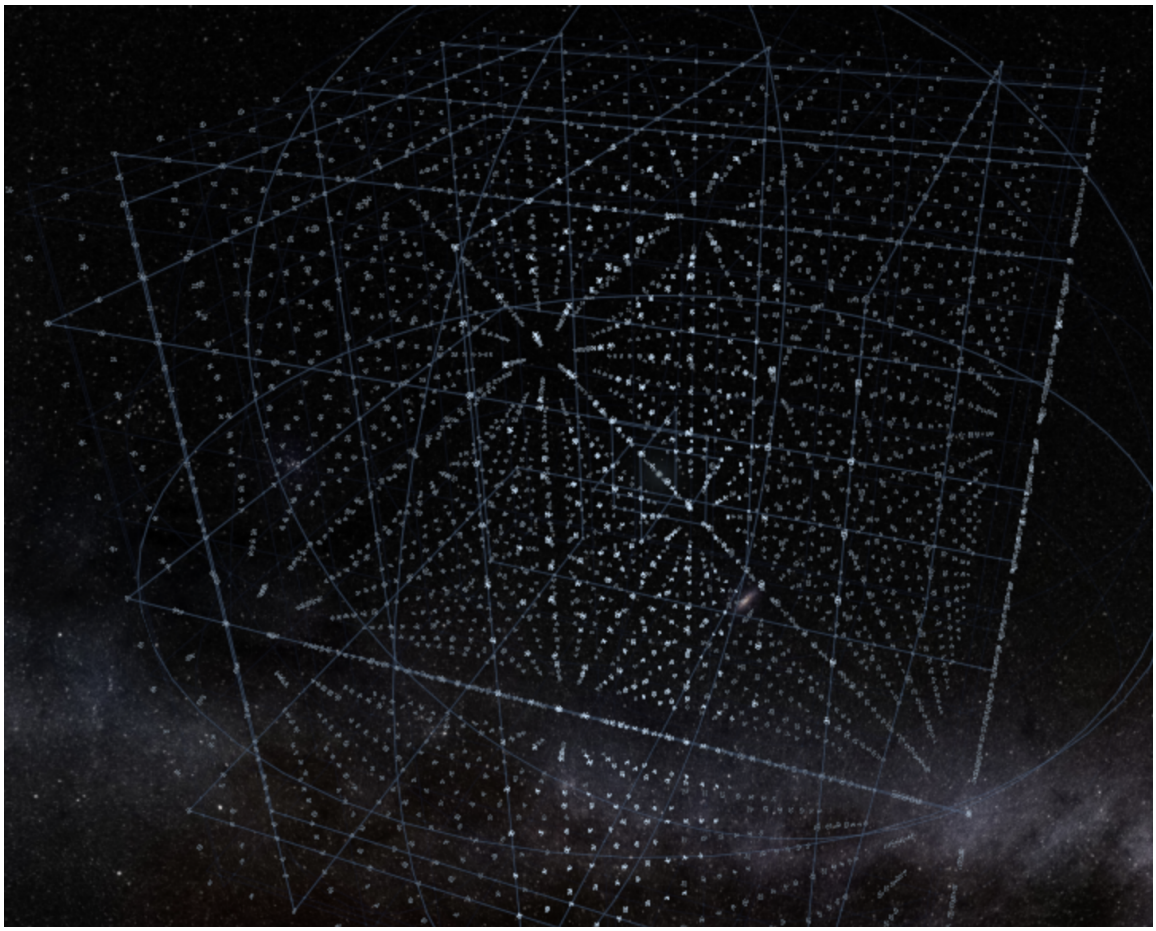
Metabrow a custom built browser for the 3d spatial web that implements metaurls and includes NFT technology.

Metadao A dao that controls a metaverse by by storing metadata related to that metaverse in a mutable format that can be updated by those members. The metametaverse is itself a metadao. Additionally, the initial implementation includes cross-chain capable metadao that allows multiple levels of nesting and decision making, allowing sub-metaverses to make decisions that have an effect on their parent metaverse.

Metalambda The definition for functions (i.e. lambda calculus) that can be deployed in multiple rendering engines that allow for the creation of MetaLife.

