Procedural music generation with Hierarchical Wave Function Collapse

Visualizing HWFC-generated music and "locking in" parts of the output for later reiteration

Daniel Lihotský
Supervisor: Rafael Bidarra
EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Daniel Lihotský
Final project course: CSE3000 Research Project
Thesis committee: Rafael Bidarra, Joana de Pinho Gonçalves

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Procedural music generation with Hierarchical Wave Function Collapse: Visualizing HWFC-generated music and "locking in" parts of the output for later reiteration

Daniel Lihotský
Delft University of Technology
Delft, The Netherlands

ABSTRACT

Procedurally generating a coherent and emotionally resonant piece of music can be very challenging. The Wave Function Collapse (WFC) algorithm is very effective when it comes to generating randomized patterns and maps that resemble an input sample. A version of this algorithm using a hierarchy of sections, chords and melody was used to create a model capable of generating music. In this paper, we extend the capabilities of this model to improve its utility and help composers more effectively utilize this music generation method to create engaging pieces. Our model offers improvements over previous methods by allowing composers to retain desirable elements of the music output while regenerating others, thus streamlining the iterative nature of music composition. We consider and compare different music visualization techniques and explore various user interface (UI) interaction methods to facilitate the effective selection of elements from the output. We designed and implemented this model with the conclusion that it significantly enhances the user experience and allows for creating a much more sound and complete piece of music compared to the original.

CCS CONCEPTS

• **Applied computing** → **Sound and music computing**; • **Human-centered computing** → **Visualization techniques**; **User interface design**.

KEYWORDS

Wave function collapse, procedural music generation, mixed-initiative, music visualization, user interface interaction

ACM Reference Format:
Daniel Lihotský. 2024. Procedural music generation with Hierarchical Wave Function Collapse: Visualizing HWFC-generated music and "locking in" parts of the output for later reiteration. In *Proceedings of Research Project (CSE3000).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

CSE3000, June 23, 2024, Delft

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Procedural content generation (PCG) is a field of computer science that has seen a rise in popularity in recent years with the introduction of large language models (LLMs) such as ChatGPT. It is important that we research the full potential of these PCG models to really see what value they can bring to our day-to-day lives. One of the prominent PCG algorithms is the Wave Function Collapse (WFC) algorithm [3]. This algorithm was originally created to generate randomized patterns and maps but has since been repurposed and modified to create a model for generating music, where instead of images, the output is comprised of sections, chords and melody [7].

This music generation model uses user-specified constraints to guide the algorithm. Mixed-initiative PCG is an approach where humans and algorithms collaborate to create content. This concept combines the strengths of human creativity and machine efficiency, resulting in a more effective and innovative content-creation process.

Creating music is inherently an iterative process, where producing a satisfactory piece requires multiple versions that slightly improve on one another. The current model, proposed by Varga and Bidarra [7], enables composers to define constraints and then generate an initial output. However, if the composer is not satisfied with the generated piece in its entirety, they must start over.

We propose a model that allows composers to selectively retain parts of the generated output for subsequent iterations. This approach enables continuous refinement of the piece across multiple generations, eliminating the need to start over after each iteration.

In the remainder of this paper, we explore how we can modify Varga and Bidarra's model, to allow the composer to lock in parts of the output for later reiteration, and how to visualize such output to facilitate effective selection of said parts.

2 RELATED WORK

2.1 Wave Function Collapse

The Wave Function Collapse Algorithm, developed by Maxim Gumin in 2016 [3], has garnered significant attention in the field of procedural content generation, mainly for its effectiveness in creating textures and patterns for video games and digital art. The algorithm derives its name from a concept in quantum mechanics, where a wave function representing a superposition of multiple states, reduces to a single state.

The basic principle of this algorithm is similar: a canvas of cells is created with each cell initially in a 'superposition' of potential

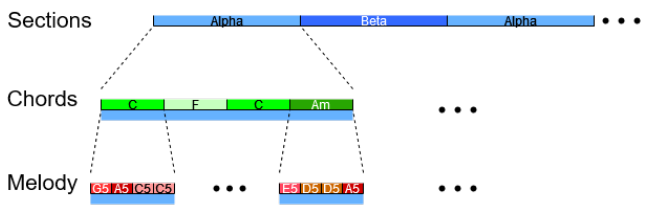

Figure 1: Visualization of the hierarchical structure proposed by Varga and Bidarra [7]

states. Through iterative steps, the algorithm progressively narrows down the possible states for each cell, based on neighboring cells and predefined rules between them, until a coherent solution is achieved. This process ensures that the generated output adheres to the local constraints defined by the input sample.

