

Final Report

|  |
| --- |
| School of Computing  Faculty of Engineering AND PHYSICAL SCIENCES |

Navigation within a parameter space for creative activity

Libby Athill

Submitted in accordance with the requirements for the degree of  
BSc Computer Science with AI

**2023/24**

**COMP3931 Individual Project**

The candidate confirms that the following have been submitted*:*

*<As an example>*

|  |  |  |
| --- | --- | --- |
| **Items** | **Format** | **Recipient(s) and Date** |
| *Final Report* | *PDF file* | *Uploaded to Minerva (DD/MM/YY)* |
| *Scanned participant consent forms* | *PDF file / file archive* | *Uploaded to Minerva (DD/MM/YY)* |
| *Link to online code repository* | *URL* | *Sent to supervisor and assessor (DD/MM/YY)* |
| *User manuals* | *PDF* | *Sent to client and supervisor (DD/MM/YY)* |

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)

© 2024 The University of Leeds and Libby Athill

# Summary

*In many fields designers often generate digital artefacts by adjusting numerous parameters and selecting from multiple options. To enhance the creative process, a system can support users in navigating this design space; this involves visualising parameter changes (tweaking design elements such as line thickness, colour or shape) as steps in a journey or timeline, suggesting designs close to a given point, and enabling zoom in/out for detailed exploration. The project aims to design and specify a navigation system for a design-focused parameter space, drawing inspiration from computer drawings by artist Darrel Viner in the 1970s; in Viner's work, the artefacts take the form of intricate geometric patterns and abstract compositions, with parameters influencing elements like symmetry, complexity, and spatial arrangements. We aim to allow artists and designers to manipulate these parameters in my solution to produce their own unique designs.*

# Acknowledgements

*<This page should contain any acknowledgements to those who have assisted with your work. Where you have worked as part of a team, you should, where appropriate, reference to any contribution made by others to the project.>*

*Note that it is not acceptable to solicit assistance on ‘proof reading’ which is defined as “the systematic checking and identification of errors in spelling, punctuation, grammar and sentence construction, formatting and layout in the text”; see*

https:://www.leeds.ac.uk/secretariat/documents/proof\_reading\_policy.pdf

# Table of Contents (example of how to format)

Summary iii

Acknowledgements iv

Table of Contents v

Chapter 1 Introduction and Background Research 1

1.1 Introduction 1

1.2 Literature Review

Chapter 2 Design and Implementation 2

2.1 Tables using the ‘table caption’ and ‘table description’ Styles 2

2.2 Figures using the ‘figure caption’ and ‘figure description’ Styles 2

List of References 3

Appendix A Self-appraisal 4

A.1 Level 2 Heading with ‘heading 2’ Style 4

A.1.1 Level 3 Heading with ‘heading 3’ Style 4

A.1.1.1 Level 4 Heading with ‘heading 4’ Style 4

Appendix B External materials 4

B.1 Level 2 Heading with ‘heading 2’ Style 4

# Chapter 1 Introduction and Background Research

## 1.1 Introduction

In many fields designers often generate digital artefacts by adjusting numerous parameters and selecting from multiple options such as brush thickness, colour, saturation, hue and shape. Users will want to access these changes by interacting with a software’s interface, usually via buttons, dropdowns, or sliders. To enhance the creative process, a system can support users in navigating this design space; this involves visualising these parameter changes as steps in a journey or timeline, suggesting designs close to a given point, and enabling zoom in/out for detailed exploration. The project aims to design and specify a navigation system for a design-focused parameter space, drawing inspiration from computer drawings by artist Darrel Viner in the 1970s; in Viner's work, the artefacts take the form of intricate geometric patterns and abstract compositions, with parameters influencing elements like symmetry, complexity, and spatial arrangements. We aim to allow artists and designers to manipulate these parameters in a solution to produce their own unique designs.

## 1.2 Literature Review

Software assisted art has been of great interest since the 1960s and 1970s in which many artists focused on creating works with pen plotters [1] to draw artworks often of a tiled nature, using mathematical algorithms to create image sequences and art that are forced to follow a logical pattern. Darrell Viner, a sculptor-turned-computer-artist used the PICASSO programming language formulated by John Vince to create works that display an artistic artefact changed over iterations [2].

