Muse: From Pallette to Playlist

Zariah Folkes
zlfa2021@mymail.pomona.edu
Student at Pomona College
Claremont, California, USA

Kaitlynn Gray
kagray@g.hmc.edu
Student at Harvey Mudd College
Claremont, California, USA

Deniz Bajin
dbzv2022@mymail.pomona.edu
Student at Pomona College
Claremont, California, USA

Eshanya Agrawal
eaaa2021@mymail.pomona.edu
Student at Pomona College
Claremont, California, USA

Abstract

Music-making is an intimidating industry, and many music lovers find the space hard to enter without heavy musical knowledge. There are limited tools that create space for casual creators and music-making novices to make non-professional music with a low floor. The barrier of music theory, complex terminology, sheet music etc. all make it very difficult for non-professionals to pursue music making in an experimental, low-stakes environment.

Muse is an app that uses people's existing knowledge of colors and applies them to mixing and layering sound. Applying skeuomorphism, we use the aesthetics of a DJ mixing booth and allow users to pick colors, change the elements within the colors, and mix different colors to create and modify sound. We hope to create an app that is fun and intuitive, and accessible to all age groups and people with different backgrounds with music.

Muse is an attempt to create a music making tool that does not require prior music knowledge and works with different instruments to create aesthetic sounds the user can save and go back to. In the process, the user builds their own relationship with sound and its association with colors. We found that users did enjoy this unique approach to sound creation and color association, and in a further expansion of this project we would like to create the option for users to make their own color-sound associations before playing with mixing on the DJ booth.

ACM Reference Format:
Zariah Folkes, Kaitlynn Gray, Deniz Bajin, and Eshanya Agrawal. 2024. Muse: From Pallette to Playlist. In . ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Music production has become more accessible in recent years. Beginner to professional musicians no longer need ties to record labels, recording studios, or even physical instruments to create music people enjoy. Softwares like Garageband are tools that allow non-music producers to overlay sound snippets into songs. However, distribution still remains a problem. While moving away from monopolized agencies and studios, websites like YouTube and Spotify also take agency away from the musician by commodifying streaming and setting ambitious ceilings on them for any monetary compensation. Capitalist platforms like these are primarily focused on income maximization. Conversely, In this new landscape of distributed music production with emerging technology, we believe that there is an upcoming space for casual creator systems that intend to make music composition accessible for non-professionals. Ph. D Katherine (Kate) Compton defines casual creator software as systems that exist "halfway between productivity software and entertainment software." Casual Creator systems prioritize the "enjoyment of process" rather than the "quality or control of the final artifact that is created"[9]. We argue that platforms should focus more on the enjoyment of art-making rather than emphasizing art-making for commercial purposes through casual creation systems.

AI-generated music is a key, albeit ethically ambiguous creator in this space of casual creation. There are issues with consent, copyright infringement, and credit being distributed between the

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

creator of the tool, the user, and the tool itself. A prominent example that highlights this issue is 'The Lost Tapes of the 27 Club'[7] that claims to bring light to mental health issues between the music community, but instead nonconsensually uses famous artists' voices to generate new music. Moving away from this, our goal was to build a tool that creates space for casual creators to make quick, fun, lo-fi music outside the professional space for music creation and commodification using computational tools that takes basic instruments and builds on them to create aesthetically pleasing sounds.

Many people have an intuitive knowledge of rhythm and sounds, but translating the ideas of music into real sound can often be inaccessible. Casual creators who want to play around with melodies often find it difficult with limited applications for beginner-friendly music creation. The process of merging and mixing different components and instruments with limited knowledge can also be overwhelming. If we found a way to connect people's already existing understandings of the world with the knowledge of sound, it would not only make music creation more accessible and instinctive, but also make the process of music creation fun. We settled on color as our body of existing knowledge since it is an easily accessible and widely known concept that people learn and view from a young age and is fundamental in people's existing understanding of the world.

The music production tools available right now have very steep learning curves. We conducted formative needfinding interviews, which revealed that users thought even apps like GarageBand, which are tailored towards novices, were inaccessible even though it is tailored to the beginner audience. In it, the software relies on a level of musical understanding that people find complex for those who do not have experience reading music. As a result of this vacuum in music casual music creation, there are swaths of people who want to make music and are interested but don't know where to begin.