2.2 Mixed-initiative Systems

Mixed-initiative content generation is an emerging research area that explores collaborative content creation between humans and computers. This approach integrates human input to guide PCG. Langendam and Bidarra introduced a model incorporating mixed-initiative concepts into WFC [4]. These enhancements improve user interaction and control, enabling direct influence over the generation process. A mixed-initiative approach not only enhances the intuitiveness of WFC for users, but also expands its utility in creative domains such as game-level design and graphic arts. These advancements, which facilitate interactive algorithm steering, manual editing, and manipulation of generation parameters, have opened new avenues for creative expression and design flexibility in procedural content generation.

2.3 Music Composition with Hierarchical WFC

The original WFC algorithm was expanded upon by Alaka and Bidarra [1], through the introduction of a hierarchical structure to the canvas cells. Each cell can belong to a group that shares specific constraints, with the group itself having overarching constraints. For instance, in a video game map, a canvas of forests, plains and cities is generated, with each region itself being a canvas that consists of smaller elements, like buildings and roads for a city. Alaka and Bidarra's model illustrates how integrating a hierarchy of cells can enrich the design process.

Varga and Bidarra introduced a procedural music generation model utilizing the WFC algorithm with this hierarchical structure [7]. In their model, the canvas is divided into sections that contain chords, that contain melody notes. Constraints are inherited hierarchically, with each level deriving constraints from its parent (see Figure 1).

In this model, composers can establish specific constraints at each level. This can be achieved by directly assigning notes or chords to specific positions, or by defining relationships that neighboring cells must adhere to. Some constraint parameters may depend on values chosen for higher-level cells in the hierarchy. Moreover, the model allows for the definition of prototypes; selecting a prototype as the value for a cell triggers the imposition of corresponding constraints on the canvases beneath it in the hierarchy.

2.4 Visualization of Music

Music visualization has evolved significantly since its inception. Modern advancements in technology and computational methods have expanded the scope and complexity of music visualizations. An extensive survey of music visualization techniques has been conducted by Hugo B. Lima et al. [5]. It gives a clear overview of a large variety of music visualization methods grouping them by what main aspect of music they visualize (structure, pitch, harmony, melody, etc.) and how they do it (color, shapes, line graph, etc.).

3 VISUALIZING HWFC-GENERATED MUSIC

The selection of parts from the music output heavily relies on the exact way the musical elements are visualized to the composer. It is crucial to present the output in a manner that makes the selection of desired elements easy and intuitive. In visualizing music, we have an almost limitless array of options to consider. We need to narrow down our options to fit our specific use case. This can be achieved by defining the properties that our visualization model should exhibit to facilitate effective selection.

3.1 Required Properties

1. **Show chords and melody.** While it may seem self-evident, many music visualization techniques do not prioritize displaying the precise notes being played, instead focusing on other intrinsic properties of the music piece they are conveying [5]. Our model must clearly indicate the specific chord and melody notes in chronological order, as this is what our HWFC algorithm is producing. It should also distinctly differentiate between chord notes and melody notes.
2. **Clearly indicate the hierarchical structure.** Since our model is based on the one proposed by Varga and Bidarra [7], the generated output will be hierarchically structured, as illustrated in Figure 1. It is essential to clearly indicate to the composer which chords belong to which section and which melody notes belong to which chord.
3. **Notes and chords indicate their pitch.** The visualized chords and notes must clearly indicate their pitch. This can be achieved through explicit notation (e.g., C, D# for chords and A5, E5 for melody notes) or with some kind of symbol or their position on the canvas, similar to staff notation. This is crucial for composers to easily identify what different parts of the output comprise, without needing to play that particular part.
4. **The duration of the note is indicated.** The visualization must clearly indicate the duration of each note. This can be achieved through various means, such as the physical length of the note, specific symbols or shapes representing different durations, or numerical indicators.