A white paper with black squares

Description automatically generated

**Figure 1.1** Darrel Viner’s “Untitled”

Viner drew his artefacts using a robotic arm controlled with ROS to move a physical pen held by a robotic arm, using iterative functions to draw artworks that included repetitive patterns of shapes that, over time, would change due to parameter changes in each loop.

Similarly, artist Harold Cohen used AARON – a primitive AI program – to create physical artwork using turtle plotting devices, providing the program with a prompt and generates drawings using set rules and decision-making algorithms to generate the composition of the pre-determined elements, such as plants or human figures [6].

A line art of people in a forest

Description automatically generated

**Figure 1.2** Harold Cohen’s “Adam and Eve” [6]

Drawing from the historical context of software-assisted artwork, we can conclude that the use of technology in a creative space can allow the exploration of methods used by traditional artists, enabling users to witness the transformation of their work as parameters are adjusted and tweaked, providing control over the creative process. As seen in Cohen’s work, a system may also be capable of making decisions based on a set of rules that are either collected from existing artworks or from user choices (Cohen’s AARON system “learned” from parameters set by Cohen [6]), allowing software to take a co-creator position in artistic spaces, not only responding to commands but providing suggestions.

Inspired by the works mentioned, our goal is to create an environment where this iterative exploration of parameters is enabled in an easy-to-understand and user-friendly manner to facilitate a software-aided workspace.

# Chapter 2 Design and Implementation

## 2.1 Methodology

In development, we have to consider how the problem is going to be approached and how the time of the project is going to be divided to tackle this.

## A screenshot of a computer Description automatically generated

A diagram of a computer system

Description automatically generated

## 2.2 Developing a Navigation Space

To explore creative parameters, a user will need an easy-to-understand interface that allows them to interact with the software tool. This interface should follow all relevant UI principles to create a pleasant, simple experience for the user; Shneiderman states “when an interactive system is well design, the interface almost disappears, enabling users to concentrate on exploration” [3], and this is one of the many aims of this tool; the user should be able to focus purely on their artwork and not the burden of a user-“unfriendly” interface.

To align with Shniederman’s assertation, user stories can be a useful tool to pinpoint the practical implications of such principles. By creating a narrative, we can find the requirements essential for a user-centric solution.

#### User Stories

|  |
| --- |
| As a designer,  I want a software environment that has a variety of drawing tools  So that I can freely create and experiment with design parameters |
| As an artist,  I want an easy-to-understand and accessible interface  So that I can work with a new piece of software intuitively without interruption |
| As a designer,  I want a tool that will save my progress over time  So that I can explore how my design has changed in each iteration |
| As an artist,  I want a drawing tool that allows me to sketch freely and offer suggestions  So that I can explore creative possibilities |

## 2.2.1 Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Requirement Description | Testing Method | Validation Criteria |
| 1. | Develop an intuitive UI for the user’s interaction with parameters and the navigation of changes over time | Questionnaires, interviews, accessibility extensions e.g. WAVE | User feedback, responsiveness, screenshots |
| 2. | Implement real-time visualisations of the user’s changes to their artefacts | Automated testing of rendering | Short latency, correctly represent changes alongside previous artefact |
| 3. | Create a zoom function for exploration of artefacts | Test cases that verify zoom functionality | Screenshots of functioning zoom, high-resolution zoom of artefacts with no/low latency |
| 4. | Create proximity-based design suggestions to the user | Automated unit testing (analyse user design patterns, test logic, test performance) | User feedback, accuracy of suggestions (surveys), metrics (how frequently do users use this feature?) |
| 5. | Create a design-tracking structure within the system to allow for users to revisit previous iterations of artefacts | Automated testing | Historical data integrity (check stored data matches previous records) |

## 2.3 Designing an Interface

When designing an interface, we must consider the requirements of “standardization, integration, consistency and portability” [3]. As this is a standalone piece of software, we do not have standard features the way Windows or MacOS do, however users are likely to have interacted with similar tools, and consequently have a preexisting schema of where they will expect buttons and functions to lie on an interface, for example most navigation tools will be in the top corners of an application window, and the user will come to expect this from other similar applications. Integration and portability also face the same conclusion: we will not be focusing on the ability to share data across platforms, as this solution is likely to be web-based and therefore should work on both MacOS, Windows, and Linux. However, consistency emerges as the cornerstone of UI design and must be adhered to in all circumstances; if layouts, typographies and colour schemes are not consistent, users will have to relearn the UI every time it changes, hindering user experience and efficiency greatly.