Music learning and composition should be about supporting experimentation and exploratory workflow. Our app is geared toward an audience of novices who lack experience in music making by giving them a low floor tool that lets them tinker with sound and creativity while having some fun. As music is a universal language, it should be communicated in ways that many people can understand.

This led us to wonder: What if we can communicate music using color? By using primary colors – something that most people are familiar with, we aim to provide an accessible music production experience using a body of existing knowledge that feels intuitive. The process of using color-based methods to optimize memorization and material retention is used in university classrooms [10]. We aimed to return to the concept of color association and apply it to music composition. For example, while painting with watercolors, one stroke of the paintbrush creates a light shade. With every brush stroke, the color gets darker— the same process can be applied to sound: adding volume with every stroke of the musical paintbrush. A darker shade then would become a louder voice. Using skeuomorphism, we modified the familiar design layout of a DJ mixing console to create a connection between color and sound mixing as well as modification.

To evaluate the solution and our beta product, we use a mixture of two tools that provide us with a deep qualitative and quantitative user feedback. Sitting with users and making them use the think aloud protocol will help us understand the design flaws, what is intuitive and what users are struggling with within the tool. It will also give us an initial understanding of engagement and reactions to the tool. Based on the intuitions and what we see users struggle with, we will try to make our design animations more obvious and make tutorials accordingly. We will also add a Likert scale feedback form at the end of the user experience. This detailed feedback will help inform design modifications. and also help us understand if users relate with our color-to-sound allocations. The questions will ask users about their experience with the app concept in general, if the color to music associations made sense, what they had fun doing and creating, and what they'd like to see changed. This will help us understand changes and audiences we should target and operationalise. The likert scale being a simple 1-10 answer will also ensure more users take the feedback form and it is not needlessly time consuming for evaluators.

2 Methods and Tools

2.1 Formative studies and Design Goals

We started with the concept of colors and sound and tried to relate them to each other in different ways. We came up with the DJ mixing booth aesthetic and started thinking about how we can conceptualize and connect color and music theory in a meaningful way and what kind of users would be interested in such tools. We conducted four need-finding interviews with people who had varying levels of knowledge within music and color theory to settle the debate between our two concepts: using colors to make sound or using sound to make colors.

We talked to a music major (with synesthesia), an art and design major, a photography major, and a DJ. The overwhelming response when explaining them our ideas was to associate colors to create sound since colors are more easy to understand on a structural, intuitive level. By asking interviewees about music production apps they have checked out, we realized that sound creation apps usually have high learning curves in terms of understanding elements of music and changing them. This is when we started thinking about what elements of music we can introduce to the user through our app. We also saw that a lot of people had visual learning styles and felt more comfortable visually that auditory which seemed less intuitive to follow and decided to leverage visual learning in implementing sound design. Further, the interviews made apparent the fascination regarding music production and the intuitive nature of the DJ mix booth in creating it.

Something we realized when we talked to a person with synesthesia— a neurological condition in which she can see colors when hearing sounds- was that our concept of associating colors with sound might conflict with their conception of sound and colors. We decided that as an extension to our app long-term we would like to create a feature that gives users the option to associate their own colors with sounds.

We finally settled on using colors and color mixing to create music on circles while implementing sliders that changed color

shades and simultaneously also changed sound elements. We decided to keep our aesthetic retro and settled on specific fonts. We also decided to save color creations and generate fun names for them using a color API. Our final design goals became:

- Support amateur composers by limiting the number of components that go into the music
- Make interactions with the tool more intuitive by using skeuomorphism
- Support reflection-in-action by providing real-time auditory and visual feedback

2.2 Mapping Design iterations

Our main goal for users when using our tool was that they were able to use their subjective understanding of color to make universally good sounding music without the knowledge of music notes. We had ambitious goals for the tool in the first iterations of the design. Our paper prototype laid out the main components and the aesthetic of our prototype. We wanted a soundboard looking color pallet that when a color is clicked, the color is populated on one of the main "spinners" of the soundboard. When selecting a second color, the two colors mix, which plays the sounds associated with each color together. Our initial goals included giving users the ability to mix, layer, or overlay sounds. We also wanted to incorporate features like a play/pause button, a record button, a download option, and an undo system. The paper prototype (Figure 1) depicted the following:

- **Main mixing functions:** The color palette, soundboard spinners(the selected color), and a saved mixtures page
- **Sliders** labeled with color terminology that adjusted both color and sound
- **A selection animation** to facilitate more intuitive user input. The paper prototype gave insight on how we envisioned the overall look of our too, and was mainly referred to when creating future design iterations.

