Lyn: Live Coding Artificial Life Simulations

Major Qualifying Project

Written By:

EVELYN TRAN

Advisor:

CHARLES ROBERTS

A Major Qualifying Project
WORCESTER POLYTECHNIC INSTITUTE

Submitted to the Faculty of the Worcester Polytechnic Institute in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science.

8/24/2023 - 4/27/2023

Abstract

Lyn is a web-based platform that enables users to live code and explore artificial life (AL) simulations for artistic purposes. We developed Lyn to be a simple and intuitive environment that targets users with little to no programming experience. The platform was inspired by previous live-coding systems and includes features such as a new programming language designed specifically for Lyn, GPU acceleration of simulations, audio reactivity, and a variety of documentation sources for easier exploration of the system. We evaluated Lyn's usability with quantitative and qualitative methods. These user studies showed that while users enjoyed Lyn and appreciated its real-time feedback they also wanted more guidance on how to use the system, including feedback on how to debug programming errors.

TABLE OF CONTENTS

Page

List of Tables **iv**

List of Figures **v**

1 Introduction **1**

2 Background **3**

2.1 Artificial Life . . . 3

2.1.1 Reaction Diffusion . . . 4

2.1.2 Cellular Automata . . . 5

2.2 Live Coding . . . 6

2.2.1 Hydra . . . 6

2.2.2 The Force . . . 7

2.2.3 LiveCodeLab . . . 8

2.2.4 Gibber . . . 9

3 Implementation **10**

3.1 Language Design . . . 11

3.1.1 Functions . . . 11

3.1.2 Inputs . . . 12

3.2 System Design . . . 12

3.2.1 Parsing and Visuals . . . 13

3.2.2 User Interface . . . 13

3.2.3 Additional Features . . . 15

4 Evaluation **18**

4.1 Results . . . 18

4.1.1 The interactivity and real-time feedback from live coding were enjoyable and interesting. . . . 19

4.1.2 Current features can be further developed to make them more accessible and easier to find. . . . 19

4.1.3 New features could be implemented in the system to create an easier and more enjoyable experience. 20

5 Conclusions and Recommendations **22**

Appendix A: Study Protocol **i**

Covid Considerations i

Opening briefing for testers i

Think-A-Loud ii

Appendix B: Survey **iii**

Appendix C: Study Questions **iv**

Preliminary Questions iv

Interview Questions iv

Appendix D: Grammar **v**

Appendix E: IRB Approval **viii**

Bibliography **ix**

List of Tables

Table		Page
3.1	Notes of features of different live coding systems	11
4.1	Code for open responses answers and transcripts	18

List of Figures

Figure		Page
1.1	Screenshot of Lyn	2
2.1	Screenshot of Lyn with reactive diffusion simulation	4
2.2	Screenshot from Karl Sim's explanation of the Gray-Scott Model	5
2.3	Screenshot of Lyn with Conway's Game of Life	6
2.4	Screenshot of LiveCodeLab with sample visual	8
2.5	Screenshot of Gibber with sample demonstration	9
3.1	Code Sample using Lyn	12
3.2	Screenshot of before and after calling the kaleidoscope function within Lyn	14
3.3	Screenshot of Lyn's parameter information with the graph tracking changes in parameter values based on audio	14
3.4	Buttons for the "random" button and command list	15
3.5	Screenshot of the parameter information	15
3.6	Screenshot of the command list	16
3.7	Screenshot of audio tab	17
3.8	Color preview within the Lyn text area	17

CHAPTER 1

INTRODUCTION

The field of artificial life (AL) has been gaining significant attention in recent years and for good reason [1]. With the ability to study the nature of "life", researchers and enthusiasts alike have been able to explore these simulations to uncover new insights into different fields such as biology, evolution, and the origins of life. However, the accessibility of these simulations has often been limited to those with advanced technical knowledge, making it difficult for many to explore and participate in this fascinating field. In particular, many simulations are limited by the CPU and require GPU acceleration to run at interactive rates, which is a complex topic requiring extensive development expertise.

To address this challenge, we created Lyn, a web-based platform for live coding artificial life simulations available at `https://evelyntrvn.github.io/Lyn/`. We aimed to enable users of all programming skill levels to explore the possibilities of AL simulations with interactive coding. By offering access to these simulations through a web browser, this platform enables anyone with an internet connection to experience the excitement of creating and experimenting with artificial life.

Beyond the educational possibilities, the system also offers a creative medium for those interested in exploring the art of AL simulations [2]. Users can use the platform to create unique and beautiful visualizations or to experiment with different parameters through code to explore the potential of these virtual worlds.

Figure 1.1: *Screenshot of Lyn*

CHAPTER 2

BACKGROUND

2.1 Artificial Life

Researchers aim to understand, design, and build artificial systems that exhibit lifelike behavior in the interdisciplinary field known as artificial life, also known as a-life, alife, or AL. It draws from a multitude of fields, including biology, chemistry, computer science, physics, mathematics, evolutionary science, origins of life research, and more [3]. We can define literally AL as "life made by humans rather than by nature" [4]. The findings from this field have contributed to the understanding of different principles of life and its process, as well as to the development of technologies and artistic practices.

AL was officially established as a scientific field when Christopher Langton, an American computer scientist, hosted the first-ever alife workshop in 1987. He coined the term "Artificial Life" for "a new discipline that studies 'natural' life by attempting to recreate biological phenomena from scratch within computers and other "artificial" media [5]. However, the discipline has its roots in the former works of computer scientist John von Neumann, who proposed the concept of self-reproducing automata in 1948 [3]. Since then, artificial life has grown into a vibrant research field, with numerous theoretical and practical applications.

While AL has contributed to the development of novel technologies and scientific understandings of life, it has also inspired many artists, as the discipline brings questions such as what defines life and whether life can exist in a machine [6]. Today especially, computerized systems and other technologies have enabled AL to find its way into creative fields. Artists have recognized the aesthetic of mimicking behaviors found in nature and its processes and have explored it through different computerized systems. AL techniques provide also provide a new opportunity for interactivity where the dynamics of biological systems which have not been explicitly defined before can be investigated and enjoyed [7]. This form of artwork can be viewed as controversial

due to the fact that a given piece is generated by a computer, but it also provides new possibilities and inspirations outside of traditional artistic techniques [4]. AL extends the scope of art-as-it-is and opens new experiences to both its audience and the artist.

The discipline has only existed for roughly over three decades, but throughout that period, computational capabilities have quickly advanced in everyday consumer technologies. With that in mind, AL findings and technologies have since provided new methods, tools, algorithms, questions, ideas, and insights toward a multitude of practices. As the domain continues to grow, it has not only made it possible to investigate both life-as-it-is and life-as-it-could-be but has also opened up an endless range of possibilities for technological and creative pursuits.

2.1.1 Reaction Diffusion

Reaction-diffusion systems are systems that model how two chemicals transform and react to one another as they diffuse through a medium. When determining the change of state for the substances, one would need to account [8]:

- The changes in position and velocity based on Newton's laws of motion
- The osmotic pressures based on chemical data, and stresses from elasticizes and motion
- The chemical reactions
- The diffusion of the chemical substances

The behavior of these fluids can produce different patterns and can be defined through reaction-diffusion equations such as the Gray-Scott equations.

