GYMNASIUM JANA KEPLERA

Parléřova 2/118, 169 00 Prague 6

Spatial Hypergraph Evolution

Graduation Thesis

Author: Kryštof Mitka

Class: 4.A

School Year: 2020/2021

Subject: Computer Science

Supervisor: Šimon Schierreich

Prague, 2021

GYMNASIUM JANA KEPLERA
Kabinet informatiky

ZADÁNÍ MATURITNÍ PRÁCE

Student:	**Kryštof Mitka**
Třída:	**4.A**
Školní rok:	**2020/2021**
Platnost zadání:	**30. 9. 2021**
Vedoucí práce:	**Šimon Schierreich**
Název práce:	**3D Vizualizace evoluce hypergrafu**

Pokyny pro vypracování:

Vytvořte nativní aplikaci pro operační systémy Linux a Windows, která v čase vizualizuje evoluci 3D hypergrafů Stephena Wolframa. Aplikace bude zejména a) poskytovat grafické rozhraní pro možnost změny základního nastavení - vizualizovaného grafu, b) možnost posunu v čase renderování vizualizovaného grafu a c) nabízet export videa evoluce z vybraného úhlu kamery.

Doporučená literatura:

[1] The Wolfram Physics Project: Finding the Fundamental Theory of Physics [online]. The Wolfram Physics Project. Dostupné z: https://www.wolframphysics.org/.

[2] MARTIN, Robert C. Design Principles and Design Patterns. www.objectmentor.com, 2000. Dostupné z: https://fi.ort.edu.uy/innovaportal/file/2032/1/design_principles.pdf.

[3] FOWLER, Martin. Patterns of enterprise application architecture. Boston: Addison-Wesley Professional, 2003. ISBN 978-0321127426.

URL repozitáře:

https://github.com/KrystofM/3d-hypergraph-evolution

vedoucí práce *student*

V Praze dne 29. 10. 2020

Declaration

I declare that I have prepared my work independently and have used only the sources and literature listed in the list of bibliographic records. I have no objections to making this work available in accordance with Act No. 121/2000 Coll. on Copyright, on Rights Related to Copyright and on Amendments to Certain Acts (Copyright Act) as amended.

In Prague April 7, 2021

Kryštof Mitka

Acknowledgment

I would like to thank my professor and supervisor Šimon Schierreich for all the consultations and the hard work he had put in this year.

Abstrakt

Evoluce hypergrafů Stephena Wolframa skrývá potenciál a jednoduchou krásu, ale pro širší netechnickou veřejnost neexistuje v podání, které by bylo dostatečně jednoduché a interaktivní. Účelem tohoto projektu je vytvořit přístupnou, snadno použitelnou a interaktivní prostorovou vizualizaci vývoje hypergrafů takovým způsobem, aby kdokoli na světě mohl hned začít zkoumat tyto modely, aniž by musel projít technickou dokumentací výzkumu a bez požadavku stáhnutí speciálního softwaru. Projekt byl vypracován za použití technologií jako WebGL a ThreeJS společně s algoritmy na vizuální vykreslování a zároveň s důrazem na praktické uživatelské rozhraní a bezproblémovou interakci. Výsledkem je uživatelsky přívětivý web poskytující prostorové vykreslení vývoje hypergrafu jak pro desktop, tak pro mobilní publikum.

Klíčová slova

evoluce hypergrafů, prostorové zobrazení hypergrafů, interaktivní hypergrafy, stephen wolfram

Abstract

Acknowledging the potential and the simple beauty of Stephen Wolfram's hypergraph evolution, but the lack of tangibility and interactivity for the wider non-technical public there is. The purpose of this project is to create an accessible, easy to use and interactive spatial visualisation of the hypergraph evolution so that anyone in the world can quickly interact and get a deeper understanding of these models without having to go through much of the research or having to download special software first. Using technologies like WebGL and ThreeJS along with visual rendering algorithms while keeping in mind a practical user interface for seamless interaction. The result is a user-friendly website delivering spatial renders of the hypergraph evolution to both desktop and mobile audience.

Keywords

hypergraph evolution, spatial hypergraph, 3d hypergraph, interactive hypergraph, stephen wolframs

Contents

1. Theoretical part

When the Wolfram Physics Project was released most of the public were able to see a picture or two of some evolved hypergraphs and were left staring at something untangable. To understand more about the Project you could dive deep into the 1000+ pages of resources and spend a few hours trying to render your first hypergraph rule. As a Cellular Automata enthusiast I was among those who skimmed through the resources in hope of finding some cool interactive spatial renders of different rules, to my disappointment I was left staring at few more static pictures of some other hypergraphs. In my search of some easy - not having to download anything - quick to use interactive application, where I could closely explore the rules proposed by Wolfram, I was left stranded.

Therefore, my goal with this project was to make a super easy interactive application that anyone can quickly play around with - without first knowing what is a Hypergraph or how Cellural Automata work. Simply just to watch and appreciate how complexity arises from simplicity.

1.1 Graphs vs Hypergraphs

In this section we will explore the difference between graphs and hypergraphs, their different types and definitions in terms of an abstract data type.