The second iteration of our design was on Figma. The goal of the Figma prototype was to map out the main user input interactions and to replicate our aesthetic goals on a digital platform. We expected that by creating our tool on a digital platform and simulating what happens when different buttons were clicked, it would give us more insight on what code would need to be implemented when we started programming. Our Figma prototype included three key components: a start page showcasing our finalized logo from this design iteration (Figure 2), a help page with brief instructions for using the tool, and the main page complete with full mixing and saving functionality, minus the sound integrations due to the limitations of figma. (Figure 3).

The main page was a web of interactions connecting various subpages accessible through clickable buttons (Figure 4). The prototype begins with a start page displaying the logo and a play button. Clicking play navigates users to the main soundboard page. Then, when a user selects a spinner it gets highlighted, allowing users to assign a color. Once both spinners are populated with colors, pressing save generates a smaller circle representing the mix of the two colors, visually indicating the combination has been saved. The difference between the paper prototype and the figma prototype was the number of colors available on the soundboard. It was at this stage of implementation where we decided to only have a few

Figure 1: Paper Prototype of Muse

colors and we added the concept of having people upload their own sound/color combination to our list of possible ideas to explore in this project.

2.3 First Draft of Code

Implementation was broken down into stages. At its first stage, we had some barebones starter code (Figure 5). We had some generic instruments as the sounds and we mainly focused on the color mixing and audio mixing features. The UI of the moveable sliders and a save button were implemented, but they weren't functional at the time. This was still a huge step, because this gave us a foundational piece of code to build on. At this point, Our goals in this design iteration were to create more user-intuitive code by creating the selection animation and fixing the help page, get the sliders to manipulate the music and color simultaneously, create the "save mixtures" feature, and to hone in on our overall UI design. Majority of our time was spent on making sure our main functions went properly, which afforded some challenges and sacrifices (refer to limitations and future work).

2.4 Final Design Iteration

The final design iteration consists of updated UI, functional sliders, colors, and a functional mixed colors page. The selection tool

Figure 2: The official logo created during the Figma design implementation

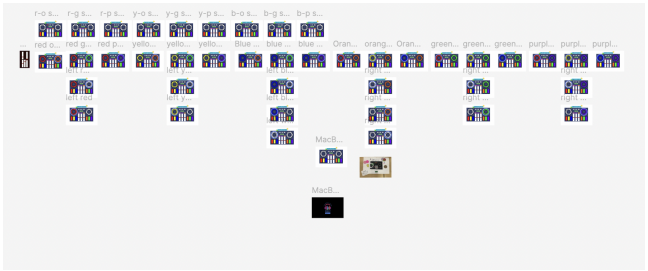

Figure 3: Muse Figma Prototype

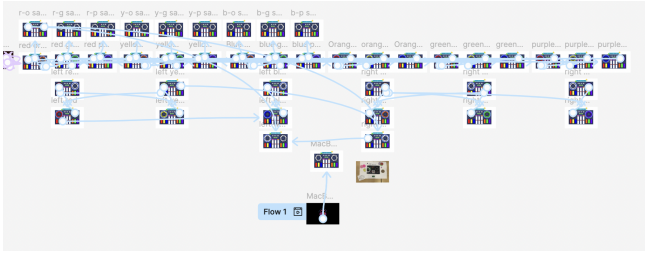

Figure 4: Figma Prototype Interaction web

was fully implemented and set so that the left spinner is automatically selected when you join the page. The UI was changed to a Black and Gray color scheme, purposefully simple so that the different selectable colors stand out. The number of sliders started off with 3 sets and ended up being 2 sets in the end. The save button and mixed colors page is partially implemented, ensuring people's color creations are saved but not storing music with them just yet. We changed the music for the final iteration in order for the tool to sound appealing when mixing two different sounds together.