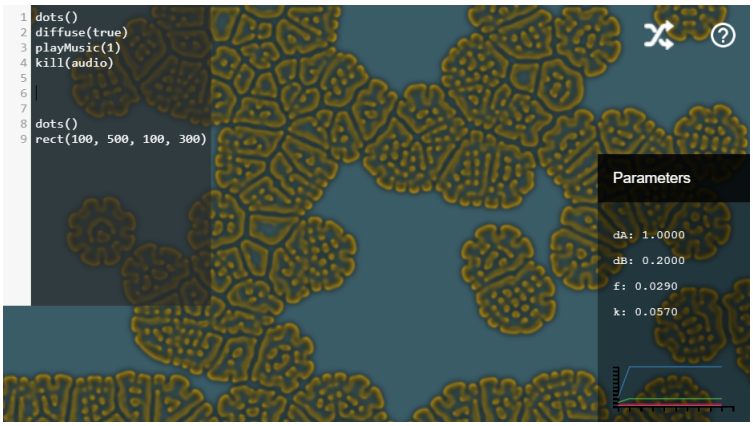


Figure 2.1: *Screenshot of Lyn with reactive diffusion simulation*

The Gray-Scott model simulates the behavior of two chemicals given that:

- Chemical A is added into a system at a specified "feed rate"

- A reaction occurs when two Chemical B's convert an A into a B
- Chemical B is removed from the system at a specified "kill" rate

The diffusion rate of both Chemicals A and B can also be defined as they can both spread at different speeds. When simulating this model, different concentrations of Chemical A and B can be specified within a grid of cells. As the grid of cells updates its state, it can produce a multitude of patterns when simulating the model on a larger scale. The behavior above can be described by the equations below in Figure 2.2:

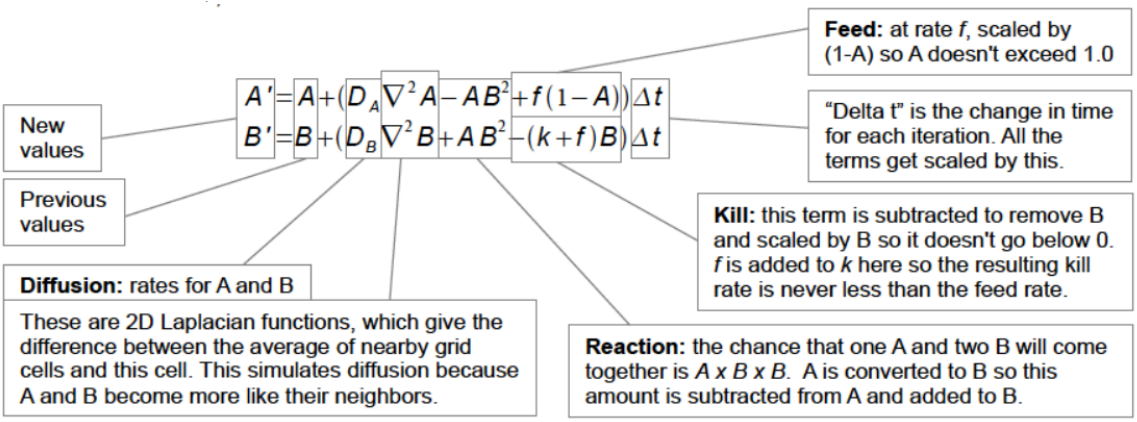


Figure 2.2: *Screenshot from Karl Sim's explanation of the Gray-Scott Model*

While these rules may seem simple, changing these variables can produce a number of different complex and dynamic behaviors such as a "coral growth" simulation (DA=1.0, DB=.5, f=.0545, k=.062) and a "mitosis" simulation (DA=1.0, DB=.5, f=.0367, k=.0649) [9].

2.1.2 Cellular Automata

A cellular automaton is a discrete and abstract computation system that consists of a collection of cells on a grid in a given shape that transforms and evolves based on a set of rules that define a cell's new state depending on the states of its neighboring cells at a given time step. As mentioned before, this concept was created by von Neumann back in 1948 but is still being explored to this day.

A famous example of a cellular automaton is John Conway's Game of Life created in 1970 [10]. This zero-payer game is very simple, where cells are either "dead" or "alive" and evolve from one generation to the next based on only a couple of rules:

- If a live cell has zero or one neighbor, it dies from isolation.
- If a live cell has two or three neighbors, it lives on to the next generation.
- If a live cell has more than three neighbors, it dies from overcrowding.

- If a dead cell has exactly three neighbors, it becomes alive in the next generation.

Since Conway's invention of the Game of Life, it popularized the use of cellular automata with simulations that model can model the behaviors of ants, traffic, galaxies, and more. Even with its simple guidelines, it can produce complex patterns and proved that a computer could be "programmed" to simulate life [9].

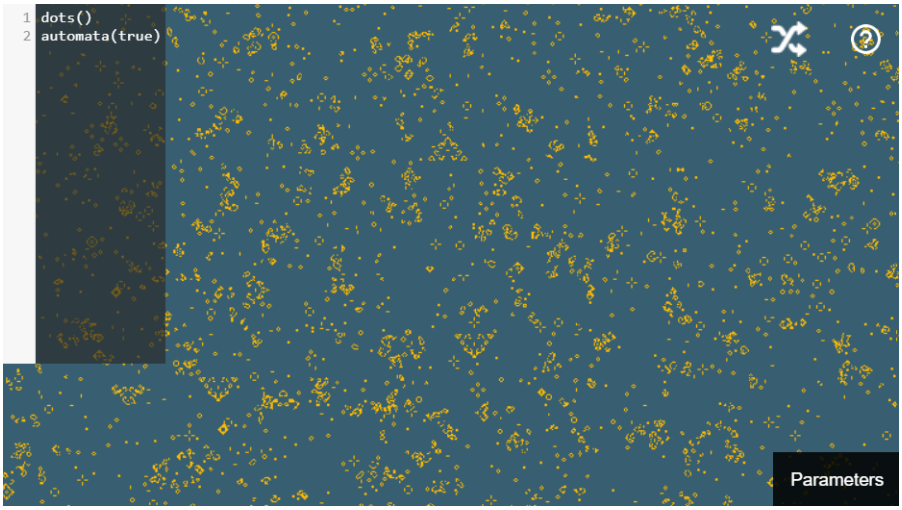


Figure 2.3: *Screenshot of Lyn with Conway's Game of Life*

2.2 Live Coding

Live coding performances consist of performers creating works by programming them as the code is being executed at the same time [11]. While live coding has its roots in creating music and in musical performances, it has also been used within other creative fields such as visual arts, dance, poetry, and more [12]. It offers the opportunity for audience members to explore and experience how the manipulation of different algorithms can affect the generated outcome. As well as for the artist themselves, as they navigate how to improvise on their instrument to generate their creative work for their audience to enjoy [13].

To develop a live coding system, current and previous live coding environments were looked into for a better understanding of what makes a live coding system simple and intuitive. This section explores the different features as well as language design that these environments had that contributed to the development of Lyn.

2.2.1 Hydra

Hydra is a live coding environment for creating visuals that run on the browser. It was developed by Olivia Jack and was written in JavaScript and compiled into WebGL. The syntax of Hydra

utilizes chaining a set of transformations to create the visuals which was inspired by analog modular synthesis [14].

The interface of Hydra is designed to be intuitive and user-friendly, with a simple and straightforward layout that allows users to quickly start creating their own visual compositions. The interface consists of a main code editor window, where users can write and modify their code, as well as a preview window that shows the current output of the code. Users can also interact with the output of the code using a range of controls and parameters, allowing them to modify and manipulate the visual output in real time.