To explore how the user will interact with the software, we need to begin designing the interface. For formatting, we want to ensure we design according to Gestalt principles; buttons in proximity are seen as being related in function, and buttons with high degree of similarity will be seen as having similar behaviours. Grouping and association facilitates a more efficient and user-friendly experience (Johnson, 2013).

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

**Figure 2.1** A sketch of a wireframe for the interface of the tool, showing interaction.

The figure shows a sidebar of parameters the user can change via clicking, which will open an additional dropdown menu of different shape options. This would also be implemented for other functions/parameter changes. The user can interact with the left canvas, zooming, drawing and editing artefacts to see the full result of their changes on the right canvas, and can use the “apply changes” button to save their changes – this will be considered a “step” in the history of the artefact, which will be stored in a history to be rendered as a .GIF when exported (see 2.1.1 Requirement 1). To ensure that we are designing with Gestalt principles, the sidebar on the left contains all the specific parameter changes that the user wishes to experiment with, aligning with the principle of proximity to indicate operations that change the artefacts all sit on the left side of the window [4].

Distinctly separating and grouping functions away from the main canvases also embodies the figure-ground principle; we want to contrast the main subject (the canvas) from our background and surrounding interface to aid users in easily identifying where their “workspace” is in comparison to their tools (Johnson, 2013).

For a creative tool, we want a neutral colour scheme so that it is non-distracting to the user, particularly if they are engaging in colour-centric tasks on the canvas. The palette to be simple to not distract from the canvases, but bold enough to highlight tools selected and buttons.

To ensure that the colours used are suitable for colourblind users, we want to test these with relevant filters to simulate the different types of colour blindness on the palette (Nichols, 2019).

A row of different colors

Description automatically generated

**Figure 2.2** Comparison of original colour palette (True) alongside simulations of colour blindness on perceived colour [5].

Once the colours were applied to the frameworks, they were sent out for feedback.

A computer screen shot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Figure 2.3** Coloured versions of the UI frameworks.

Viewers said the layout and design for the UI was intuitive and easy to understand at a glance, however a point was made that it’s hard to see the full extent of the changes that have been made, and when our goal is focused on parameter changes over time, it would be wise to have a feature that displays this.

The framework was tweaked to display a new “history” sidebar instead of an editing canvas and a view canvas. This was only done in the “light” theme as this was the majority preference in the review.

A screenshot of a computer

Description automatically generated

**Figure 2.4** Amended UI sketch with a “history” tab.

Users said this framework was an improvement, as it presents multiple previous time steps to help the user iterate instead of them having to attempt to remember what their earlier changes were, using the undo and redo buttons to view these changes rather than for editing purposes.

## 2.4 Implementation Framework

As we want this solution to be usable on all operating systems and we want the tool to be easily accessible, it is reasonable to develop this as a web application, letting the work be accessible from a browser without the need for additional installations or environments. This means we will be mainly working with HTML and JavaScript, as we will need to provide interactive aspects to the window for users to engage with creative tools.

### 2.4.1 p5.js/Processing

Processing is a programming language and environment built for visual design and graphical programming, and p5.js is an open-source JavaScript library that allows Processing code to be executed in a web browser without the need for additional plugins. This would be ideal for the tool we are trying to develop as it also facilitates real-time interaction and p5.js allows for image exporting, meaning the user will be able to save the artefacts they create. Combining HTML elements such as buttons with p5.js functions, we should be able to provide the user with a simple interface to design and develop their own artefacts on a p5.js canvas.