1.1.1 Undirected simple graph

First we define a graph in terms of an abstract data type. A graph consist of a set of **vertices** (also called **nodes** or **points**) and of a set of **edges** (also called **links** or **lines**), every edge consists of a pair of vertices. Now a simple definition of a graph as a pair $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ where:

- $\mathbf{V} \rightarrow$ set of **vertices**
- $\mathbf{E} \subseteq \{\{x,y\} \mid x,y \in \mathbf{V} \text{ and } x \neq y\} \rightarrow$ set of **edges**

For simplicity we do not allow for loops $x \neq y$ (an edge connecting the same vertices), we do not allow for multiple edges (two or more edges connecting the same vertices) and we do not care about the ordering of vertices in an edge. We would call this type of graph **undirected simple graph**. Notice that an edge by definition only connects a pair of vertices $\{x,y\}$.

photo

1.1.2 Undirected simple hypergraph

We can think about hypergraphs as a generalization of graphs. As noted, edge of a graph only connects two vertices, however, edge of a hypergraph can connect any number of vertices, therefore we call it a **hyperedge**. Formally we define hypergraph as a pair $\mathbf{H} = (\mathbf{X}, \mathbf{E})$ where:

- $\mathbf{X} \rightarrow$ set of **vertices**

- $\mathbf{E} \subseteq \mathcal{P}(\mathbf{X}) \setminus \emptyset \rightarrow$ set of **hyperedges**

Notice that the set of hyperedges is a subset of the powerset $\mathcal{P}(\mathbf{X})$ without the empty set. By that definition we do not allow for loops, multiple edges and direction of hyperedges, therefore we are defining it as an **undirected simple hypergraph**.

This is an important type of hypergraph as it will be used in the visual spatial implementation.

1.1.3 Directed multihypergraph permitting full and partial loops

This type of hypergraph is basically the most generous you can think of. First we care about the direction in which the hyperegdes point. Second we allow for multiple edges joining the same vertices and last we allow full and partial loop:

- Hyperedge Full Loop: $\{a, a, a\}$
- Hyperedge Partial Loop: $\{a, a, b\}$

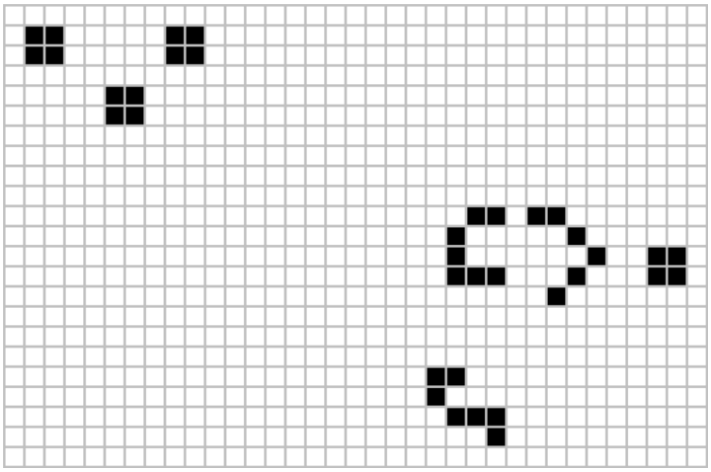
I will not go deep into the formalities of defining exactly this type of hypergraph. Try to remember the features as this will be the backbone of our calculations.

1.2 Cellular Automata

In this seciton I would like to give a simple introduction into a class of programs called Cellular Automata. In the last section we focused on the **Hypergraph** part of the project, here we will be focusing on the **Evolution** part of the project. It is imperative to understand this class of programs as some of the concepts introduced here are the underlying features of Wolfram Physics Project.

1.2.1 Game of Life

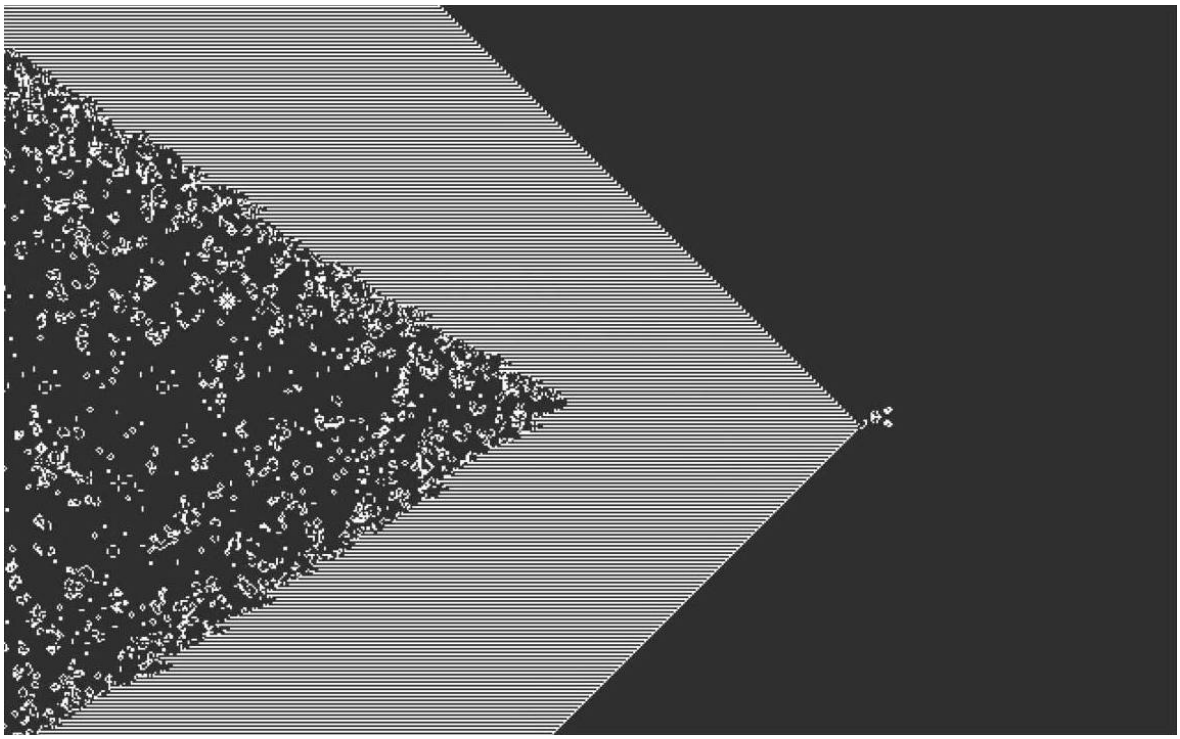
I would like to start with probably the most famous example of Cellular Automata called **Jon Conway's Game of Life**. Imagine we have a grid of square cells each of which could either *live* or be *dead* (*black* or *empty*, respectively).