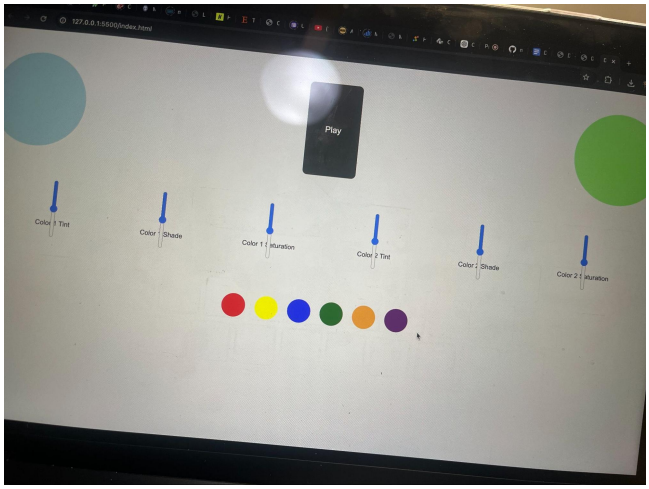

Figure 5: The First Coding Display of our Computational Tool

Inspired by the artistic computational tool inbflat.net, we chose diverse sounds that sounded nicely together instead of generic instruments.

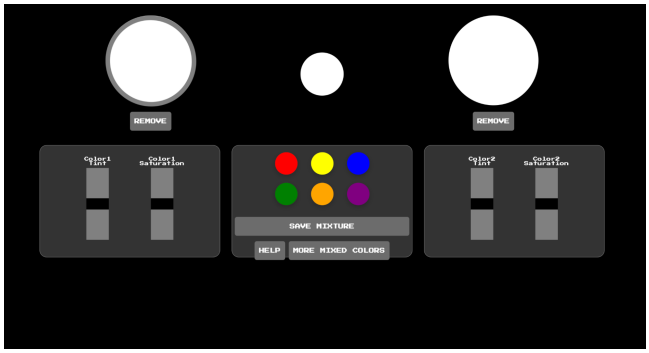

Figure 6: The final Display of our Computational Tool

2.5 Tool walk through

When the user opens the tool, they are greeted with the MUSE logo and slogan, "From palette to playlist." They are asked to click the start button to begin their color mixing journey (Figure 7).

Once the "START" button is clicked, the user sees the main page with the DJ mixer. The main page has the following elements: two large white circles on the left and right side of the page with gray "REMOVE" buttons underneath, a smaller white circle in the middle of the two large circles, a pair of sliders underneath each "REMOVE" button labeled "Color1 Tint", "Color1 Saturation", "Color2 Tint", and "Color2 Saturation," and color palate with associated buttons between the two pairs of sliders. The color palette contains six colors, red, yellow, blue, green, orange, and purple. They are each displayed in a circle smaller than any of the other circles on the page. There are three buttons underneath labeled "SAVE MIXTURE," "HELP," and "MORE MIXED COLORS." These buttons are all in the

Figure 7: Intro Page

same format as the "REMOVE" button. All labels are in VHS font. The background is black. The large white circle on the left side has a gray frame around it. This selection display just described is depicted in Figure 6.

There are two paths the user can follow: clicking the "HELP" button to get instructions on how to use the tool or experimenting with the tool. We have kept instructions and labels purposefully minimal in order to ensure user experimentation and creativity. The user clicks the "HELP" button, and their screen displays a page with white VHS font text on a black- gray background and a white button labeled "Back to Mixer." This text gives the user instructions on how to interact with the mixer (Figure 8).

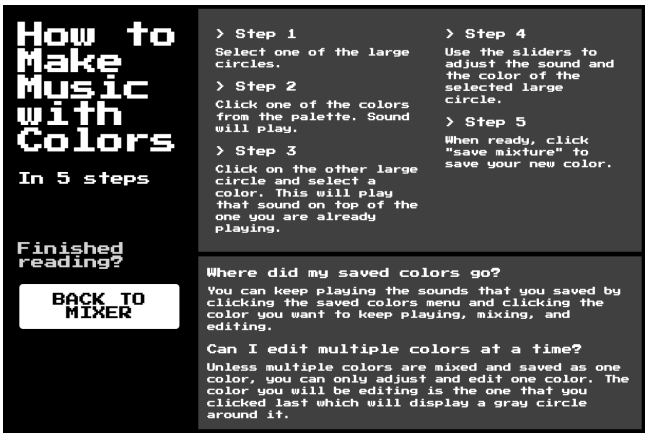

Figure 8: Help Page

Then, the "BACK TO MIXER" button takes the user back to the main page. Now, the user knows that they need to click on a color from the palette. When they click on a color, the color immediately displays in the left circle and an audio file starts playing. The smaller middle circle displays a color that is the mixture of the colors in the two large circles which are currently the color the user picked and white in Figure 9.