Metamathematics The study of sets of rules comprising mathematics and physics. It is a key building block for metalambdas. The Wolfram Institute is one of our key partners for integrating this (`https://writings.stephenwolfram.com/2022/03/the-physicalization-of-metamathematics-and-its-implications-for-the-foundations-of-`

Metametaverse the initial seeding of 3374 cubes at a distance of 1-7 from the central star calculated by an offset function and governed by a DAO of the same name. which determines the spatial configuration and math/harmonics of related metaverses.



Metamodel An abstract mathematical model that can be used to compose storylines and complex behaviours using metaverse objects and metaurls (i.e. a composible state machine based questing framework)

Metanomic Metanomic is a variant of the nomic game set, whereby multiple nomic games can created and run through a common metaruleset and framework, including a notion of mutation of the ruleset.

Metaphysics Sets of mutable physics rules that can be deployed in aprticular game engine, including the possibility of testing the feasibly of any particular parauniverse.

Metaship A spaceship (issued as an NFT) that acts as a cursor into the spatial web.

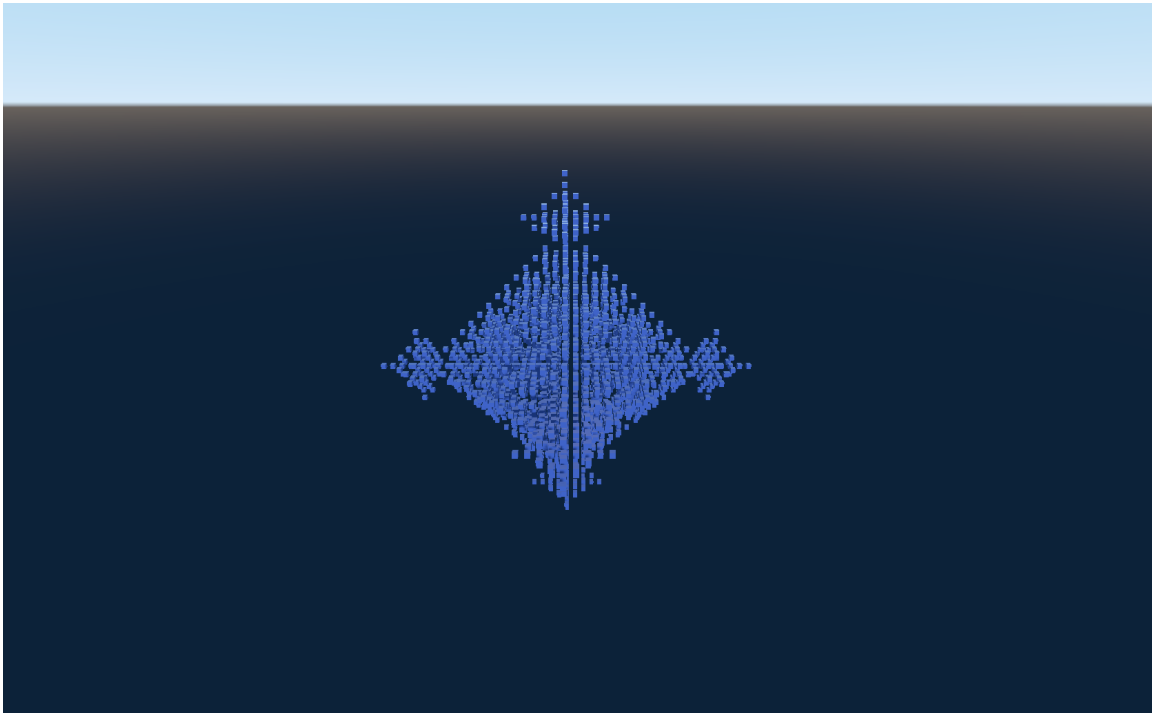
Metaurl is a url in the format /2.3.3.3/3.3.3.4/i/me.vox that describes the location of an object in 3d space. It can be merged with more merged readable formats