3.2 Possible Visualization Techniques

In this section, we consider and compare a couple of music visualization techniques that most align with our goals. For each technique, we assess whether it satisfies the properties defined in

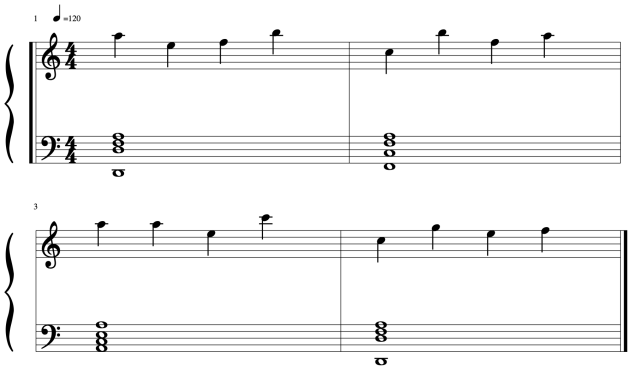

Figure 2: One section of a HWFC-generated music output visualized using staff notation

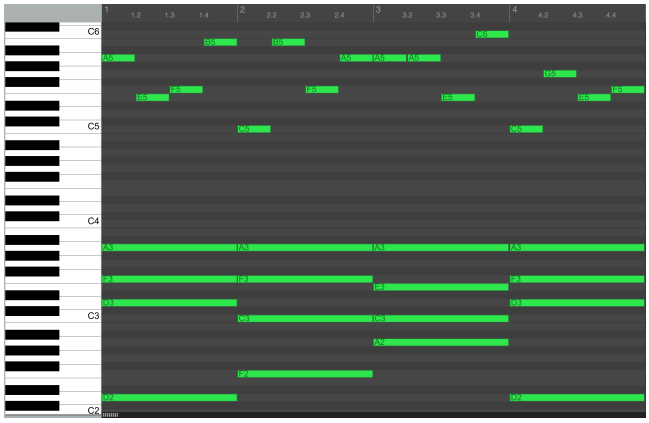

Figure 3: One section of a HWFC-generated music output visualized using a piano roll

Subsection 3.1, to what extent it does so, and how to modify it so it fully satisfies these criteria.

Sheet Music - Staff Notation. When discussing the visualization of music, the first method that often comes to mind is sheet music, more specifically staff notation. Staff notation is the traditional and most widely recognized method for representing musical compositions. It provides a detailed and precise notation of pitches, rhythms, dynamics, and articulations, allowing musicians to accurately perform a piece as intended by the composer. We visualized how a piece of music generated by our HWFC algorithm would look like using this notation (see Figure 2).

It shows one section of four chords, each with four melody notes. It clearly displays both melody and chords, satisfying property 1. The sheet is divided into measures (denoted by single vertical lines), and our chords are always a full note which spans one measure. This division visually distinguishes between chords and indicates which melody notes correspond to those chords. Our sections, however, are not any tangible element when it comes to music notation. When generating more sections, we would not see where one ends and another begins. This can be easily remedied by adding separators.

The time signature, in this case 4/4, determines the melody length (the top number/numerator), and the beats per minute (BPM) are indicated in the top left corner as ♩= 120. Properties 3 and 4 are also satisfied. Any person able to read sheet music, can determine the pitch and duration of the notes from their vertical position on the staff (the five lines) and their shape.

A problem with this visualization, however, is that it is too complex for our use case. Staff notation is designed to provide comprehensive details necessary for performing a piece of music, which are not necessary for our purpose. Another issue is that although whole notes (𝅝), which form chords, have the same duration as the four melody quarter notes combined, they do not occupy the same amount of space on the staff; they end much earlier. For our goal of selecting parts of the output, it would be more advantageous if the parts played simultaneously were always vertically aligned.

Piano roll layout. Let us consider a more simplistic approach, one that is commonly used for viewing and editing music digitally. This visualization technique is often used in digital audio workstations (DAWs). It displays notes as horizontal bars on a grid, with the vertical axis representing pitch and the horizontal axis representing time. Using a DAW, we visualized the same output shown in Figure 2, using a piano roll layout (see Figure 3).