Now we define 4 very simple rules for each of the cells by the state of their neighbours.

1. Any live cell with fewer than two live neighbours dies.
2. Any live cell with two or three live neighbours lives.
3. Any live cell with more than three live neighbours dies.
4. Any dead cell with exactly three live neighbours becomes a live cell.

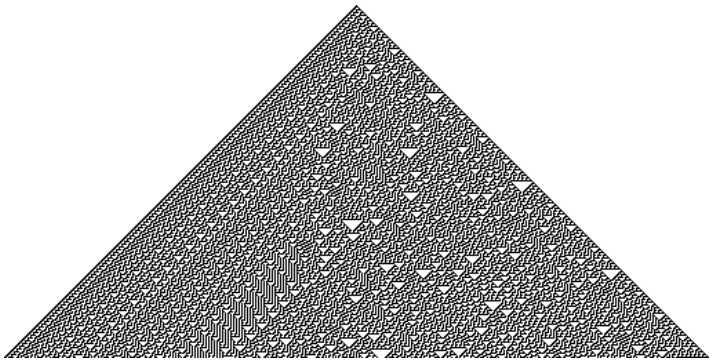
These rules are then applied to all cells from left to right, top to bottom. The application of the different rules to all cells once is called a **step in evolution**. The rules might seem very easy, however, given the right initial positions of live and dead cells **complexity arises from simplicity**.



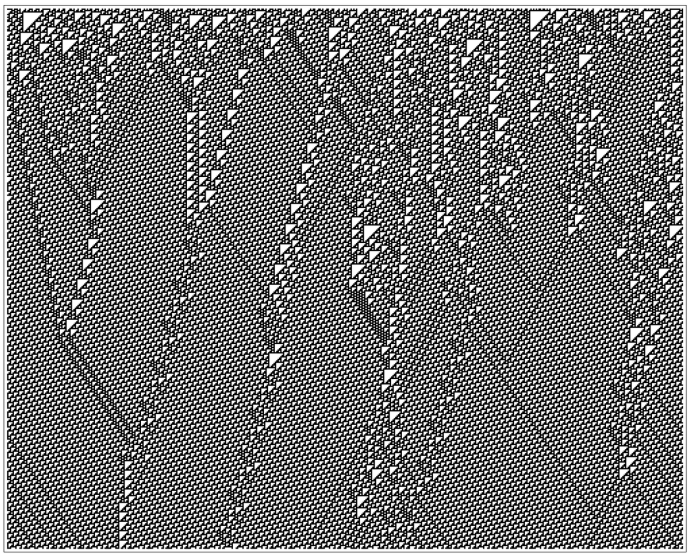
1.2.2 Emerging Complexity

The phenomena of emergent complexity from simple rules is what drives the interesting nature of Cellural Automata. It is just recently that this type of program has been put to use in real world systems such as biological and chemical systems. One of my all time favorites cellural automata, where complexity arises, are the rules 30 and 110 from Stephen Wolfram Elementary Cellular Automaton, which he started working on way back in time with Richard Feynman.

Rule 30 was the first where Wolfram realized the full complexity that was produced. Feynman famously asked him how did he know that the rule 30 was going to produce such results. Wolfram told him that he did not know, it was only achieved by computing all of the possible rules.



Rule 110 is among the cool rules because in 2004 Matthew Cook proved that is it Turing complete, meaning it is computationally universal and can be used to simulate any Turing machine.



In the computational universe there is a lot of different cellural automatons and rules where complexity occurs. It is important to understand this phenomena to understand the Wolfram Physics Project relations to physics.

1.3 Wolfram Physics Project

We have covered the bare minimum that is needed to understand the Wolfram Physics project. Understanding the definition of **Hypergraphs** and **Evolution** in terms of a series of steps, where we apply rules everywhere possible as in Cellural Automata. In this section we will be combining the two **Hypergraphs** and **Evolution** as the fundemenatal stones of Wolfram Physics Project. Most of the rules and text in this section will be directly stated from the technical part of Wolfram Physics Project, therefore if you want a more deep dive I would recommend following information from there. Here I will provide a simple summary of what is needed.