### 2.4.2 Matplotlib

Matplotlib is a Python library that is used for creating visualisations, typically for data analysis and scientific computing. To integrate this into a web solution, we would need to combine this alongside another framework such as Django. However, Matplotlib doesn’t facilitate the same real-time interactivity that p5.js does and must re-render with every parameter change and would require more handling of user inputs. Performance would likely suffer too, as every image would need to be generated server-side before being sent to the client, which would delay the iterative design process we wish to encourage in the tool.

## 2.5 p5.js Implementation

To ensure the application has p5.js functionalities, the html code includes the URL of the p5.js library stored on the Cloudflare CDN (Content Delivery Network). This method is very straightforward and allows the library to be loaded quickly. The issue with this however is that if Cloudflare encounters issues, the site may not function properly. The p5.js library could also be downloaded and hosted locally; however, this would require library files to be manually updated, adding extra overheads for a solution that aims to be simple and easy-to-use for non-technical users.

p5.js is easily included as a *<src>* tag and to ensure drawing capabilities are functional, HTML and JavaScript buttons are utilised for user interaction; these buttons trigger JavaScript functions that interact with the canvas drawing functions that are found within the imported library.

A screenshot of a computer

Description automatically generated

**Figure 2.5** Screenshot of the implemented p5.js canvas and associated buttons.

A screenshot of a computer code

Description automatically generatedDrawings are stored as objects with the parameters like type, colour, position, and stroke weight stored as attributes within these. Drawing commands are stored in an array to track the drawing history and to allow for undo/redo functionality.

Undo/redo is stored as a stack, with each function popping or pushing to the stack; every command makes the program store the current state, and the most recent command is pushed to an array of all commands to keep a history. This history is stored in the stack, so that when a user wishes to undo, the previous state can be restored. Similarly, when an item is undone, it is popped from the undo stack and pushed onto the redo, meaning when the function is called, previously popped commands can be restored and redrawn onto the canvas. When a new drawing command is executed, the redo stack is cleared, as a new action invalidates any previous stored redo states.

**Figure 2.6** Console screenshot showing data type of drawing commands.

A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated

**Figure 2.7** Screenshot of console post-drawing commands, in which shapes have been drawn, undone, then redone.

To ensure that users can save their designs, a “download canvas” button is provided to the user in which their drawing is downloaded in the browser that the site is hosted on. Similarly, a “download gif” button is also presented, allowing the user to download their drawing’s history and have it displayed as an animated gif in a new tab (this can also be downloaded if desired).

## 2.6 Suggestion Feature

Using the stored drawing history, a recommender can be developed to inspect and identify patterns in the user’s actions, such as repeated use of specific colours, shapes, or positions.

Suggestions can vary broadly depending on how we decide to generate them. A user may want suggestions that are aligned with their current task, but they may also want new and original ideas to be suggested to them to explore changes in their design and go down avenues that were previously ignored. Suggestions could include:

Shape & Position

* A drawing of the same shape (the most frequently used one)
* A drawing of a new shape (the least often used one)
* Place the drawing at the midpoint between all shapes on the canvas
* Place the drawing at the furthest point from all shapes

Colour

* A drawing with the most frequently used colour
* A drawing that combines the colours of all shapes on the canvas (additive)
* A drawing that picks an interpolation between all colours of all shapes on the canvas
* A drawing that picks the most contrasting colour from all colours on the canvas

With these in mind, we need to consider how these recommendations are selected and displayed to the user. To follow the mock-up designs provided in the earlier sections, a sidebar of visual presentations of these suggestions should be offered to the designer where they can select thumbnails of the canvas with the added suggestion to apply it to the main workspace. However, not all recommendations will fit in this sidebar, and attempting to do so would clutter the workspace and make it difficult for the user to focus on the main task of developing their designs.

A form of frequency analysis is a technique that can be applied to select recommendations that the user often selects, meaning the recommender is able to provide suggestions that are likely to be adhered to instead of offering unhelpful designs that clutter the UI. With this, the type of recommendation that gets selected is tracked, and if not immediately undone (it can be assumed that users will select recommendations and apply them purely to inspect the change, then may undo the canvas to restore their original design) the program will continue to suggest similar recommendations, prioritising suggestions that the user found interesting and helpful. The issue with this, however, is a cold start, in which there is no user data for the solution to start from. This can be solved by randomly selecting our starting suggestions with every new command until a suitable number of them have been applied to the canvas, allowing the program to begin to “tally” selected drawing suggestions. The initial set of randomised suggestions may also be diversified, meaning they cover a wide range of recommendation types, as opposed to only focusing on a particular parameter, consequently avoiding bias towards a specific type during the cold start phase.