It satisfies property 1. Property 2 is not as visually clear as in the previous example, but we observe that melody notes consistently appear above their corresponding chords. Numbering at the top demarcates where a chord ends and another begins, also indicating quarter note separations (with 1.2, 1.3, etc.). Section separation is not shown for the same reason as in our previous example, but this is easily addressable. Property 3 is handled similarly to our previous example. The vertical position of the note indicates the pitch. A nice addition is that this information is also explicitly written on the note itself. The duration of the notes (property 4) is shown with physical length along the horizontal axis. Consequently, parts of the output with the same duration occupy the same length on the axis. This eliminates the shortcoming of staff notation, where duration is only indicated with the shape/symbol of the note.

4 SELECTING GENERATED ELEMENTS

Now with a clear vision on what properties we expect our HWFC-generated music output visualization to have and which techniques we can use, we explore how we can enable the composer to interact with these visuals to select parts of the output. Selecting elements in a 2D space involves creating an intuitive and responsive interface that also allows for precision.

4.1 Selection Preferences

In this subsection, we discuss which elements the composer is likely to want to select together and provide reasons for these choices.

Selecting entire sections. Our music output is generated in sections. These sections encompass the underlying chords and can give them some higher meaning. Composers can define constraints specifically imposed only for that section. A section can represent for example an intro to a song or the chorus. Given that chord

progressions can vary greatly from section to section, the composer will often want to keep/regenerate entire sections for this reason.

Selecting chords and melody in isolation. Our music output combines the sound of melody notes being played over underlying chords. The precise combination of these two elements is what makes the music emotionally resonant, structurally sound, and engaging for listeners. If a composer does not like the sound of a subsection (a chord and its melody) of the output, it is most often times because the chord does not compliment the melody well or the melody is not pleasant. This is why it is crucial that the composer is able to keep/regenerate just chords, or just melody for a specified area on the horizontal axis.

Selecting individual notes of a chord. Each of our chords is comprised of four notes played simultaneously. We always select these notes together, because regenerating only a part of these would still result in a fundamentally different chord. Providing the ability to select these individual notes would unnecessarily complicate the UI interaction without offering additional utility to the composer.

Selecting individual notes of the melody. The smallest elements in our output are individual notes in the melody. It is evident why selecting the melody over a chord or over the entire section is crucial. Sometimes the melody over one chord might be much longer than what we saw in our examples in Subsection 3.2. With a longer melody there might be passages that a composer wants to keep but for example, does not like how it culminates. We want to give the composer the ability to select even the smallest parts of the melody to maximize utility.

4.2 Element Selection Methods

In this subsection, we compare various UI interaction methods and assess their suitability for enabling composers to efficiently select parts of the output in all the ways deemed crucial in Subsection 4.1.

Lasso (freeform) selection. This type of selection offers the greatest flexibility for choosing elements within a 2D space. Users can freely draw any continuous shape using their cursor, and upon closing the shape, any element contained within it will be selected. It facilitates precise selection of individual elements in the output, and for groups of notes forming a chord that do not need individual selection, those can always be selected together. However, this method is less efficient for selecting larger parts of the output because continuously drawing a shape for every selection could become tedious. This method is better suited for selecting elements arranged in irregular shapes.

Rectangular selection (bounding box). This selection allows the user to drag their cursor and the start and end points form the opposing corners of a rectangle. This method is simpler and more intuitive than lasso selection but offers less precision. Given that elements in the visualization techniques we explored in Subsection 3.2 are typically arranged in rectangular forms, makes this method better suited for our use case.

Select by label. Both methods mentioned so far are not optimal for selecting larger portions of the output simultaneously. We can streamline selecting entire sections by labeling them and allowing the user to click on the section label to select it. We can do the same to allow selection of the entire chord progression or the entire melody. Combining these functionalities enables users to, for example, select all chords within a section. However, this approach lacks the desired precision as it is not feasible to label every single element individually.

4.3 Deselection

The method for deselection depends on which selection technique we use. When using methods such as clicking a label or button to select, deselection is straightforward. The button or label will act as a toggle: if not every element encompassed by that label is selected, clicking it will select everything; if all elements are already selected, clicking it will deselect them.