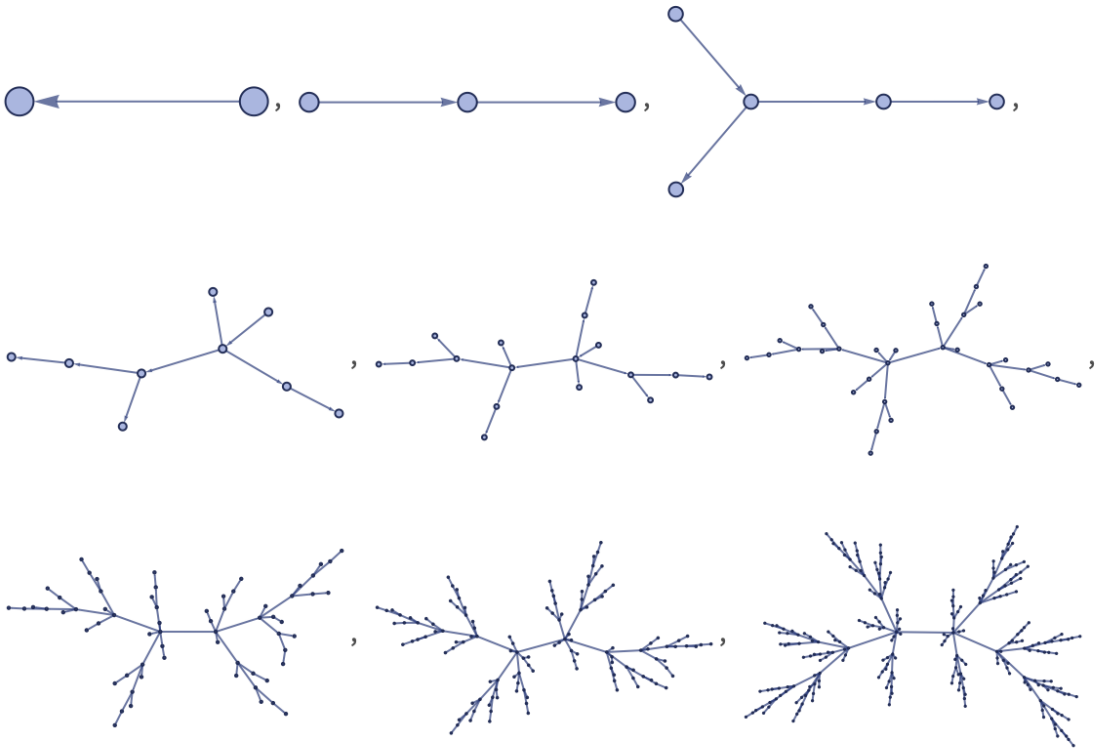
1.3.1 Evolution of Hypergraphs

We will start with simple structures. Note that the evolution rules we will be defining are defined among k-nary hypergraphs. Wolfram defines simple transition rules for the hyperedges that will be applied everywhere a match occurs. Lets start with one of simplest rules for a graph:

$$\{\{x, y\} \to \{x,y\}, \{y,z\}\}$$

"Here x, y and z stand for any elements. (The elements they stand for need not be distinct; for example, x and y could both stand for the element 1.) The rule states that wherever a relation that matches {x,y} appears, it should be replaced by {{x ,y},{y,z}}, where z is a new element. So given {{1, 2}} the rule will produce {{1,2},{2,z}} where z is a new element. The label for the new element could be anything—so long as it is distinct from 1 and 2. Here we will use 3, so that the result of applying the rule to {{1,2}} becomes: {{1,2}, {2,3}}"

Applying this rule a few more evolution steps yields a discrete structure:



Along these lines we could define all sorts of different rules adding more edges to transformation:

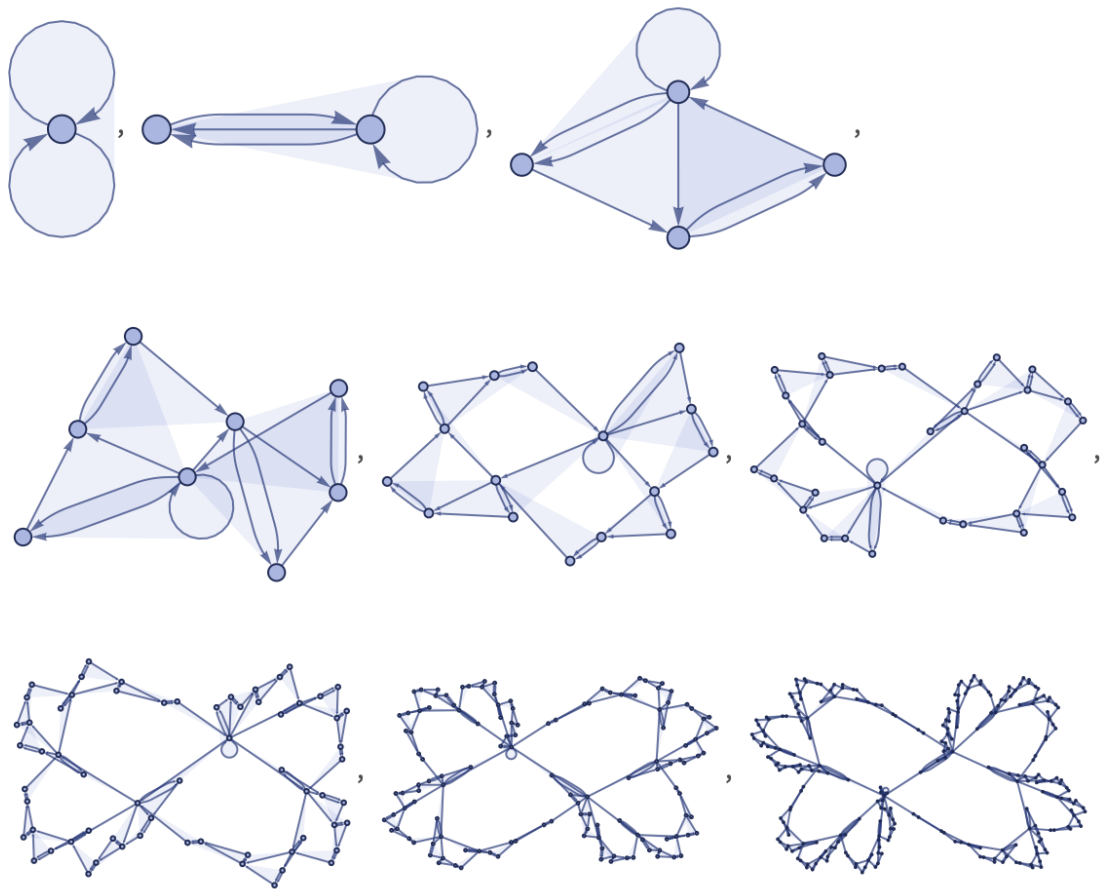
$$\{\{x, y\} \to \{x,y\}, \{y,z\}, \{z,x\}\}$$