The user interface was also inspired by TidalCycles, a live-coding interface focused on creating music created by Alex McLean, where it takes a pattern in time as the base element and everything is based on that given pattern [15]. For Hydra, "the base element is a transformation from coordinates to color", meaning all other functions revolve around changing the colors or coordinates [16].

The syntax of Hydra is designed to be simple and expressive, allowing users to easily create complex and dynamic visual patterns through the use of chaining and transformation functions. For example, the following code generates a simple visual pattern using Hydra's syntax:

```
shape(5)                          //create a shape with 5 sides
    .color(2.83,0.91,0.39) //setting the color of the shape
    .rotate(90, 1)         //rotating consistently by 90 degrees in 1 sec
    .scale(2)         //scaling the shape by 2
      .out()
```

Hydra's functions are not very complex which allows users with little to no experience with coding an accessible way to explore live-coding visuals. The functions can also be broken down into five different categories: source, geometry, color, blend, and modulate. Each category has between 7 to 16 functions which all enable users to create a multitude of dynamic and interactive visual components.

2.2.2 The Force

The Force is a coding environment developed by Shawn Lawson for generating audio-responsive visuals with the OpenGL Shading Language (GLSL), designed as a live-coding system for a single performer [17]. The syntax is based on GLSL which makes it difficult for users if they do not have a prior understanding of the language or how GPU-run simulations may run. For example, the following code changes the visual from white to black:

```
void main () {
  float color = sin( time );    //setting color based on time
  gl_FragColor = vec4( color,color,color, 1. ); //setting the color of the visual
}
```

Similar to Hydra, the interface consists of a code editor for users to input their code, however, the code will automatically compile and run as the user types without the need to click a "Run" button or press a keyboard shortcut to update the image given that there the compiler does not return an error. On the bottom of the screen, the platform had additional features such as adjusting the font size of the code editor, being able to add audio input, a function reference, and more. While the syntax may be a bit difficult for new users to get used to, having the function reference on the platform allowed for more ease of use.

2.2.3 LiveCodeLab

Another live-coding environment is LiveCodeLab, which was developed by Davide Della Casa and Guy John to generate music but focuses on creating 3D visuals. LiveCodeLab 2.0 was released in 2014 and was inspired by other live code environments such as Jsaxus' from its graphic styles as well as Flaxus from its visual effectiveness [18]. The main motive of the project was to develop an environment that would be easily accessible to individuals with limited computer skills, allowing them to produce expressive and creative works using code [18].

The syntax of LiveCodeLab was designed to be concise and very readable, using simple terms to quickly create visuals and audio such as "box" or "ball". For example, a sample piece of code for a simple 3D visual can be found below:

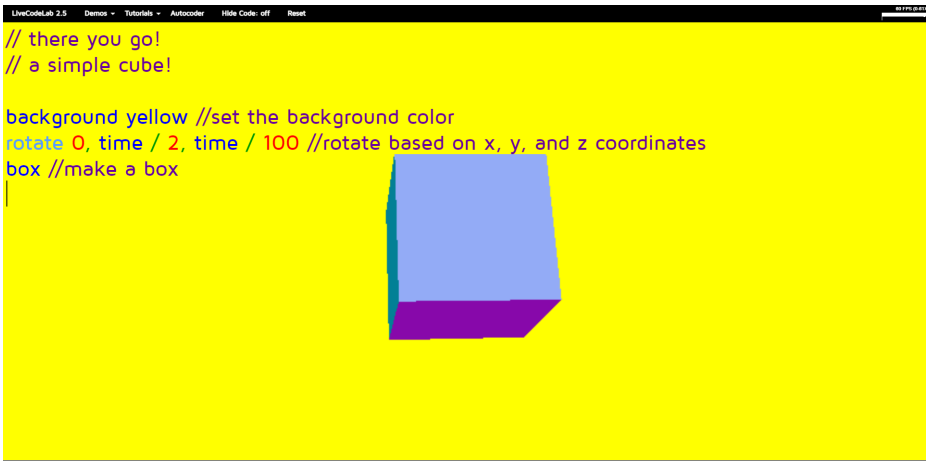


Figure 2.4: *Screenshot of LiveCodeLab with sample visual*

In addition to the simplicity of the syntax, the editor itself is also simple to use and navigate. Similar to other environments, LiveCodeLab processes the code as the user types it out without the need to press any buttons or key commands for updates. It also provides pre-written tutorials and sample demonstrations in the navigation bar, which will load the content into the editor. Additionally, it features an Autocoder which will randomly modify the user code in the editor to allow for further exploration and show users what else can be done with the environment.

2.2.4 Gibber

Gibber, developed by Charlie Roberts, is a live-coding environment and programming library that is designed for audiovisuals [19]. Users can access the environment either by downloading the environment from Github or visiting the Gibber website [20] [21].

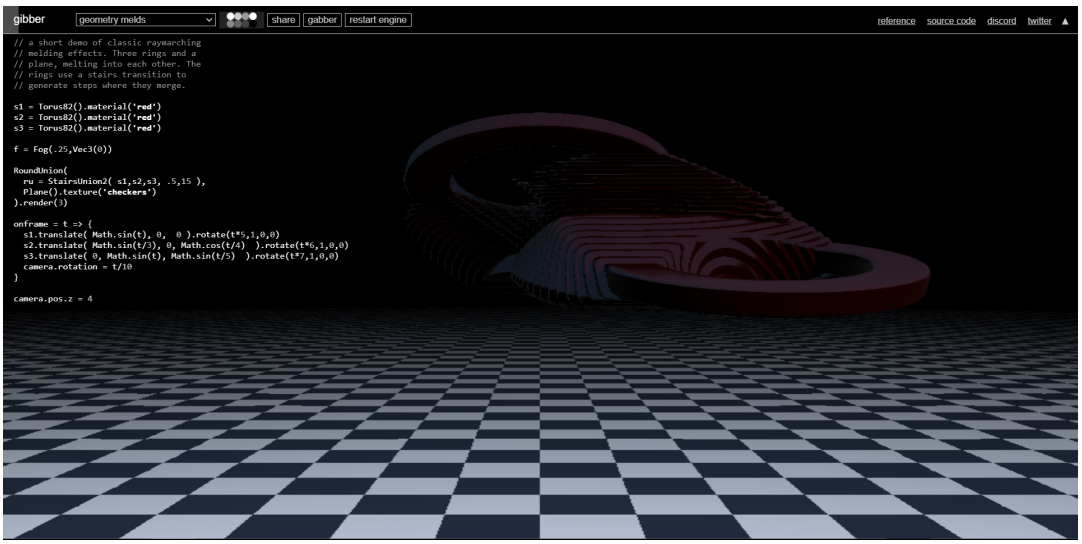


Figure 2.5: *Screenshot of Gibber with sample demonstration*

The browser-based platform was created with the intent that users would code in JavaScript, using the Gibber library to make live coding performances [22]. Because of this, users with JavaScript experience can do more. To run the code, users have the choice of whether to execute all the code or a given line or selection based on the keyboard commands, making algorave performances more accessible for non-programmers. Within the navigation bar, it also provides various tutorials for users to utilize, which will populate the code editor with example code and comments on how to use the library and environment.

Chapter 3

Implementation

In an effort to make AL simulations with live code while still being accessible to both programmers and non-programmers, the systems mentioned previously were used as inspiration and the basis of Lyn. Whether or not the user knew about AL simulations or programming, our goal was for users to quickly jump in and experiment with generating visuals. Table 3.1 notes some of the main features and notes that I integrated into the development of Lyn.