An additional problem with this is the user may get caught in a feedback loop; the user’s current task may focus on a particular feature, such as colour, and consequently the solution will log a high frequency for colour suggestions and continue to provide the user with these. However, once their focus turns to shape or position, suggestions associated with these will be less likely to be displayed in the sidebar, as their frequency will be low. For this reason, balanced sampling should be implemented to prevent biases towards specific suggestions. Dynamically adjusting weights or periodically resetting frequency counts keeps recommendations new and inspiring as opposed to stagnant.

Another solution for this would be including a form of contextual relevance to suggestions, adding further rules to how recommendations are selected; if a user has drawn a shape that is greyscale, it is unlikely that they will want to see colour suggestions and would rather see ones related to alignment or geometry. This could involve analysis of the current state of the canvas and identifying relevant patterns or features rather than purely frequency-based analysis.

For this reason, it’s apparent that a hybrid form of recommender will need to be implemented for the suggestion feature to provide useful and relevant, but also diverse, recommendations to the user whilst avoiding bias and feedback loops.

# Chapter 3 Results

<Results, evaluation (including user evaluation) *etc*. should be described in one or more chapters. See the `Results and Discussion' criterion in the mark scheme for the sorts of material that may be included here.>

# Chapter 4 Discussion

## 4.1 Conclusions

Overall, the scope of my project was far too large. Darrell Viner and Harold Cohen’s works focus mainly on how machines can aid artistic endeavours through algorithmic problem solving and making decisions based upon pattern recognition and data analysis. There should have been a larger focus on this and how to develop the recommendation system by taking an iterative approach and getting feedback from testers to identify which algorithms provide the most relevant and useful suggestions. Due to “feature creep” (the continuous addition of new features) the progress of the product was slowed as development of key components would be postponed for smaller features to be implemented first.

## 4.2 Ideas for future work

In the case that this program is being reattempted or amended, using an API such as React would have made the UI far easier to implement and would have made it look sleeker and more professional.

Due to the lack of complete implementation, the broadness of my requirements and the encompassing title of a “creative workspace”, the scope of my solution was far too wide. The focus should be on the aided generations of the suggestion algorithm of the solution and how this solves the problem of iterative design. Creating a drawing platform from scratch consumed much of my time, leaving less of it to develop a fully functioning recommender system. In hindsight, a simpler plan would have been creating a fork of an existing open-source drawing suite, such as Aseprite [7] or Inkscape [8] and built off this with a larger focus on how a design suggestion system could be implemented.

# List of References

References:

[1] Paul, C. 2019. *Histories of the Digital Now*. [online] Whitney Museum of American Art. Available at: https://whitney.org/essays/histories-of-the-digital-now [01/02/24]

[2] Viner, D. 1974. *Untitled*. [Plotter drawing] At: Victoria and Albert Museum, E.468:165-2013. Available at: https://collections.vam.ac.uk/item/O1277843/untitled-plotter-drawing-darrell-viner/ [21/01/24]

[3] Shneiderman, B. and Plaisant, C. 2004. *Designing the user interface: strategies for effective human-computer interaction*. Pearson. Available at: http://seu1.org/files/level5/IT201/Book%20-%20Ben%20Shneiderman-Designing%20the%20User%20Interface-4th%20Edition.pdf [02/02/24]

[4] Johnson, J. 2013. *Designing with the mind in mind: simple guide to understanding user interface design guidelines*. Morgan Kaufmann. Available at: https://books.google.co.uk/books?id=\_dLVDwAAQBAJ&lpg=PP1&ots=t0EWonPs3g&dq=info%3AVzYO-mOHJVsJ%3Ascholar.google.com&lr&pg=PP1#v=onepage&q&f=false [20/02/24]

[5] Nichols, D. 2019*. Coloring for Colorblindness*. [online] Davidmathlogic.com. Available at: <https://davidmathlogic.com/colorblind>

[6] Maccorduck, P. 1991. *Aaron’s code: meta-art, artificial intelligence, and the work of Harold Cohen*. New York: W.H. Freeman.