or

$$\{\{x, y\} \to \{y,z\}, \{y,z\}, \{z,x\}, \{z,x\}\}$$

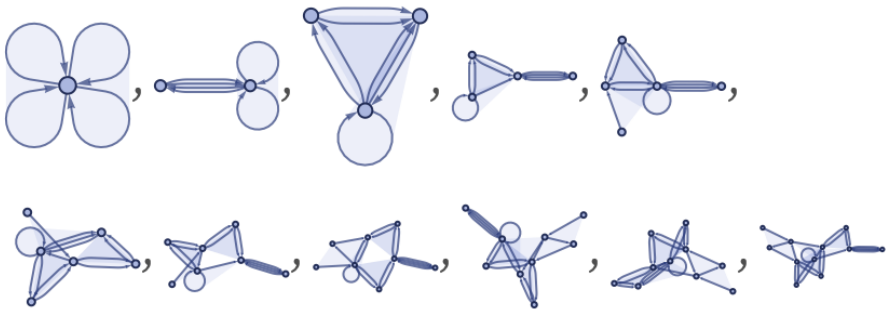
We can also start applying rules on hypergraphs. We allow for direction, multiedges, full and partial loops, therefore we are using the aforementioned **directed multihypergraph permitting full and partial loops**.

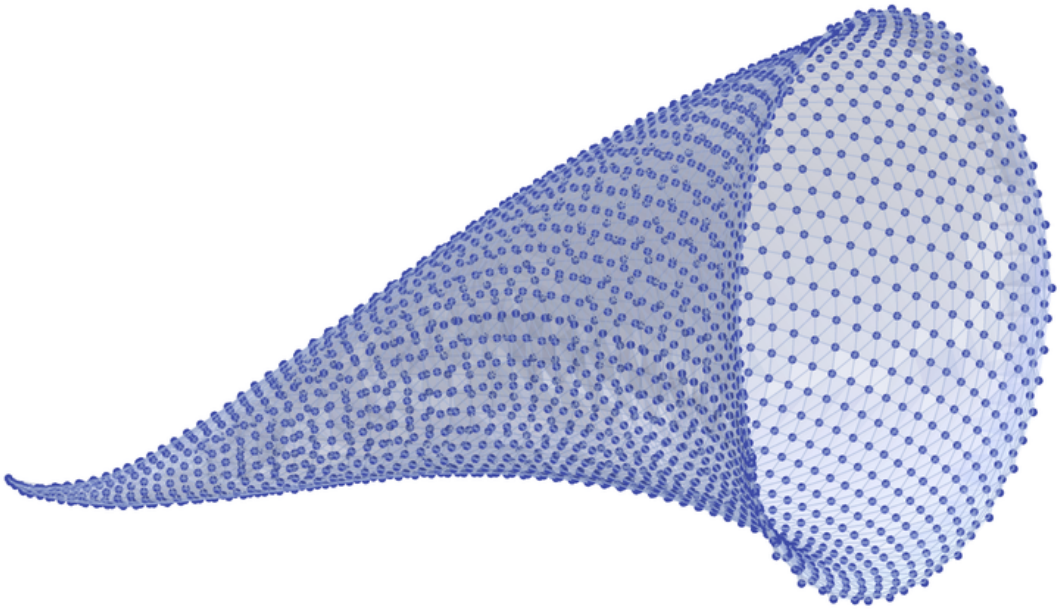
$$\{\{x, y, z\} \to \{x,y,w\}, \{y,w,z\}\}$$



Finally we can apply rules on pairs of hyperedges, some of which yield complex geometric structures:

$$\{\{x, x, y\},\{x, z, u\} \to \{u,u,z\}, \{y,v,z\}, \{y,v,z\}\}$$





1.3.2 Potential Physics relations

Stephen Wolfram made several big claims about the potential relations to Physics, claiming that he has derived Einstein's Theory of Special Relativity, his models inherently support the Quantum Mechanics and the list goes on. Wolfram believes the emergent complexity from his models is one of the key factors when discussing the relation to Physics. His claims, however, have been received with a slight kickback from the scientific community as he has not gone through a classic approach of peer review. Personally I find the idea that one of these rules would be the underlying rule of the universe beautiful, but right now there is no clear evidence that would support that idea. I would recommend reading more into this area of research on the Wolfram Physics website.

2. Implementation

In this chapter I will focus on the technologies that I have used to create the application along with some difficulties I had to face and peaks I have successfully overcome.

In the light of my goal to create a very simple and more importantly - not having to download - type of application, I have decided to create an application for the **web**. This was one of the first decisions I had to make and in retrospect I do believe it was the right decision, since the start of interacting with the Spatial Hypergraphs is nearly instant from typing the website's address.