Hydra	• Web-based • Intuitive because the functions build off one another • Readable syntax • Functions separated into different categories • Random generator
The Force	• Web-based • Function reference • Audio input
LiveCodeLab	• Web-based • Simple UI • Easy/readable syntax
Gibber	• Web-based • JavaScript based • Executing all or line/block of code

Table 3.1: *Notes of features of different live coding systems*

3.1 Language Design

3.1.1 Functions

Given that one goal of Lyn is for users to be able to experiment with different simulations, users should still be able to adjust the associated variable values easily. Keeping this in mind, to adjust a variable's value, users simply had to write the function name with a pair of parentheses containing the new value. If a function requires any inputs, they should be within the parentheses and separated by commas.

This standard carried through to other functions, such as starting the diffusion simulation or creating a rectangle on the screen. Inspired by Hydra, and because the focus of Lyn is on AL simulation, the basis that functions revolved around was the AL simulations themselves. As a

```
1 dots()
2 diffuse(true)
3 playMusic(1)
4
5 feed(0.062)
6 kill(audio)
7 colorB( #EDFF00)
8 kal()
```

Figure 3.1: *Code Sample using Lyn*

result, functions were divided up into five categories: simulations, geometry, color, post-processing, and audio. Sample code from Lyn can be found below:

Within the code sample in Figure 3.1, `dots()` the random dots are first generated on the screen. The reaction-diffusion is then called by `diffuse(true)` and `play(1)` causes the audio file that was uploaded by the user to play. Following this, the feed rate is set to 0.062 with `feed(0.062)`, and the kill rate is set to the playing audio's frequency with the `kill(audio)` function. In line 7, `colorB(#EDFF00)` updated color B to a new color, and in line 8, `kal()`, or the kaleidoscope effect, is applied on top of the simulation.

3.1.2 Inputs

Certain functions may need the user to include an input for the function to work. The different input types that are implemented into Lyn include *primaries*, *booleans*, *colors*, and *audio*.

A primary consists of either a float or integer value. These are the main numerical value used when exploring the simulations, specifically when setting variables to varying numbers.

A boolean, similar to other programming languages, indicates either true or false. This is mainly used when calling a simulation and setting it as true or false. When set to true, it runs the simulation and when set to false, causes the simulation to stop playing.

For setting different colors for the visual representation of the simulation, a user can input a color value by typing in a hex value. When a user types any hex value, a color preview will appear next to the input as seen in Figure 3.1. When clicked, a color picker will appear which will be discussed in Section 3.2.3.

As another focus of the project was to incorporate artistic performance, involving music was another interest. Being able to set a variable to an audio input was implemented to see how a simulation changes based on the audio's frequency. In Lyn's current state, the audio input can only be used for setting the reactive diffusion inputs.

3.2 System Design

To make Lyn an easy-to-find and use platform, it was developed using HTML, CSS, JavaScript, and GLSL to create a web-based system. The visuals are run using the GPU to allow for more

efficient processing and smoother animations. To create this live-coding platform, many other tools were used to develop Lyn which will be discussed in this section.

3.2.1 Parsing and Visuals

3.2.1.1 Peggy.js

Peggy.js is a parser generator, based on Peg.js, which is an open-source library for producing parsers in JavaScript. Input types, as well as the structure of functions, were specified in a grammar to outline the rules of the new language. Based on that grammar, Peggy generates a parser to iterate through the users' code and outputs the JavaScript code that is to be run. I was able to create a simple language, defined in the grammar found in Appendix 5, and generate a parser within the Lyn system so that individuals with little to no experience would not require previous knowledge of current programming languages.

3.2.1.2 WebGL

WebGL or the Web Graphics Library, allows for 2D or 3D graphics to be rendered in compatible web browsers through GLSL and the HTML canvas element. This set of functions, which is included in almost all modern browsers, enabled Lyn to take advantage of the graphics processing unit, also known as the GPU, on a user's device when creating the visuals of the simulation. By having the visuals run on GPU, processing of the simulations can be done more efficiently and smoother as compared to being run on the central processing unit, or CPU. Reaction-diffusion would not be able to run fullscreen using the CPU alone; incorporating the GPU was an essential component of developing Lyn.

3.2.2 User Interface

3.2.2.1 CodeMirror

CodeMirror is a code editing component that can be integrated into websites to enable text input. It has many features that make it useful for live coding and programming in general, such as line numbers, being able to undo or redo, flexible styling, and allowing editor extensions.

These extensions included the https://github.com/replit/Codemirror-CSS-color-picker which enables a color picker to appear in the CodeMirror text editor when a user is specifying a color. This lets users not worry about knowing the hex code of a specific color, as they can simply select a color on the color picker.

3.2.2.2 Post-Pre

Post-Pre, developed by Cole Granof, is a set of presets based on merge-pass, a post-processing library that allows for a combination of effects to run easily on an image, canvas, or texture. By

integrating this library, it allowed for more visual effects to be generated within the Lyn system for users to explore.

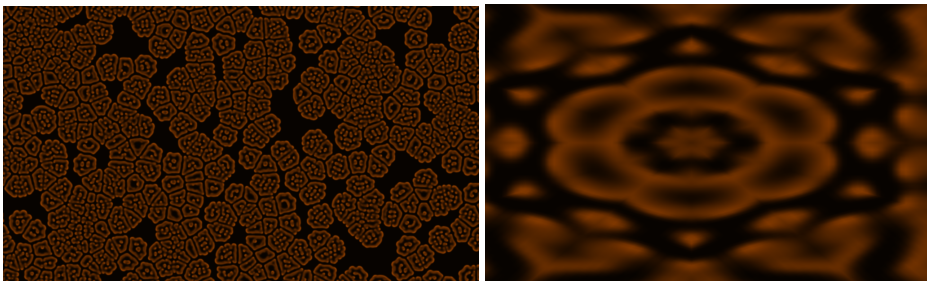


Figure 3.2: *Screenshot of before and after calling the kaleidoscope function within Lyn*

3.2.2.3 D3.js

D3.js is a JavaScript library and framework that focuses on creating visualizations by efficiently manipulating documents based data using HTML, SVG, and CSS [23]. It allows for fast and dynamic animations and interaction for data visualizations. This tool was utilized to allow users to see how the variables change over time if a user were adjusting them manually or through audio reactivity which is later discussed in Section 3.2.3.

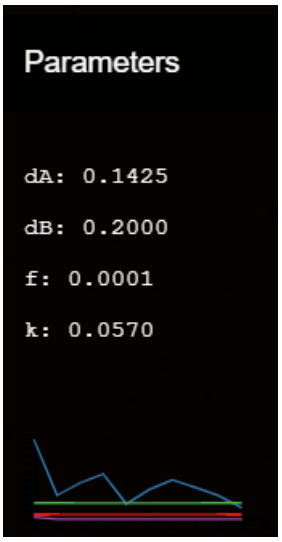


Figure 3.3: *Screenshot of Lyn's parameter information with the graph tracking changes in parameter values based on audio*

3.2.2.4 GitHub and GitHub Pages

GitHub is a web-based platform for version control and collaboration, primarily used in software development. It allows users to easily track changes in code over time, merge changes, and manage issues and bugs. Throughout this project, GitHub was utilized to keep a record of updates in the code such as new features or bug fixes.

GitHub also provides other tools that make it a useful platform to use such as GitHub Pages, which is a static-site hosting service provided by GitHub that is built off the HTML, CSS, and JavaScript files within a repository. This service is currently being used to host Lyn for users to access.