Then, the user can click any other color to hear all the audio files. This will replace the color in the left large circle. Once the user decides on a color-audio combination they like, they can start looking for another color to mix it with. To do this, they click the right large circle. Immediately, the gray frame moves to the right large circle. Now, they can pick a color for the right circle. When they click on a color on palette, this color immediately displays on the right large circle and its audio file starts playing. Now, two audio files are playing on top of each other and each large circle displays the color the user picked for them (Figure 10). As you can see, the middle circle displays the mixture of both the colors, showing how color elements interact in association with sound elements.

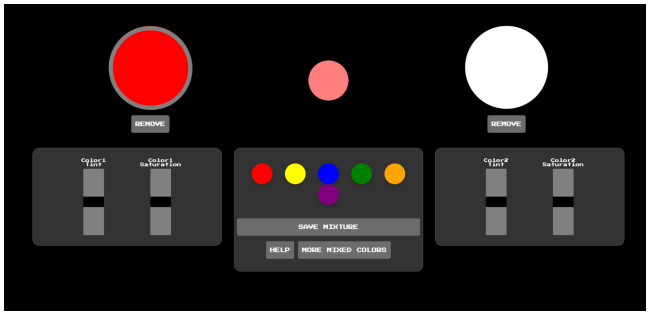

Figure 9: Mixing page with one color selected

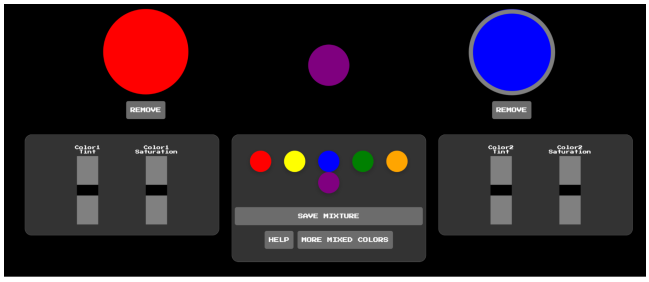

Figure 10: Mixing page with two colors selected

The two audio files that are playing on top of each other are the files that are attached to the colors displayed in the two large circles. The user can now either change the colors in the large circles by clicking the large circle they want to modify and choosing a color from the palette to replace the color that is in the circle, or they can use the sliders to modify the colors.

To use the sliders attached to a color, the gray selection frame must be at the large circle that is being modified. The user selects the left circle by clicking on it. There are two sliders: Tint and Saturation. If they move the tint slider up, the color lightens and the audio plays faster. If they move it down, the color darkens and the audio slows. While the sliders are moved, the mixed color displayed in the middle of the two large circles is also modified to reflect the updated mixture of the color in the two large circles since the sliders modify the color display in the large circle they are attached to. Once they find a tint and speed that they like, they move on to the next slider which is saturation. When they move the slider up, the color gets more saturated and the volume of the audio increases. When they move it down, the color gets less saturated and the volume of the audio decreases.

Once the user finds a saturation and speed they like, they are now ready to save the music that they created. To do this, they simply click the "SAVE MIXTURE" button underneath the color palette. Now, they can keep mixing the color that they just mixed by clicking the "MORE MIXED COLORS" button. This button takes the user to a different page titled "Saved Mixed Colors" that displays a color palette that consists of all the previously mixed and saved colors created by the user along with AI generated names for them (Figure 11).

Figure 11: Saved Mixture Page

If you scroll down underneath the palette, there is a gray button labeled "BACK TO MIXER." When the user clicks that button, the user is taken back to the main page that now displays white in the two large circles and the smaller circle in the middle (Figure 6).

2.6 Tool implementation

MUSE is designed to be a web application, so we used HTML, CSS, and JavaScript to build the base of our tool. To allow the team to work on the code base simultaneously, we created a GitHub repository to host our code. To aid in our programming process, we relied on assistance from ChatGPT [1]. This helped to speed up our programming process when we ran into bugs, and handle simpler code so we could focus on more advanced functions.