2.1 Technological stack

2.1.1 WebGL, ThreeJS

Since this project is about 3D modelling on the web, there is probably no better combo than the classic WebGL and powerful ThreeJS. I will get more into how exactly I have implemented the 3D modelling later on in another section, however, it is important to mention these two technologies, as they are fundamental to spatial rendering. WebGL is a JavaScript API that allows GPU-accelerated usage of physics and image processing and effects as part of the web page canvas. ThreeJS is a framework build on top of WebGL, using it as a means of rendering, very clean and easy to use framework that has a large community and performs fantastically.

2.1.2 TypeScript

Typescript is a build-on to JavaScript, meaning it compiles to JavaScript, but pure JavaScript is also supported. The great things about it are that it is an open-source language, adding a lot of types, type checking, basically it makes for a far better language than JavaScript. I will be using it, since it produces more readable code, than JavaScript alone.

2.1.3 Svelte

Svelte is new approach to reactivity, instead of having to do a lot of work in browser like React or Vue, Svelte compiles your code during build into a the purest forms of JavaScript update events. This approach yields incredible speed results as there is no Virtual DOM, so no diffing during update events is needed. At first I have thought that I would be using classic React framework to handle reactivity in my application, but since I have always wanted to tryout Svelte at some point I thought it would be a great idea to do so here.

2.1.4 Sass

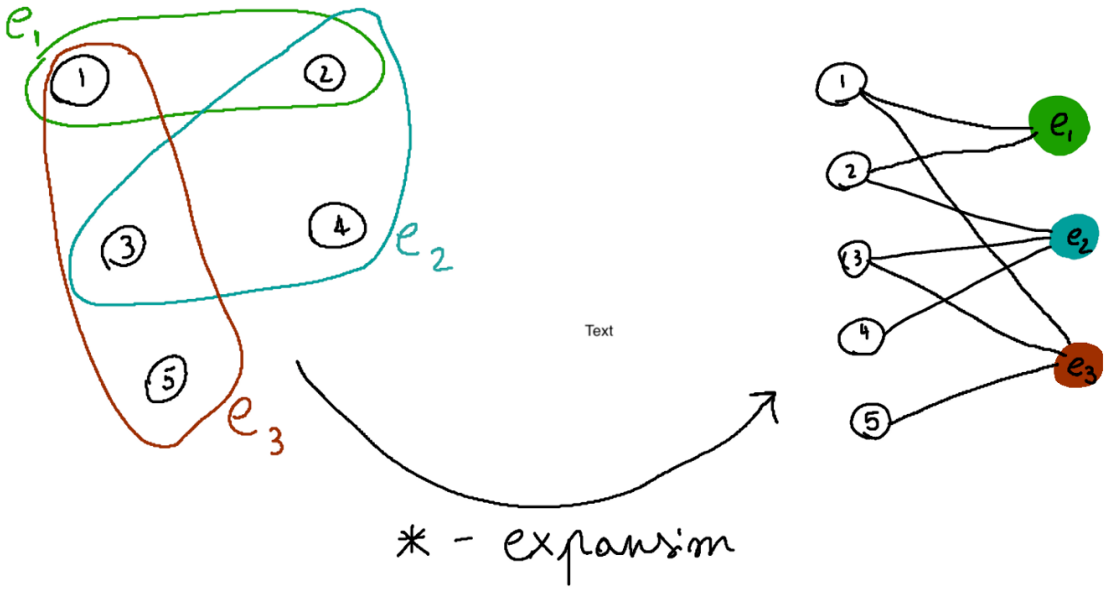
Sass is a preprocessor language that is complied to CSS. Sass provides an array of added functionalities like mixins, variables, loops, function and much more. It also gets rid of brackets and semicolons in code and allows for combining selectors. Overall it is more functional than classic CSS, which is the main reason I have chose to work with it in this project.

2.2 Hypergraph Spatial Visualisation

In this seciton I will go through the way I settled on visualisating hypergraphs in space. At first I was quite overwhelmed by the amount of different options when researching this problem. There is no clear way to visualise a hypergraph in 3D space. The main problem comes to the fact that depending on the maximum number of vertices that occurs in one of the hyperedges in a specific hypergraph the optimal space dimension for embedding this hypergraph changes. In other words, the more vertices in hyperedges, the more dimensions you need to optimally visualize the whole hypergraph. When the number of vertices in an edge is two, you can use a line to connect them both, when the number of vertices in an hyperedge is three, you can use a plane to connect the three of them, with 4 you can use space and so on.

2.2.1 Star expansion

Luckily after some more search I have found what is called a star expansion algorithm. This is an algorithm that creates a new vertex for each edge (optimally for each edge with 3 or more vertices) and connects all the vertices in the edge to this vertex. Here is an illustration (the right side is an Eulers diagram of hypergraph):

