GNG2101 Deliverable G

Accessible Digital Drawing: Prototype 2

Submitted by

Group B25

Adhish Maheswaran, 300133918

Gabriel Beaupré-Jacques, 0300119485

Trinity Bates, 300129927

Hong Yue Wang, 300105373

November 5 2020

University of Ottawa

Introduction

The goal of our project is to make a desktop application that provides an accessible digital drawing experience for our client, Madison, an artist who suffers from a fluctuating disability. During our study break, we developed the second prototype of our product. It is a fully functional Electron application that has large buttons that can select GIMP tools using macros. It also has a speech-to-text component that allows the user to search for tools using their voice. We recently had our third client meeting where we showed them our second prototype in order to gain some feedback and an approval of our design before moving on with the development of the final product.

1. Client Feedback

In our third client meeting we demonstrated our second stage prototype and made inquiries to continue to make a product that meets or exceeds client's needs.

There was only one issue that our client had with our prototype, which was the button colors should be changed to more muted colors and we need to avoid the usage of red. Changing our button design to muted colors and avoiding red must be done for the final product.

Since our client is new to GIMP and digital design, the client is currently satisfied with the buttons that we have created and has approved of an overall user friendly interface. Although there were no issues with the current choice of buttons, research can be conducted to find if there is a need to add any more buttons.

2. Second Prototype

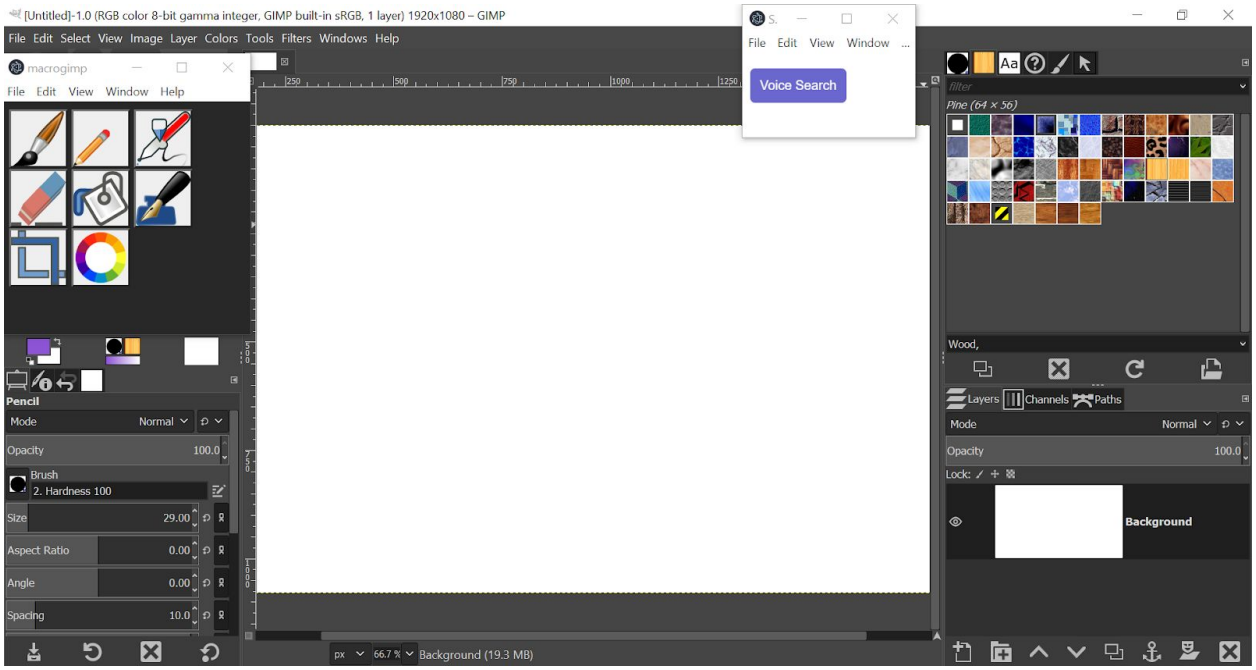
When we met with our client for the third time, we already had a functional second prototype to show them. There have been many significant changes made to our product since our last prototype. Our prototype is still a desktop application created using the JavaScript library Electron.

However, we are no longer using the program Macro Recorder to record macros since we discovered during development that Macro Recorder files cannot be opened directly, and have to be triggered by pressing a "Play" button in Macro Recorder's graphical user interface. This was undesired since we want a great user experience for our client, and having to install the Macro