9 *Appendix: Evolution of a Fractal Lifeform*

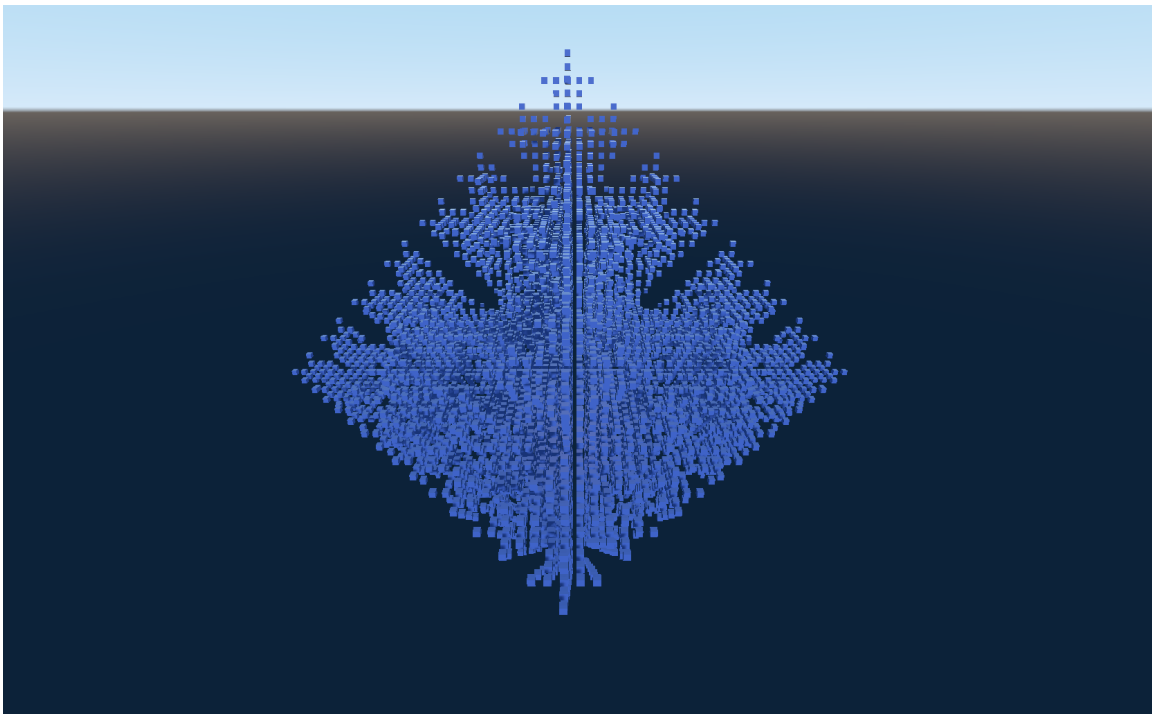
With certain starting conditions we can now simulate certain lifeforms starting with crystalline objects. This is a simpler version than described in the paper but illustrates how a single lifeform can evolve complexity.

9.1 *State after 10 computational steps*

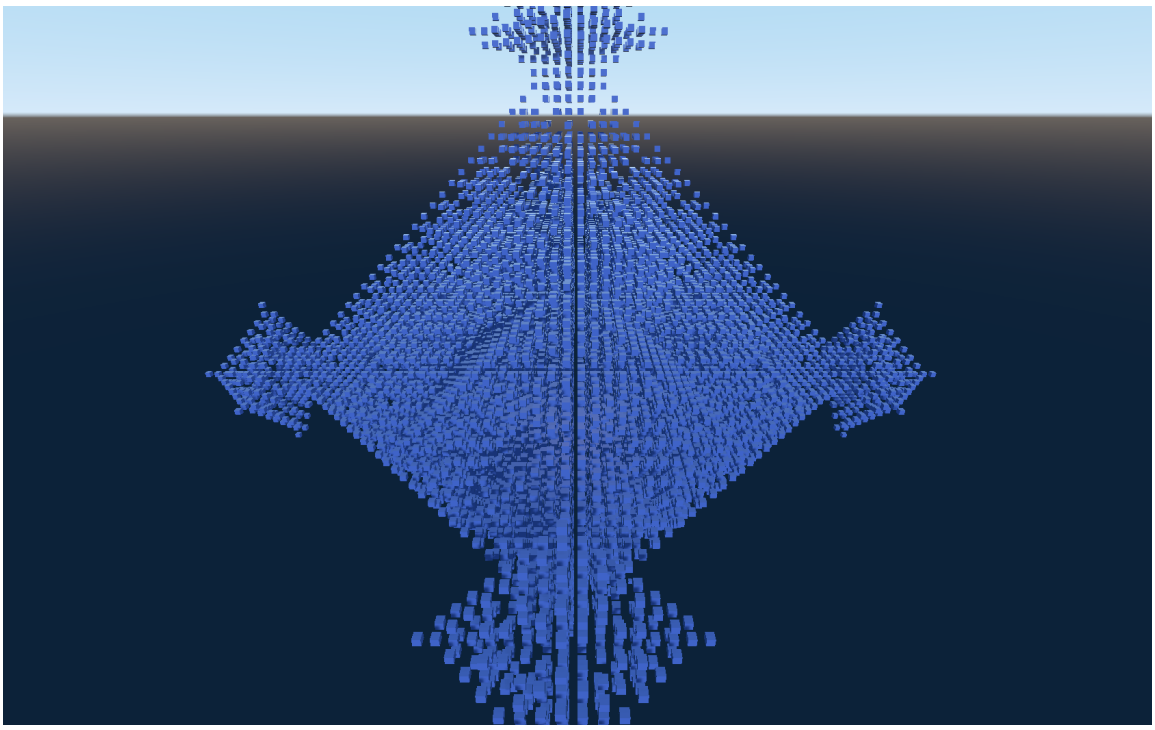
9.2 *State after 20 computational steps*