[7] Igara Studio S.A. 2011. *Aseprite – Animated sprite editor & pixel art tool*. [online]. [www.aseprite.org](http://www.aseprite.org)

[8] Inkscape. 2004. *Inkscape: Draw Freely*. [online]. https://inkscape.org

# Appendix A Self-appraisal

## A.1 Critical self-evaluation

## A.2 Personal reﬂection and lessons learned

Due to the lack of complete implementation, the broadness of my requirements and the encompassing title of a “creative workspace”, the scope of my solution was far too wide. The focus should be on the aided generations of the suggestion algorithm of the solution and how this solves the problem of iterative design. Creating a drawing platform from scratch consumed much of my time, leaving less of it to develop a fully functioning recommender system. In hindsight, I would have created a fork of an existing open-source drawing suite and built off this with a larger focus on how a design suggestion system could be implemented. Aseprite or Inkscape drawing programs are both open-source drawing tools that may have been suitable replacements for attempting to create an entire drawing suite from scratch.

I’ve also seen the value in version control software, as I struggled momentarily with attempting to program new features and causing bugs/errors in areas of my code, then being unable to restore previous functions. If I had made branches for these experimental features, I’d be able to avoid these problems.

I believe my workflow would’ve been more consistent with the aid of these version control tools, as GitHub issues could’ve been assigned, and I could’ve used the built-in kanban board that is provided in the repository.

## A.3 Legal, social, ethical, and professional issues

### A.3.1 Legal issues

As p5.js is open source and the solution does not aim to be sold or redistributed beyond the academic purpose of this project, they may be fewer legal concerns regarding the use of this library. The use of this library adheres to the GNU LGPL that p5.js is licensed under and its creative commons license. As no sensitive user data (passwords, names, demographics etc) is stored other than their drawing commands and drawings themselves, GPDR regulations are being upheld.

### A.3.2 Social issues

Accessibility has been heavily considered for this art tool to tackle the issue of users with disabilities being unable to use the tool to its full capacity; colour-blindness filters have been run on the prototyped UI and on the program itself. However, those with motor disabilities may struggle with the mouse-only navigation of the interface. A future fix for this would be to introduce keybinds to specific buttons/functionalities in the solution.

### A.3.3 Ethical issues

An issue with a tool that provides suggestions and designs is the issue of ownership; there may be questions about who owns the resulting artwork and whether the tool would need to be credited as a co-author. The generation algorithm that is used may also generalise works, as if many users are being assisted with the same recommender, results could gradually be homogenized, leading to a lack of originality, consequently raising concerns about authenticity of the artwork.

To ensure transparency, it’s important that the users can see why and how the recommendations are made, hence why this information is available in the README.txt of the project repository. This means they can inspect the criteria for these suggestions and understand the factors influencing the tool’s recommendations. This helps artists control the degree of influence the tool has over their artistic vision and designs.

### A.3.4 Professional issues

As p5.js is being included through a URL where the library is hosted externally, issues may arise in the case that the hosting service experiences downtime or technical issues. This would make most of the solution unfunctional, however over the course of development the hosting service hasn’t created any issues. Having the library stored externally also means that version control is handled by the hosting service and maintainers of the p5.js open-source platform, allowing changes to the library to also impact the current solution. This means over time as p5.js functions change, the solution may also need to be amended to cater to updates or include new functionalities that users may wish to access within the generative art space.

To ensure users know how to use the system, comprehensive documentation should be provided to them to explain how to interact with the drawing tool and use its recommendations successfully.

# Appendix B External Materials

<This appendix should provide a brief record of materials used in the solution that are not the student's own work. Such materials might be pieces of codes made available from a research group/company or from the internet, datasets prepared by external users or any preliminary materials/drafts/notes provided by a supervisor. It should be clear what was used as ready-made components and what was developed as part of the project. This appendix should be included even if no external materials were used, in which case a statement to that effect is all that is required.>