3.2.3 Additional Features

Outside of the live coding and visualization aspect of the Lyn system, additional features were added to the platform with the goal of providing a user-friendly experience and allowing for more exploration of the simulations. This section discusses and showcases these features below.

The random button when clicked randomly selects from 4 presets of the reactive diffusion parameters, providing users a start for exploration and provides references to different variations of the reactive diffusion.

Figure 3.4: *Buttons for the "random" button and command list*

Parameter information can be found in the "Parameter Info" box where the current values of each parameter in the reactive diffusion can be seen and monitored by the user. Additionally, there is a graph where the user can observe how the value of a variable fluctuates or compares to other variables. The graph was implemented mainly for the case when a variable is set to an audio input.

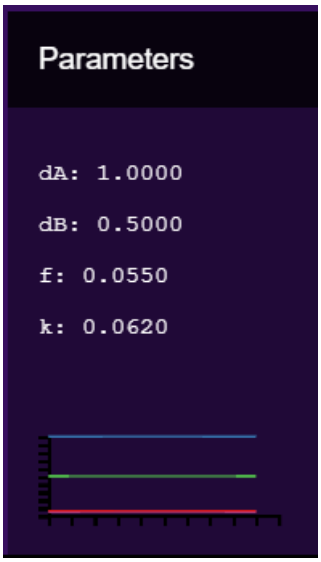


Figure 3.5: *Screenshot of the parameter information*

A command list can be found in the top right for users to reference all the different functions and how to utilize them to generate various and unique visualizations. Within the command list, functions are broken up into different tabs depending on their functionality which are “Simulation“, “Audio”, “Shapes and Colors”, and “Effects”. Additionally, the first tab within the command list provides a quick overview of the platform as well as instructions on how to run the code.

Figure 3.6: *Screenshot of the command list*

We also implemented audio reactivity within Lyn to provide more opportunities for users to play and explore how simulations change depending on the frequency content of the audio. Within the command list, users can navigate to the “Audio” tab and upload up to three audio files. By doing so, these audio files can be referenced and played using the `playMusic(1)` function where `1` indicates that the audio file in track 1 should be played. From there, variables can be set to the audio by simply typing in “audio” where a number value should be within a given function.

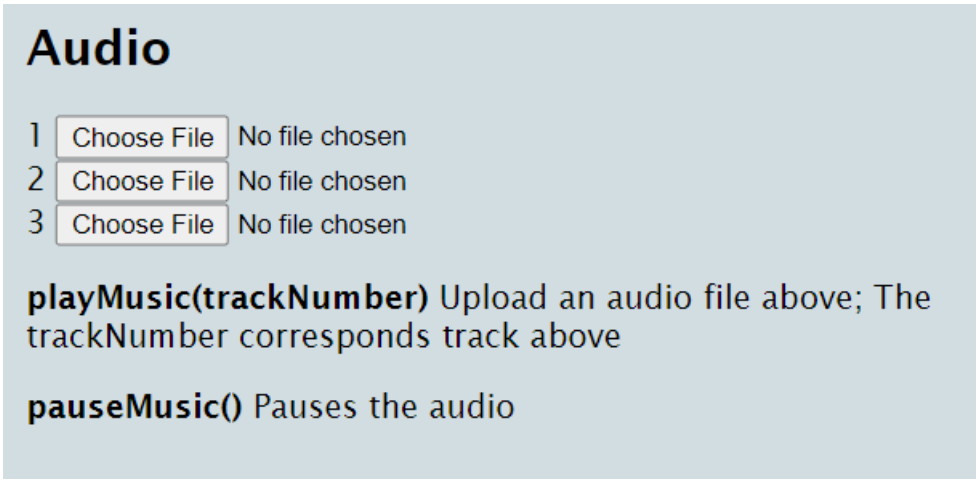


Figure 3.7: *Screenshot of audio tab*

As many people do not memorize hex codes, to ease the process of setting different colors, users can utilize the color picker feature within the code area to select a color. A color picker box will appear once a user begins typing any hex code. Once the small color preview appears, the user can click that box for the color picker to open and users can select their color. From there, a hex code will be generated based on that color.

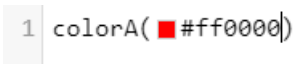


Figure 3.8: *Color preview within the Lyn text area*

For running code, users can either run all the code within the text area together by clicking CTRL + ENTER or they may choose to only run a selected block of code or individual line with SHIFT + ENTER. Being able to only run a line of code or a selected block was implemented to allow for more flexibility in the live coding experience, especially if a user wanted to create visuals based on the timing of any audio.

CHAPTER 4

EVALUATION

To evaluate the usability of Lyn, we conducted user testing where participants performed a think-a-loud [24] as they performed various tasks on the platform (Appendix 5). Following this, they answered a survey based on the System Usability Scale (SUS) [25] and a series of open-ended questions regarding their experience and feedback (Appendix 5, Appendix 5). Six sessions were held with WPI students; they provided valuable insights and comments. Out of the six participants, only two had prior coding experience with computer science as a major, and one participant with some coding experience from classes.

To analyze the open responses, the think-a-loud and responses to the open-ended questions were then transcribed and coded to identify patterns and improvement points in the system. The transcripts were coded with the following categories found in the table below.

Language Design
Improvements in current features
Suggestions for new features
Difficulties/dislikes

Table 4.1: *Code for open responses answers and transcripts*

4.1 Results

The survey had a SUS score of 64.5, meaning the platform in terms of ease of use, the platform performed slightly below the average score of 68 [25]. While many positive comments were mentioned during the user testing, there are many further developments that can be implemented. Overall during the testing, participants were able to complete most of the tasks with little to no assistance. This section will discuss the major findings and opportunities for development that

emerged after the analysis.

4.1.1 The interactivity and real-time feedback from live coding were enjoyable and interesting.

During the user testing, it was clear that participants enjoyed the level of interactivity and customization available within the platform. All six participants expressed their appreciation for being able to personalize their simulations, saying comments such as “What I liked about using Lyn was that it was very customizable” when asked what things they liked and saying “Nice it looks like I’m watching, like the intro movie” after changing the colors and parameters of their simulation. Interestingly, some participants were particularly drawn to the way in which the audio reacted to the simulations, saying “I really also do like the option that there is to play audio, so it would react to whatever music I would be playing”, while others were more interested in the visual aspect of the platform with remarks about the visuals such as that the “I found this new appreciation coding so like seeing that it’ll come from the text box and then, like seeing the live feed on the side, it’s really cool”. As mentioned before that only two of the participants had prior knowledge of programming before the testing, highlighting the ease of the platform for those without prior experience, with one participant even mentioning “I would say, nice and easy to use, because the code actually is very understandable”.

Upon starting their reactive diffusion simulations, each participant had a unique reaction to the visuals generated. Most users were excited and intrigued to watch and see how the simulation evolved over time, however, a few participants also expressed discomfort with certain visualizations. Fortunately, as the platform allowed these participants to adjust the simulation or select a randomized preset, it allowed participants to find a visual that better suited their preferences. Overall, the reactions to the simulations were positive and showed the potential for the platform to engage users and spark their creativity.

4.1.2 Current features can be further developed to make them more accessible and easier to find.

All the participants made comments regarding the command list regarding either its contents or ways a user can interact with it. To start, the placement of the icon as well as the icon for the command list confused some participants. Five participants were able to navigate and find the popup with no help. Based on the options that were given to them, some determined the “?” button made the most sense which is why they clicked on it to find the list. The one participant who was unable to find the command list on the first try was due to their confusion on what a “command list” would be or look like with remarks such as “I had to think about it, and I thought like that was the only option cause nothing else really screamed command list! But even then I think the question mark more so, gives me like frequently asked questions more so than command list”. Some suggestions were to change the icon to better represent the feature.