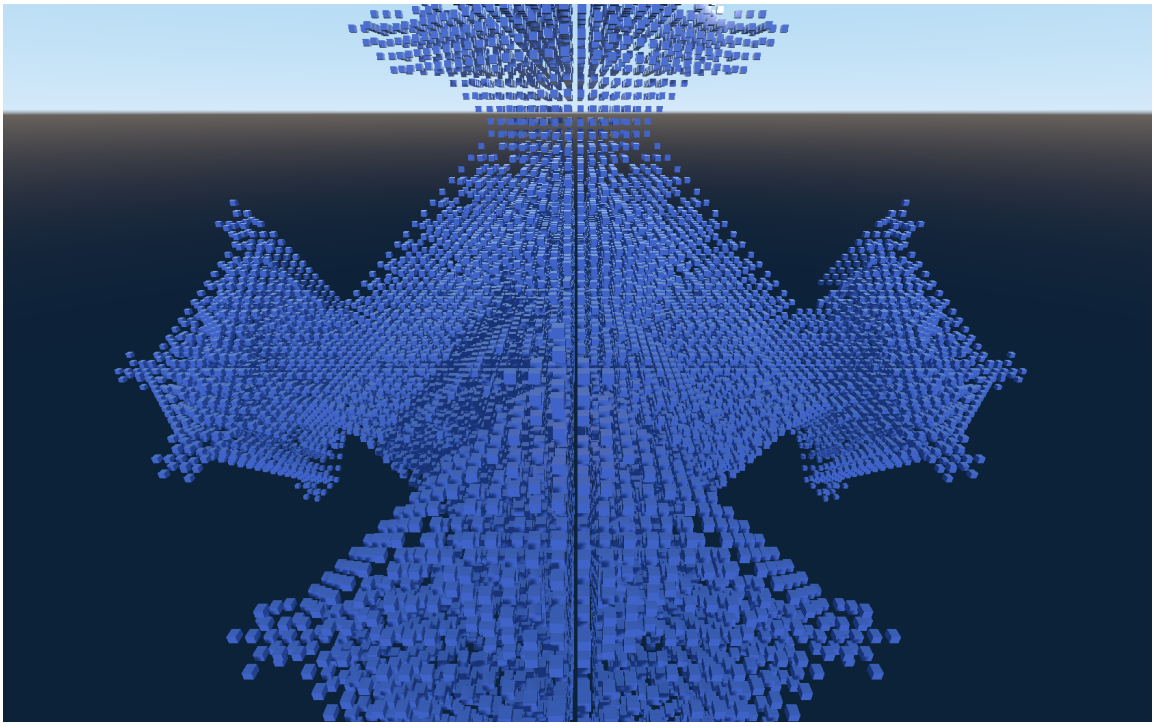
9.3 *State after 30 computational steps*



9.4 *State after 40 computational steps*



9.5 *State after 50 computational steps*



9.6 *Unreal Engine Implementation*