2.6.1 Website building: We used HTML to create four pages that users can navigate between: the introduction page, the main page with the color mixing, the help page, and the more mixed colors page that displayed previously mixed colors. We used CSS to modify the UI of each of these pages, and interactions with the elements, such as hover effects, using CSS animations. Since we wanted our project to have a retro feeling, we imported a VHS-like font from Google Fonts to use as our font style [3]. To create the glass effects behind our sliders and palette, we used a CSS glass effect generator from [2]

2.6.2 Audio Mixing: We used JavaScript to implement the audio manipulation. Using JavaScript, we assigned a music track to our main 6 colors. We sourced each of these audio files from videos on the website InBFlat [5]. This website features multiple music pieces played in B Flat that users can play in overlapping sequences to create unique music outputs. We chose sounds from this website to ensure they would sound pleasant when mixed together to improve the audio experience of interacting with our tool. To implement the changing speed and volume of each of these sounds, we used JavaScript's built in audio library to modify these components of the sound. Initially, we intended to have users control the pitch of the sound, as well. This proved much more difficult than expected. Despite using ChatGPT for help and dedicating multiple hours in office hours with our professor, we could not successfully implement this functionality. Later, after successfully implementing the speed and volume control, we learned that JavaScript's music library is incapable of modifying this aspect of the sound. If we chose to go forward with this idea, we would have had to completely abandon all of the code used for volume and pitch, and reprogram these functionalities from scratch using another library. In the interest of time, and to respect and value the progress made by the team on the audio features, we chose to scrap the pitch control and retain volume and pitch only.

2.6.3 Color Mixing: Pivoting to focus on color, we also used JavaScript to implement color mixing. First, we encoded each of the six main colors to have HSL values. HSL stands for hue, saturation, and light. We chose to encode the colors in HSL format since we could control the color by altering the hue value, then easily edit the tint and saturation by altering the light and saturation values, respectively. However, mixing colors in HSL led to interesting results. Essentially, we implemented color mixing by taking the averages between each of the HSL components between two colors. This proved effective for mixing the light and saturation aspects of the color. For instance, a color with 0 saturation (a shade of gray) mixed with a fully saturated color (a bright, non-gray color) should be a muddled version of the saturated color. In HSL, the resulting mixed color would have a saturation value that was the average of the unsaturated and fully saturated colors. This is slightly different in terms of hue. In HSL, a hue value is based on a color wheel. Thus, red is at 0 degrees, green is at 120 degrees, and blue is at 240 degrees, or an H value of 0, 120, and 240, respectively[4]. This means that when we try to take the average hue between two colors, we may not end up with the actual mix between the said colors. For instance, consider mixing red (H value = 0) and blue (H value = 240). If we were mixing red and blue paint, we would expect purple as a result. However, when we take the average H value between these two colors, we get 120, which is green. For a tool that depends on using the users' understanding of color to understand sound, this was extremely concerning. We wanted to make sure that colors behaved in a way users would expect so that they could focus on understanding sound, rather than becoming distracted by the color mixing. To mitigate this, we were directed to a post from Stack Overflow on mixing colors naturally in JavaScript, though this relied on colors being in RGB format [6]. RGB encoding determines what a color will be by altering the amount of red, green, and blue in a color (R, G, and B values, respectively). We used ChatGPT to generate a function that would convert our color from HSL to RGB for mixing. We then passed the RGB values of the colors we desired to mix into the Stack Overflow code to get a more natural result. In this way, mixing red and blue would now result in purple.

2.6.4 Displaying mixed colors: We used a locally stored JSON file to store our saved color mixtures. We displayed these saved colors using JavaScript to create an HTML div (division) modeled after a

circle with its respective color name underneath. We styled the circles such that they appear in rows of 4. We programmed the mixed colors page to only show 20 colors at a time to add a cap to how many colors we would store in our JSON file. This ensured that this JSON file took up a restricted amount of space on the users storage system. To add the color name of each color underneath, we used the Color API, which is a library of 1000 color names [8]. We can send the API the HSL value of the color. The API then determines which color in its library it is closest to. Once it determines the color, it can return information about the color, including the color name. While the names may not perfectly describe the color, it can provide users with a better description of the color. This would be significant for users with colorblindness, as they would be able to distinguish between the color mixtures they created.

3 Evaluation

To evaluate our tool, we conducted usability testing with five groups from the class. They were asked to think aloud while interacting with the tool as a group. This helped us understand the user's thinking process, assess the tool's intuitiveness, and identify pain points. Users were tasked with creating a melody by mixing two colors and adjusting them with the sliders. We also collected qualitative feedback through open-ended questions after interaction with the tool to gain deeper insights into their impressions and suggestions.