Additionally, for users with less experience, it was also suggested to put the icon closer to the text box for easier access or enable keeping the reference open for users to refer to whenever.

With regard to the information within the command list, it was suggested to differentiate the function, its inputs, and its description more clearly such as through different styling or documenting the functions in a table. Additionally, the data types should be clearly mentioned for users as many of the participants had difficulty and were unsure of how to set a variable to the "audio" data type. However, the participants with little to no prior programming experience commented on how they liked having the command list and being able to copy and paste functions straight into their code and make minimal adjustments, saying "I like that they had directions for us, or like templates in the sense, so that way I could just copy and paste".

Outside of the command list, the color picker was a feature that was often not discovered or was suggested without knowing had been implemented. When users were asked to change one of the colors of the simulation, three participants asked for a hex code to input because they did not have one in mind, and four participants made a comment regarding the color picker square that appeared but did not click on it. The two participants who did click on the color picker found it quite helpful but did mention it may not have been intuitive to click the color picker box without prior programming experience.

4.1.3 New features could be implemented in the system to create an easier and more enjoyable experience.

While users were able to navigate the platform in its current state, participants provided valuable feedback on how to improve the Lyn platform experience The most common suggestions were regarding tutorials, the text box as well as the audio features.

While participants with less coding experience were able to complete the tasks and generate different visualizations, they expressed a need for a tutorial on the platform and the language. Current live-coding systems can be used as a reference for future development such as having tutorials by having simple preset code with comments to guide users, such as in Gibber. Another approach could be to implement a tutorial showing the different features of the platform with a popup when a user starts the website.

For the coding area, a common issue that was found was the lack of feedback when a participant had an issue in their code. While going through different tasks and there was an error in the code, participants were often confused by the lack of response. When asked for suggestions, one participant recommended "having like an error function just like, oh, this is not gonna work because of this". For future iterations of Lyn, either providing error messages or underlining errors within the coding area would allow users to be able to identify their mistakes and prevent confusion when no changes occur if the current code cannot run. Additionally, participants found it cumbersome to go back and forth between the command list and the coding area when they did not remember the exact function. To address this issue, an autocomplete feature within the code

area could be implemented and would allow users to write code more quickly and efficiently.

Regarding audio features, participants suggested being able to adjust the volume level of the audio on the interface and having a track slider to show where the music is in the audio track. These features would allow users to better time changes in the visuals and improve the overall experience of working with audio.

Other suggestions from participants included adding more shapes and simulations, implementing a full-screen mode to hide all interface features and focus solely on the simulation, adding a save code feature, and enabling sharing of simulations and visualizations with others.

Chapter 5

Conclusions and Recommendations

By the end of the project, the final iteration of Lyn provided a promising platform for live coding artificial life simulations for users to explore and generate. Many people showed interest in Lyn due to both AL and live coding aspects when being presented during WPI's 2023 Project Presentation Day. At its current state, Lyn offers various tools and features for users to utilize, however, there are several opportunities for improvement that remain for future development.

For the last iteration of this project, there were only two AL simulations available for users to test while there is a multitude of simulations that people can explore and enjoy within the field of AL. Even with the two AL simulations, only reactive diffusion provided parameters that could be adjusted whereas Conway's Way of Life could only be adjusted with the starting geometry. The implementation of more AL simulations involves adjusting the current language design as its current state may not offer user-friendly ways of adjusting parameters for that simulation. Outside of AL simulations, more functions and options for creating different geometry, adjusting post-processing effects, and experimenting with audio would offer users a wider toolset to generate unique visualizations. Further development of the language would allow users the ability to explore more functions and data types

Regarding Lyn's system design, the user interface can be made more user-friendly and intuitive so that users have the abilities and knowledge of using Lyn to all of its abilities. This would involve improving the current features within Lyn, as well as implementing new ones. Certain placements and indicators for features such as the command list and color picker should be updated so to better represent them and allow users to more easily find them. The formatting and explanations provided within the command list should also be improved, especially for users with little to no experience in programming, for easier understanding of functions and the language itself. Outside of these current features, new capabilities should also be implemented

into Lyn including having feedback on errors and autocomplete within the coding area, more audio capabilities, as well as the tutorials of the platform and its language.

As mentioned previously, the Lyn system has generated interest from various individuals due to its combination of artificial life and live coding aspects. The project additionally also showcased a more creative side to code. With continued development, Lyn has the potential to become a valuable resource and medium for anyone interested in exploring the possibilities of AL simulations and creative code. Lyn is free and open-source software, available at `https://github.com/evelyntrvn/Lyn`.

APPENDIX A: STUDY PROTOCOL

The beginning of each study procedure will start with an introduction to the project, explaining their right to refuse any questions, and recording consent to participate with a digital signature or in-person signature. The interview will be recorded using Zoom to capture the screen and participants' comments.

Participants will be asked a preliminary set of questions for general demographic information. Following that, they will be directed to Lyn (https://evelyntrvn.github.io/Lyn/) where they will be asked to perform a think-a-loud while completing various tasks using the platform found below.

After completing the set of tasks, participants are directed to another URL and asked to fill out a short survey to characterize aspects of their subjective experience and solicit suggestions for improving the experience. Finally, the participants will be asked a short set of open-ended questions to further elaborate their experience and opinions on the platform. Participation in the study should take 15 to 30 minutes.

Covid Considerations

Prior to a participant's session, the investigator will send an email to the participant to confirm the session details and inform volunteers that if they have COVID-19, show symptons, or has been in close contact with anyone who has COVID-19, they should inform the investigator and cancel the session. Additionally, the email will encourage but not require the volunteer to wear a mask during the session. The email template can be found below.

Before beginning the study, participants will be asked if they have COVID-19 or symptoms of COVID-19 or have been in close contact with anyone who has or had COVID-19. During user testing, only the investigator and one participant will be present in the testing room, where the investigator will be wearing a mask. The investigator will also maintain a 6-foot distance unless needed to proceed with the investigation such as navigating to the survey or fixing any issues with links. All equipment and areas that will be used will be sanitized prior to each session such as laptops, computer mouses, door knobs, etc.

Opening briefing for testers

"Hello!

Thank you for participating in this session. The purpose of this study is to obtain user feedback to evaluate the usability and design of a web-based system to allow users to explore Artificial Life (AL) simulations with live-coding and audio-reactivity.

This session will be recorded and transcribed for analysis purposes only. All your personal information will be kept confidential and used for educational purposes only. This study should take 15-30 minutes. Your participation is voluntary, and you may choose to stop any time or skip any questions or tasks that are asked. Before I continue, do you have any questions, comments, or concerns?

Before we begin, I am going to ask you to look through and sign this consent form which has more information regarding the study."