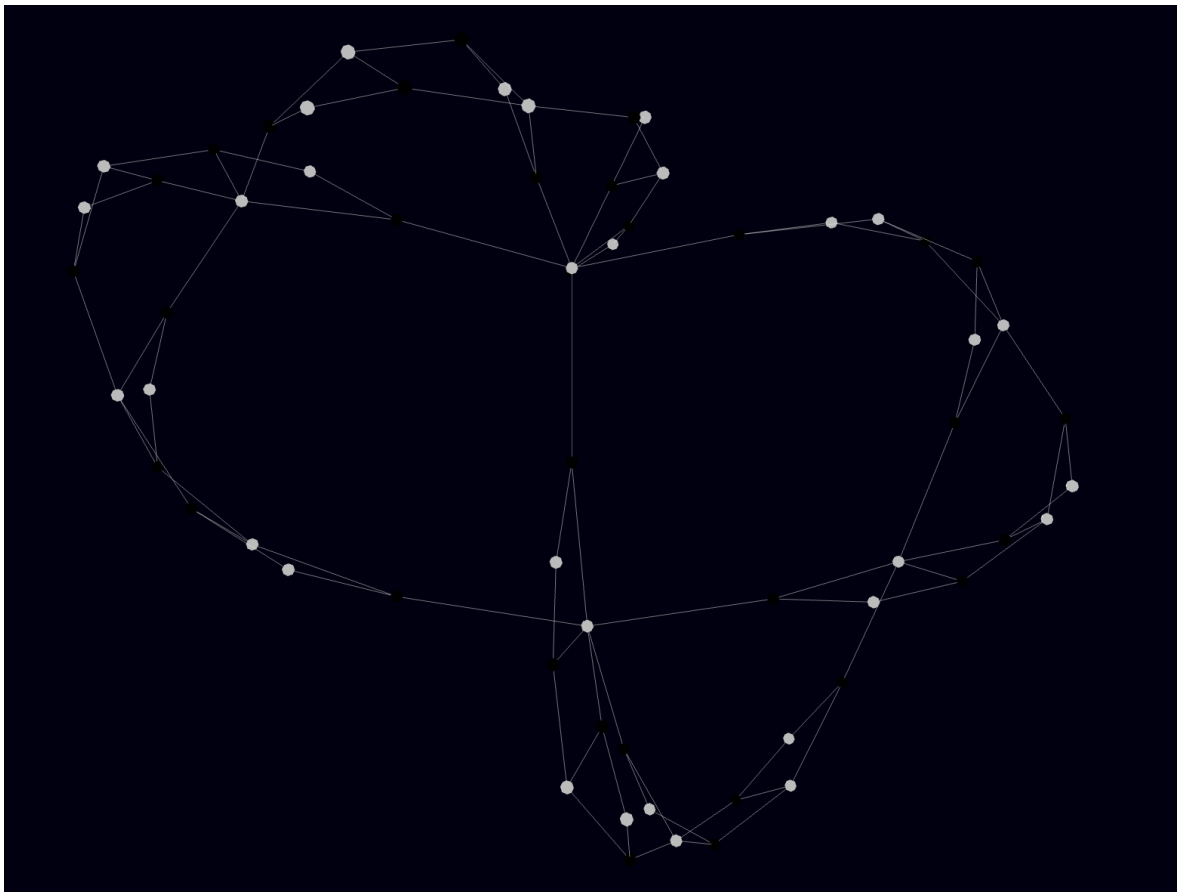


This algorithm allows **transforming a hypergraph to a bipartite graph** and as a result of that we are able to pleasantly model hypergraphs in space.

2.2.2 Force-directed graph drawing

Having an algorithm that transforms a hypergraph to graph I was able to choose from a variety of spatial drawing algorithms. I have found a library called **3D Force-Directed Graph** that belongs to the Force-directed graph drawing class of algorithms, using methods like velocity Verlet integration to render vertices in space.

Combining **star expansion** and **3D Force-Directed Graph** I was able to get first Spatial Hypergraph. I have put the **extended edges as black**, so they are not able to be seen.



Looks pretty **sick**.

2.3 Evolution Rules Mechanics

Now that I had successfully got a Spatial Hypergraph it was time for the Evolution part of problem.

2.3.1 Definitions

First I have created a class for `Hypergraph`, providing simple features. Second I have made an interface for `HypergraphRule`, which includes `name`, `optimalInitialPositions`, `optimalTicksAmount` and an `apply` method that takes in a hypergraph and returns another hypergraph that has been transformed. Every new rule then implements this interface. I have purposefully seperated this

logic from the app itself as it only computes the necessary steps for Hypergraphs and the app then does rendering and UI controls. This way I have implemented 10 different rules.

2.3.2 Rendering/UI

The app itself takes all the defined rules from the index and puts them into UI. One of the parts is the `GraphRendering` here I implemented the star expansion algorithm and also creating an interval that continually calls the apply method from `HypergraphRule`. Second important part is the controls of `GraphRendering` and everything else. Here I handle all the important UI like the **evolution progress bar** and **pause/start** option for each of the rules. Therefore you can stop graph in evolution and explore the specific step of evolution. Contorls also give you the option to turn off/on autozoom along with getting rid of them by toggling C. This is a great setup for when a user would like to record a specific rule from a specific camera angle.

There was not a lot of problems while implementing this setup. Maybe the only one worth mentioning is the weird nature of Svelte. From trying it out I have to say I do not like this default approach of global store singletons as a means of controlling the reactivity. Therefore I have started seeking different ways of wrapping stores inside of custom classes or creating custom stores.

2.4 Deploying to Web

This is a short section about the way this project was deployed to the internet. For all of it I have used the Amazon Web Services.

2.4.1 S3 Bucket with CloudFront

Built the project and uploaded the files to a S3 bucket on AWS called *3dhypergraphevolution.com* and allowed for static website hosting also created a *www.3dhypergraphevolution.com* bucket that points to the first one. On Cloudfront I have created a distribution for both of these buckets and added a certificate to provide secure connection.

2.4.2 AWS Route 53

Bought the domain `3dhypergraphevolution.com` here and created a hosted zone. Added two A records that point to the CloudFront distributions created in the step before.