Our primary design goals were to make our tool accessible, intuitive, and enjoyable. The feedback we received largely supported these goals. One user remarked "everything is super intuitive," and another that "the interface is cool." Users appreciated the color-to-sound feature, describing it as "engaging" and "unique." Feedback showed users that appreciated the creative potential of the tool, describing it as "fun" and "unique," but also highlighted areas for improvement, such as adding more sliders for echo, reverb, etc., making the mixed color display using paint logic, enabling custom sound assignments, and allowing users to upload their own music.

4 Limitations

The final implementation of Muse looks slightly different than what we expected. Our original goal was to have sliders that manipulated speed, pitch and volume of a sound. However, the audio API that we used to build the program did not support the pitch-shifting feature, so we were unable to get a third set of sliders working using the API we had. In order to minimize the risk of having to restart, we decided to only lower the number of sliders. Another goal we set was the ability to mix more sounds using saved colors, and having the users implement their own colors. Due to timing, we were unable to implement these features. Finally, we wanted to users to be able to access their saved sound along with their saved color which is a function we are still trying to implement.

In the future, we want to implement the things that were in our original ideation stages, like the adding colors and the compositon with mixed color features. We also want to do more research and focus more on accessibility. Passionate about creating music with color, we want to add features that would support those who are colorblind.

5 Conclusion

Muse is designed for beginner music producers who want to gain exposure and experience playing with different components of sound. Muse is a creativity tool that serves as a first stage in understanding how different components of sound impact the eventual music produced. In our needfinding interviews and research, we identified that an understanding of music theory, how different components of sound work - such as pitch and volume - as well as knowledge of music production software, is required to produce quality sounds.

Muse applies our users' understanding of color to this new realm of sound. Muse creates a bridge between a color's tint and saturation, and a sound's pitch and volume, respectively. In doing so, we provide users with a visual representation of how these components of sound impact the output. With the insight we gained from our prototyping sessions, we have designed an intuitive interface that resembles a DJ board to reduce the learning curve in engaging with our tool. Thus, we have removed the barrier of needing to learn new software to produce music. These features allow us to achieve our ultimate goal of providing users an interactive experience to gain a better understanding of different components of sound in an accessible environment. In our case, it is pitch and volume. We hope that a project like Muse is the first step in a series of projects that help introduce beginners to music production through relying on non-audio cues. With these non-audio cues, users will be able to conceptualize how a music piece will sound through applying music to mediums they may be more familiar with. While Muse uses colors, future projects could create a tactile experience using vibration. Or, a project could expand our functionality to include components of sound outside of pitch and volume. As the future of music making will be fundamentally impacted by the influence of AI, we want to ensure that future generations of music makers retain these key music production skills. With the agency to be able to create music without AI, we hope to provide a sense of empowerment over generated music. In doing so, we aid in preserving the human touch in music.

6 Acknowledgments

We would like to thank Professor Li for all their support, advice and help through the different milestones of this project along with the detailed feedback and schedule they set up for us to achieve our goals. We would also like to thank all the students in the CSCI181DT classroom for their enthusiasm as well as insightful user feedback through the different stages of the tool. Their input was invaluable in understanding our design's limitations and improvements it needed.

References

[1] Chatgpt 4. Accessed: 2024-12-08.
[2] Css glass. Accessed: 2024-12-08.
[3] Google font: Press start 2p. Accessed: 2024-12-08.
[4] Hsl color reference. Accessed: 2024-12-08.
[5] Inbflat. Accessed: 2024-12-08.
[6] Mixing colors naturally in javascript. Accessed: 2024-12-08.
[7] Communication Arts. The lost tapes of the 27 club, 2023. Accessed: 2024-12-08.
[8] Josh Beckman. Thecolorapi: color conversion, naming, scheming & placeholders. Accessed: 2024-12-08.
[9] Katherine E. Compton. *Casual Creators: Defining a Genre of Autotelic Creativity Support Systems.* PhD thesis, University of California, Santa Cruz, 2019.

[10] I Diachenko, S Kalishchuk, M Zhylin, A Kyyko, and Y Volkova. Color education: A study on methods of influence on memory. *Heliyon*, 8(11):e11607, 2022.