When selecting with a shape, one approach is to have each new selection deselect everything else, akin to selecting files on a desktop. This method prevents users from selecting multiple non-continuous areas simultaneously. To address this limitation, users could hold the shift key to add a new selection to the already selected areas. However, this adds an additional layer of complexity. Another approach is to always add the new area to the existing selection, and then deselect by selecting an already selected area. The downside is that deselecting multiple areas can become tedious.

4.4 Visualization of Selection

There are several ways to indicate which UI elements are currently selected, all of which are applicable in our case. These are all fairly straightforward and self-explanatory, so we list them without going into detail.

- **Distinct colors** - Change the selected elements to bright or contrasting colors.
- **Color highlighting** - Change the background or border color of the selected elements.
- **Outlined elements** - Use thicker outlines or borders for the selected elements.
- **Highlighting with opacity** - Increase the opacity of selected elements while dimming the non-selected ones.
- **Shading effects** - Apply shadow or glow effects to selected elements.

5 DESIGN

From the visualization models discussed in Subsection 3.2, we have chosen to base our model on the piano roll layout, mostly because it is simpler and fits our use case the best.

The possible output notes generated by our model range from C(-1) all the way to G9. To accommodate this range, our canvas needs to be divided horizontally into 128 rows, each representing a distinct note. We have decided to label each note with its pitch value as seen in Figure 3. This is to allow the composer to compare pitch values of notes that are not immediately adjacent without needing to guess if they are in the same row or count the difference. Consequently, there is no necessity to label the rows or explicitly define the vertical axis.

Providing sufficient vertical space for 128 rows to accommodate legible text on a single page is impractical. We needed to introduce

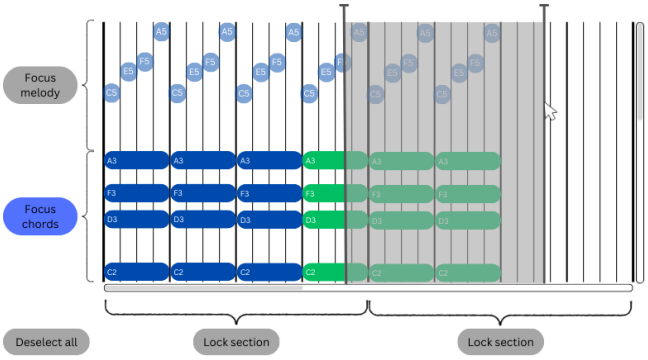

Figure 4: Diagram visualizing the main aspects of our design.

vertical scrolling. Typically, the generated output resides within the central 64 rows. Therefore, we have set our visible window to be 64 rows in height, aiming to minimize the need for extensive vertical scrolling in most scenarios.

The width of each note corresponds directly to its duration, allowing us to have a consistent time scale along the horizontal axis. Theoretically, the number of sections, chords per section and melody length are not limited, that means our canvas has to accommodate an output of any length. We can achieve this by adding horizontal scrolling if the output length exceeds what we are able to display on a single page. The canvas is divided into sections, chords and individual melody notes by vertical lines with varying thickness.

Under each section, we have a button that toggles the selection of that entire section. It is clearly indicated with a bracket, which elements fall under that button. These buttons scroll together with the sections as they move horizontally.

On the left side, we have two anchored buttons that toggle focus. The top button focuses on the melody, while the bottom button focuses on the chords. If neither button is activated, focus is on both melody and chords. Elements that are out of focus are visually indicated by reduced opacity. This is to allow the composer to constrain their current selection to only chords or only melody. Since chord and melody notes can overlap when they share the same pitch at the same time, this feature allows for distinct selection of these overlapping elements.

It is always clear whether we are selecting chords, melody or both. Therefore, we chose to have our element selection method be line selection, where the vertical position of the cursor does not matter. Similar to rectangular selection, except the rectangle always spans the height of the canvas. As the cursor is dragged along the canvas, the background color of the affected area changes, and selected notes are highlighted in a different color. Upon release, the background color reverts, while the selected notes remain highlighted. Subsequent selections of other areas do not affect previously selected elements.

The composer can deselect elements either by selecting them again or by clicking the universal *deselect all* button on the bottom left of the canvas. This button was included to simplify deselecting larger areas.