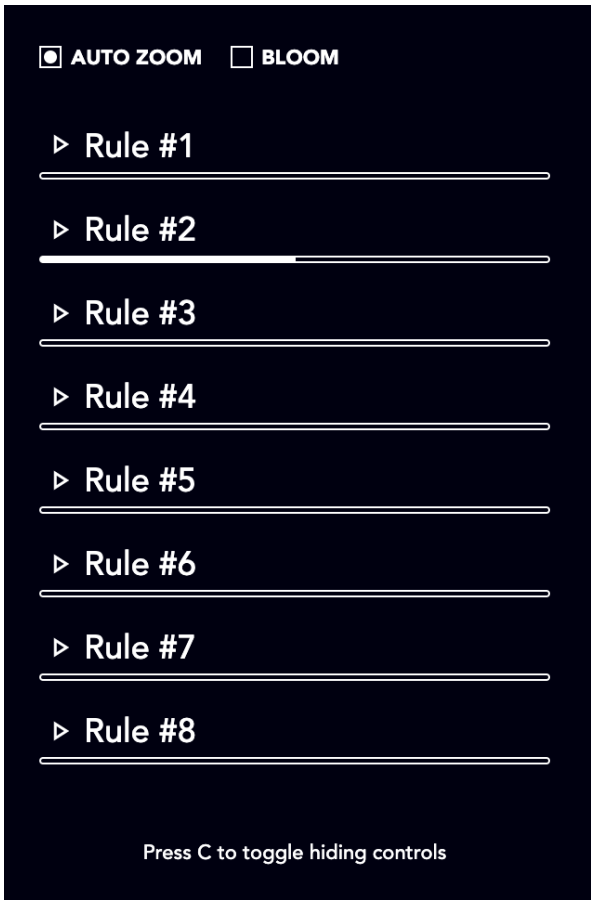
3. Technical Documentation

Since it is a web application you can start interacting with the project right away by just going to 3dhypergraphevolution.com and testing it out. In this section I will also go a little deeper for those who want to contribute.

Poslední kapitola obsahuje informace o tom, jak projekt, který v rámci maturitní práce vznikl, nainstalovat, spustit a používat.

3.1 Controls

The overview of controls:



To toggle functions `autozoom` and `bloom` just click on the label or the checkbox. The white dot means it is allowed:

You can see how far a rule is in evolution on the `progressbar` under the name of rule.

To stop a rule from playing click on the pause button next to the name of the rule.

To hide controls toggle the character C on your keyboard.

3.2 Running Locally

First clone the repository from github to desired location:

```
git clone git@github.com:KrystofM/3d-hypergraph-evolution.git
```

After cloning the repository run:

```
npm install
```

To start in localhost run:

```
npm start
```

To build for production run:

```
npm run-script build
```

3.3 Adding a Rule

1. First add a new file to *src/hypergraphs/rules*.
2. In the file create a class that implements interface `HyperGraphRule` and implement all methods and variables needed.
3. Once done with implementing, add the class to *ALL_RULES* in *src/hypergraphs/rules/index.ts*.
4. Now reload and the rule is added!

Conclusion

From knowing very little about the Stephen Wolfram Project I was able to create a interactive spatial hypergraph evolution application that can be loaded basically instantly. I have also created a pretty good looking UI that fits the project and overall look. You can now interact with hypergraphs instantly anywhere from the world, whether you are on a desktop or a mobile device as the responsivity of the website works great. More advanced users can even very easily clone and start the project along with adding their own new rule.

On a different note. As it is with every software project, there is an ever expanding array of features and optimizations that could be implemented. There could be a little more freedom in moving along the timeline with specific hypergraph rules as described in the point b) of the assignment. Right now to get to a certain step, you have to start the rule again and pause in a desired step. I could also implement native recording of the hypergraphs the way described in the point c), however, you can just hide the controls and setup camera to a certain position (with auto zoom off) and start your operating system's screen recording, which yields the same functionality (in some ways way better than a native as you can setup the proportions and format). I think it is important to question constrains and have a mindset of knowing that the constrains are always to a certain degree wrong as the opposing argument would be that they are perfect. One fun feature I would like to add in future is playing some music along with rendering the hypergraphs along with having an autoplay feature that plays all rules after each other.

In conclusion, I believe the whole project was a success and I would like to continue on with its development into the future. I will try to send it out to the Stephen Wolfram Physics community and see if it gets any response.

List of used literature

[] *Cellural Automaton — Wikipedia, The Free Encyclopedia*. URL: https://en.wikipedia.org/wiki/Cellular_automaton.

[] *Euler Diagram — Wikipedia, The Free Encyclopedia*. URL: https://en.wikipedia.org/wiki/Euler_diagram.

[] *Force Directed Graph — Wikipedia, The Free Encyclopedia*. URL: https://en.wikipedia.org/wiki/Force-directed_graph_drawing.

[] *TypeScript documentation*. URL: https://www.typescriptlang.org/docs/.

[] *WebGL — Wikipedia, The Free Encyclopedia*. URL: https://en.wikipedia.org/wiki/WebGL.

[Ang] Alessandro Angioi. *How to visualize hypergraphs with Python and networkx — The Easy Way*. URL: https://towardsdatascience.com/how-to-visualize-hypergraphs-with-python-and-networkx-the-easy-way-4fe7babdf9ae.

[Wol02] Stephen Wolfram. *A New Kind Of Science*. 2002.

[Wol20] Stephen Wolfram. *A Project to Find the Fundamental Theory of Physics*. 2020.