Think-A-Loud

Users proceed to go onto the website and will be asked to perform the following tasks while thinking aloud:

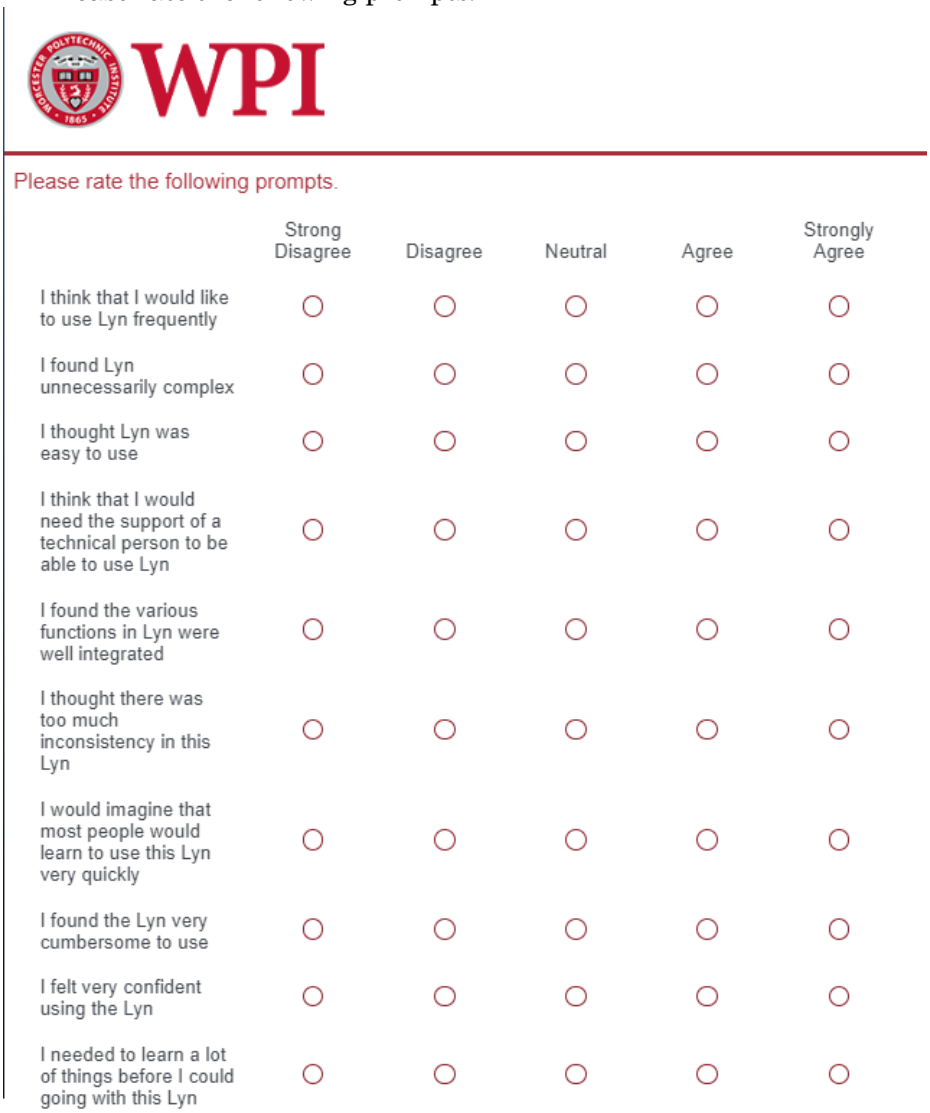
- Look through the command list
- Make random dots
- Start a diffusion simulation*
- Generate a random simulation
- Change one of the colors*
- Change the kill variable*
- Play music*
- Set the feed variable to the audio*
- Try one of the commands in the "Effects" list*
- Explore the website

* These tasks are accomplished by executing a command in the code editor.

APPENDIX B: SURVEY

After completing the tasks, participants will be asked to complete a survey which will be hosted on Qualtrics with the following questions:

Please rate the following prompts.



WPI

Please rate the following prompts.

	Strong Disagree	Disagree	Neutral	Agree	Strongly Agree
I think that I would like to use Lyn frequently	○	○	○	○	○
I found Lyn unnecessarily complex	○	○	○	○	○
I thought Lyn was easy to use	○	○	○	○	○
I think that I would need the support of a technical person to be able to use Lyn	○	○	○	○	○
I found the various functions in Lyn were well integrated	○	○	○	○	○
I thought there was too much inconsistency in this Lyn	○	○	○	○	○
I would imagine that most people would learn to use this Lyn very quickly	○	○	○	○	○
I found the Lyn very cumbersome to use	○	○	○	○	○
I felt very confident using the Lyn	○	○	○	○	○
I needed to learn a lot of things before I could going with this Lyn	○	○	○	○	○

Preliminary Questions

Do you have any programming experience?

Do you know what live-coding is?

Interview Questions

Following the survey, participants will be asked the following questions:

- What did you like?
- What did you dislike?
- Are there any features you would want to see added to the website?

Appendix D: Grammar

```
term "term" = _? body:(keyword / sentence) _? { return body; }

POINT = "."
COMMA = ","
DIGIT = [0-9]
INTDIGIT = [1-9]
TRUE = "true" / "True" { return "true" }
FALSE = "false" / "False" { return "false" }

PRIMARY = float / int

float = first:int dec:dec { return parseFloat(`${first}`) + parseFloat(`${dec}`); }
dec = point:POINT num:DIGIT+ { return parseFloat("." + `${num}`); }
int = first:INTDIGIT dig:DIGIT* { return parseInt(`${text()}`); }
boolean  = TRUE / FALSE

regular = [^{}]+ { return text(); }
expr = regular*

sentence = "{" expr:expr "}" { return expr.join("") } / regular

_ "whitespace" = [ \t\n\r]*

audio = "audio" {return "'audio'"}


/****** Key Words ******/
keyword "keyword" = col / difFct / cellFct /
                    effects / reset / music / shapes /
                    colInput  / $[^{} \t\n\r] +
```

```
// Shapes and styles
shapes = rect / dots
rect = "rect(" _? x:PRIMARY "," _? y:PRIMARY "," _? w:PRIMARY "," _? h:PRIMARY _? ")"
{ return `shape.rect( ${x}, ${y}, ${w}, ${h} )`}

dots = "dots()"{ return `shape.dots()`}


/****** Diffuse attributes ****/
difFct = diffuse / rateA / rateB / feed / kill / size
difInput = PRIMARY / audio
diffuse  = "diffuse(" bool:boolean ")"{ return `setDiffuse(${bool})`;}

rateA = "dA(" r:difInput ")"{ return `rateA(${r})` ; } //change to primary
rateB = "dB(" r:difInput ")"{ return `rateB(${r})` ; }
feed = "feed(" f:difInput ")" { return `feed(${f})` ; }
kill = "kill(" k:difInput ")" { return `kill(${k})` ; }
size = "size(" s:difInput ")" { return `size(${s})` ; }

reset = "reset()" { return `reset()` }

/****** Cellular Automata attributes ****/
cellFct = automata

automata  = "automata(" bool:boolean ")"{ return `setAutomata(${bool})`;}

/***** music ****/
music = playMusic / pauseMusic / time
playMusic = "playMusic(" trackNum:int ")" { return `playMusic(${trackNum})` }
pauseMusic = "pauseMusic()" { return `pauseMusic()` }

// Input and other
time = "time" { return Date.getTime; }

/** colors **/

hexChar = h:[0-9A-Fa-f] { return `${h}`; }
hex = "#" h:(hexChar hexChar hexChar hexChar hexChar hexChar) {return `"#${h.join("")}"`}
```

```
rgb = "rgb(" PRIMARY "," PRIMARY "," PRIMARY ")"{ return text();}

col = colorA / colorB
colInput = hex / rgb
colorA = "colorA(" h:colInput ")"{ return `col.setColor( "A", ${h} )` ; } //
colorB = "colorB(" h:colInput ")"{ return `col.setColor("B", ${h})` ;}

/** Post Processing **/
effects = editAttribute / noEffect / effect
postProcess = "kal" / "blur" / "celShade" / "foggy" /
"light" / "noise" / "oldFilm" / "vignette"
attribute = "size" / "side" /
            "period" / "speed" / "intensity" /
            "speckIntensity" / "lineIntensity" / "grainIntensity" /
            "blurScalar" / "brightnessScalar" / "brightnessExponent"



noEffect = "noEffect()"{ return `effects.noEffect()` }
effect = func:postProcess "()"_ { return `effects.postEffect("${func}")`}
effectAttribute = attr:attribute { return `${attr}` }

editAttribute = e:postProcess POINT att:effectAttribute "(" val:PRIMARY ")"
{ return `effects.setEffect('${e}', '${att}', ${val})`}
```