Once selected, elements are considered *locked in* for subsequent generations and remain selected in the new output until manually deselected. We have opted for the selection to indicate elements to retain rather than those to regenerate. This choice stems from the fact that composers typically wish to preserve only a minority of elements, and therefore need to select less.

This design was visualized using a simple diagram (see Figure 4). The green notes indicate selected notes and the gray window indicates the area currently being selected. The opaque notes indicate notes in focus and transparent are out of focus. Note that the size of the notes and distances on the vertical axis are not to scale.

6 IMPLEMENTATION

For the paper on which we based our model [7], Varga and Bidarra developed a web application enabling users to set constraints and generate music using HWFC through a graphical user interface (GUI) [2]. In their original implementation, the output is visualized in a minimalistic way in a small floating window at the bottom of the page. We implemented our design as an extension to this application. This extension introduces a dedicated page where users can view an enlarged version of the output, complete with all the functionality proposed in this paper.

The application was developed in TypeScript using the React framework. TypeScript's static type checking enhances code reliability and maintainability, while React's component-based architecture facilitates the creation of robust, scalable, and maintainable user interfaces. The application was deployed using Vite, a modern build tool that ensures efficient loading and hot module replacement during development.

We maintained consistency with the existing application theme by preserving the background image and button appearance. During the development, we closely adhered to our design, only having made minor adjustments and improvements. Notably, we expanded the section buttons to span the entire width of their respective sections. This allows users to select a section even when it is only partly visible, eliminating the need for brackets. Additionally, we integrated existing controls from the miniature version at the bottom of the page, facilitating actions such as generating, playing, and downloading the output.

One of the major challenges we faced was how to scale the notes depending on the width and height of the output, considering the size of the viewport of the browser. Our solution was to maintain a constant size for the notes (with width relative to duration), and dynamically adjust the size of the visible canvas according to the viewport. This approach ensures that notes and their labels remain legible regardless of the window size, with scrolling enabled as necessary to accommodate the entire content.

We also made some small quality-of-life improvements. For instance, when hovering over the canvas, the current column of a chord/melody note is highlighted, aiding users in determining where their selection would begin. While dragging, the selection window does not start and end at the exact coordinates of the cursor, but instead reflects the exact width of the elements being selected.

Figure 5: Screenshot of our implementation. Chords 4,5,6 are already selected and chords 7-10 together with their melody are currently being selected.

7 DISCUSSION

The general flow of the application with these new features would consist of the composer setting constraints, generating the output, locking in parts, and repeating. It would streamline the flow and in general make it easier to use if our added functionality was on the main page. This would necessitate a complete overhaul of the UI; to be able to put the canvas in the middle, while having enough room to accommodate everything that was already there. Unfortunately, this would take too much time and effort, and not get us closer to answering our research question, that is why we opted to put all of our functionality on a single separate page, without affecting the rest of the UI.

During our design phase, we came across multiple ideas that would improve the usability of the application and extend its capabilities, but we decided against including them, mainly because our focus was on the visualization and selection methods and these branched slightly outside of that domain.

- **Zooming in and out in the canvas** - A very useful feature with no discernible downsides, that would help with visibility and precision, but not strictly necessary in our case.
- **Start the playback from any point** - Currently, users can only initiate playback of the output from the beginning. If they wish to lock in a specific part of the output located towards the end but are unsure of the exact notes, they must start the playback and wait for it to reach that point. It would help the user with selection, to be able to start the playback from any point in the track.
- **Isolate playback of chords and melody** - The user can select chords and melody independently, so it would be beneficial to play them separately as well, enabling them to hear and evaluate the parts they wish to keep more clearly.
- **Play only locked in parts** - After selecting the desired parts of the output to keep, users should have the option to play only those parts to confirm if they are satisfied with their selection.

8 CONCLUSIONS AND FUTURE WORK

In this paper, we explored and compared ways how to visualize and select parts of a music output generated using Hierarchical Wave Function Collapse. We analyzed different visualization techniques, assessed their effectiveness, and evaluated user interaction methods to determine the most intuitive and efficient approach. Based on our findings, we proposed a design that incorporates these insights, aiming to give the user additional utility and improve the overall process of music generation, compared to the original model our research is based upon.