Appendix E: IRB Approval

WORCESTER POLYTECHNIC INSTITUTE

100 Institute Road, Worcester MA 01609 USA

Institutional Review Board
FWA #00030698 - HHS #00007374

Notification of IRB Approval

Date:	12-Apr-2023
PI:	Charles Roberts
Protocol Number:	IRB-23-0414
Protocol Title:	GPU-Accelerated Simulations for Artistic Performance
Approved Study Personnel:	Roberts, Charles~Tran, Evelyn~
Effective Date:	12-Apr-2023
Exemption Category:	3
Sponsor*:	

The WPI Institutional Review Board (IRB) has reviewed the materials submitted with regard to the above-mentioned protocol.We have determined that this research is exempt from further IRB review under 45 CFR § 46.104 (d). For a detailed description of the categories of exempt research, please refer to the IRB website.

The study is approved indefinitely unless terminated sooner (in writing) by yourself or the WPI IRB. Amendments or changes to the research that might alter this specific approval must be submitted to the WPI IRB for review and may require a full IRB application in order for the research to continue. You are also required to report any adverse events with regard to your study subjects or their data.

Changes to the research which might affect its exempt status must be submitted to the WPI IRB for review and approval before such changes are put into practice. A full IRB application may be required in order for the research to continue.

Please contact the IRB at irb@wpi.edu if you have any questions.

*if blank, the IRB has not reviewed any funding proposal for this protocol

BIBLIOGRAPHY

[1] W. Aguilar, G. Santamaría-Bonfil, T. Froese, and C. Gershenson, "The Past, Present, and Future of Artificial Life," *Frontiers in Robotics and AI*, vol. 1, 2014. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2014.00008

[2] D. Shiffman, S. Fry, and Z. Marsh, *The nature of code*. D. Shiffman California, USA, 2012.

[3] L. Sinapayen, "Introduction to Artificial Life for People who Like AI," Nov. 2019. [Online]. Available: https://thegradient.pub/an-introduction-to-artificial-life-for-people-who-like-ai/

[4] C. Langton, "Artificial Life: An Overview," 1995. [Online]. Available: https://www.semanticscholar.org/paper/Artificial-Life%3A-An-Overview-Langton/021e5595c614c7b85ca835c5ace37cab89834cf7

[5] S. Wilson and C. G. Langton, "Artificial Life," in *Leonardo*, vol. 24, 1991, p. 244, iSSN: 0024094X Issue: 2 Journal Abbreviation: Leonardo. [Online]. Available: https://www.jstor.org/stable/1575317?origin=crossref

[6] E. Bartlem, "Immersive Artificial Life (A-Life) Art," 2005. [Online]. Available: https://www.ekac.org/edwina.html

[7] S. Penny, "Art and Artificial Life – a Primer," Dec. 2009. [Online]. Available: https://escholarship.org/uc/item/1z07j77x

[8] A. M. Turing, "The chemical basis of morphogenesis," *Bulletin of mathematical biology*, vol. 52, no. 1-2, pp. 153–197, 1990.

[9] K. Sims, "Reaction-Diffusion Tutorial." [Online]. Available: http://www.karlsims.com/rd.html

[10] E. W. Weisstein, "Cellular Automaton," publisher: Wolfram Research, Inc. [Online]. Available: https://mathworld.wolfram.com/

[11] C. Roberts and G. Wakefield, "Tensions & Techniques in Live Coding Performance," in *The Oxford Handbook of Algorithmic Music*, Jan. 2018, journal Abbreviation: The Oxford Handbook of Algorithmic Music.

[12] T. Magnusson, "Herding Cats: Observing Live Coding in the Wild," *Computer Music Journal*, vol. 38, no. 1, pp. 8–16, Mar. 2014. [Online]. Available: https://direct.mit.edu/comj/article/38/1/8-16/94447

[13] I. zmölnig and G. Eckel, "Live coding: An overview," *International Computer Music Conference, ICMC 2007*, pp. 295–298, 01 2007.

[14] O. Jack, "Hydra: live coding networked visuals," Madrid, Spain, Jan. 2019, iSBN: 9788418299087 Pages: 353 Publication Title: Proceedings of the Fourth International Conference on Live Coding Publisher: Medialab Prado / Madrid Destino. [Online]. Available: https://zenodo.org/record/3946269

[15] A. McLean and G. Wiggins, "Tidal–pattern language for the live coding of music," in *Proceedings of the 7th sound and music computing conference*, 2010, pp. 331–334.

[16] P. Kirn, "A free, shared visual playground in the browser: Olivia Jack talks Hydra," Feb. 2019. [Online]. Available: https://cdm.link/2019/02/hydra-olivia-jack/

[17] S. Lawson and R. R. Smith, "The dark side," in *Centro Mexicano para la Música y las Arts Sonoras (Mexico): Proceedings of the Third International Conference on Live Coding*, 2017.

[18] D. Della Casa and G. John, "LiveCodeLab 2.0 and its language LiveCodeLang," in *Proceedings of the 2nd ACM SIGPLAN international workshop on Functional art, music, modeling & design*, ser. FARM '14. New York, NY, USA: Association for Computing Machinery, Sep. 2014, pp. 1–8. [Online]. Available: https://dl.acm.org/doi/10.1145/2633638.2633650

[19] C. Roberts, "gibber.audio.lib," Apr. 2023, original-date: 2014-09-22T00:19:40Z. [Online]. Available: https://github.com/charlieroberts/gibber.audio.lib

[20] C. Roberts and J. Kuchera-Morin, "Gibber: Live coding audio in the browser," in *ICMC*, vol. 11, 2012, p. 6.

[21] C. Roberts, M. Wright, J. Kuchera-Morin, and T. Höllerer, "Gibber: Abstractions for creative multimedia programming," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 67–76.

[22] C. Roberts, M. Wright, and J. Kuchera-Morin, "Beyond Editing: Extended Interaction with Textual Code Fragments," in *Proceedings of the international conference on New Interfaces for Musical Expression*, ser. NIME 2015. Baton Rouge, Louisiana, USA: The School of Music and the Center for Computation and Technology (CCT), Louisiana State University, May 2015, pp. 126–131.

[23] M. Bostock, V. Ogievetsky, and J. Heer, "D^3 data-driven documents," *IEEE transactions on visualization and computer graphics*, vol. 17, no. 12, pp. 2301–2309, 2011.

[24] C. Lewis, *Using the" thinking-aloud" method in cognitive interface design*. IBM TJ Watson Research Center Yorktown Heights, NY, 1982.

[25] A. S. f. P. Affairs, "System Usability Scale (SUS)," Sep. 2013, publisher: Department of Health and Human Services. [Online]. Available: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html