We implemented our design in the form of an extension to a web application, the one that was created to show the concepts of the original model. Our implementation has shown that our design effectively visualizes all of the important aspects of the generated output and allows for easy but precise selection of desired elements

to lock in for later iterations. It allows the user to iteratively improve on their generated piece of music, which facilitates creating much more elaborate and complete pieces than the original model.

Our new functionality currently exists on a standalone page in the application and the next step is to better incorporate it with all of the existing functionality on the main page. The user having the ability to fine-tune constraints and lock in parts of the output from the same page would streamline the creative process.

Another continuation would be conducting a user study. It would significantly enhance our research by providing empirical evidence on the usability and effectiveness of our design. By engaging with real users, we can identify usability issues, gather insights on user preferences, and ensure our interface is intuitive and user-friendly.

In conclusion, our work represents a significant advancement in the visualization and selection of music outputs generated using Hierarchical Wave Function Collapse. By enhancing the mixed-initiative approach of the original model, our design empowers users to interact more intuitively and effectively with the music generation process. This enhancement is crucial, as it significantly broadens the amount of control the user has over the generated piece, and reduces the reliance on the algorithm.

9 RESPONSIBLE RESEARCH

9.1 Reproducibility

Our research method consisted of surveying well-known music visualization techniques and UI interaction methods, comparing them, and evaluating their suitability for our goal. We then designed a model based on our findings. This process can be easily reproduced by anyone who has access to and knowledge about the model our research was based upon [7]. Unfortunately, this paper is not yet public, only submitted for publication. The details about this particular model that we provide in this paper might not be sufficient to reproduce the research, but once that paper is published, these concerns will be alleviated. Our implementation relies heavily on previous work, also done for the purpose of that paper. The repository of that model's implementation is public and can be found on GitHub [6].

9.2 Ethical Concerns

The primary ethical concerns regarding PCG revolve around the potential replacement of artists and more generally creative work done by humans. In our case, the model was specifically designed to aid composers, not replace them. The essence of a mixed-initiative approach is to integrate human input into generative models, ensuring that the human creative aspect is preserved and enhanced.

A significant issue with many PCG models, especially those based on AI, is that they function as black boxes. This means we often do not know how or why they make specific decisions. In contrast, our model is transparent and explainable. When a piece of music is generated, we can trace the algorithm's steps precisely to understand how it arrived at that particular output.

Compared to AI-based PCG models, our model does not require any input data that might be subject to copyright or involve users' personal data. Even the original WFC algorithm requires an input sample, which could be copyrighted or contain personal information, thereby limiting the use of its output to avoid legal issues or privacy concerns. Our model takes as input constraints and settings that are directly provided by the user, ensuring that it avoids these complications.

REFERENCES

[1] Shaad Alaka and Rafael Bidarra. 2023. Hierarchical Semantic Wave Function Collapse. In *Proceedings of the 18th International Conference on the Foundations of Digital Games* (Lisbon, Portugal) *(FDG '23)*. Association for Computing Machinery, New York, NY, USA, Article 68, 10 pages. https://doi.org/10.1145/3582437.3587209
[2] Anonymous. 2023. ProceduraLiszt. https://bit.ly/proceduraliszt_app
[3] Maxim Gumin. 2016. Wave Function Collapse Algorithm. https://github.com/mxgmn/WaveFunctionCollapse
[4] Thijmen SL Langendam and Rafael Bidarra. 2022. miWFC - Designer Empowerment through Mixed-Initiative Wave Function Collapse. In *Proceedings of the 17th International Conference on the Foundations of Digital Games* (Athens, Greece) *(FDG '22)*. Association for Computing Machinery, New York, NY, USA, Article 66, 8 pages. https://doi.org/10.1145/3555858.3563266
[5] Hugo B. Lima, Carlos G. R. dos Santos, and Bianchi S. Meiguins. 2021. A Survey of Music Visualization Techniques. *ACM Comput. Surv.* 54, 7 (July 2021), 29. https://doi.org/10.1145/3461835
[6] Pál Patrik Varga and Rafael Bidarra. 2023. ProceduraLiszt Repository. https://github.com/ProceduraLisztDevs/proceduraliszt
[7] Pál Patrik Varga and Rafael Bidarra. 2024. Harmony in Hierarchy: Mixed-Initiative Music Composition Inspired by WFC. Submitted for publication.