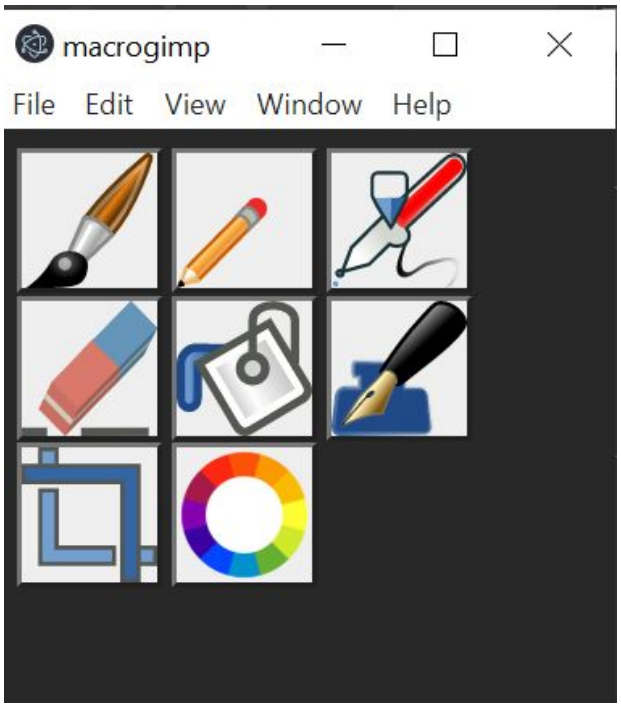
Recorder software in addition to our app and GIMP would be too much to ask from them. Furthermore, it would defeat the purpose of the tool bar if the client had to click on buttons in another application in order to make the buttons in our product work. Instead, we are using AutoHotKey scripts (.ahk). AutoHotKey is a free, open-source language for Windows. It allows the creation of macro scripts that can automate tasks such as mouse clicks and keyboard input. They can be opened without an user interface. However, the AutoHotKey software still needs to be installed. AutoHotKey files can also be converted to executable files. That way, the user would not need to install AutoHotKey to use the macros.

In our previous prototype, we used the free software called LilySpeech as the product's speech to text tool. During the development of our second prototype, we discovered that LilySpeech does not have an .exe file that could be opened by our application. Instead, the software was composed of many files in separate folders. The file structure of LilySpeech was very confusing, and we could not find any documentation for it, since the project was abandoned years ago. Therefore, we decided to create our own speech-to-text tool by using JavaScript's web speech recognition API. Since Electron uses a browser engine similar to Google Chrome called Chromium, our desktop application can access browser tools.

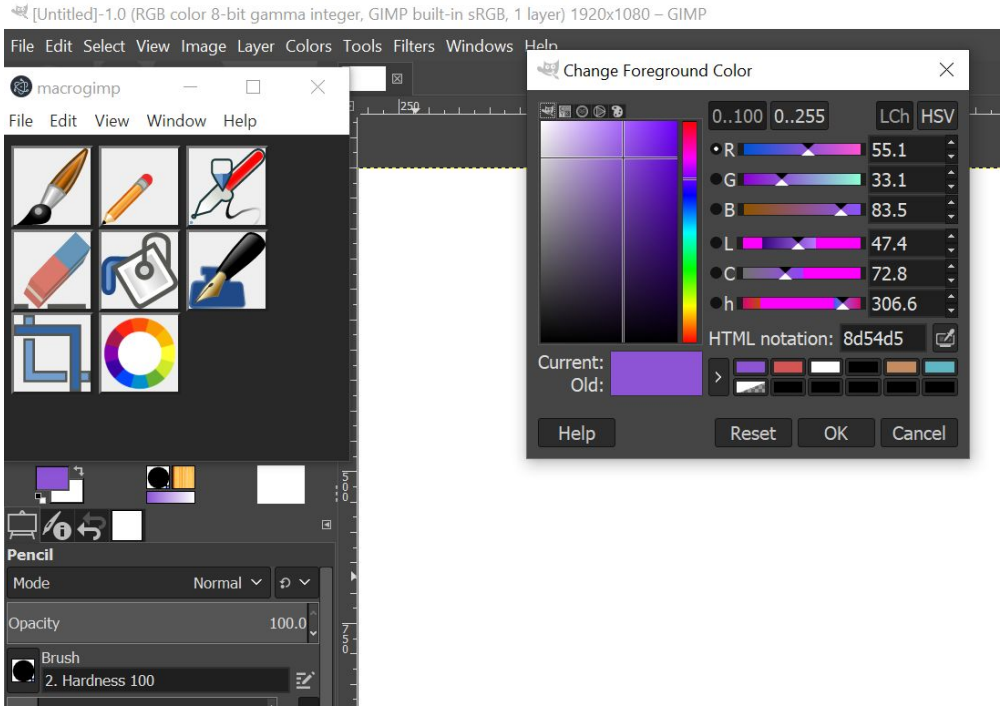
MacroGimp, our application, must be launched after the user has opened the GIMP graphics editor. When MacroGimp is launched, it opens 2 small windows that are overlaid on GIMP. We have programmed the application to make sure it always stays on top of the GIMP window. That way, the user does not need to constantly switch between applications.



The leftmost window of the application is the toolbar, where buttons are significantly enlarged in order to accommodate the client. Since the toolbar cannot contain every tool in GIMP, we prioritized tools that a digital artist would use. Our toolbar has buttons for Paintbrush, Pencil, Airbrush, Eraser, Bucket Fill, Ink, Crop and Change Color.

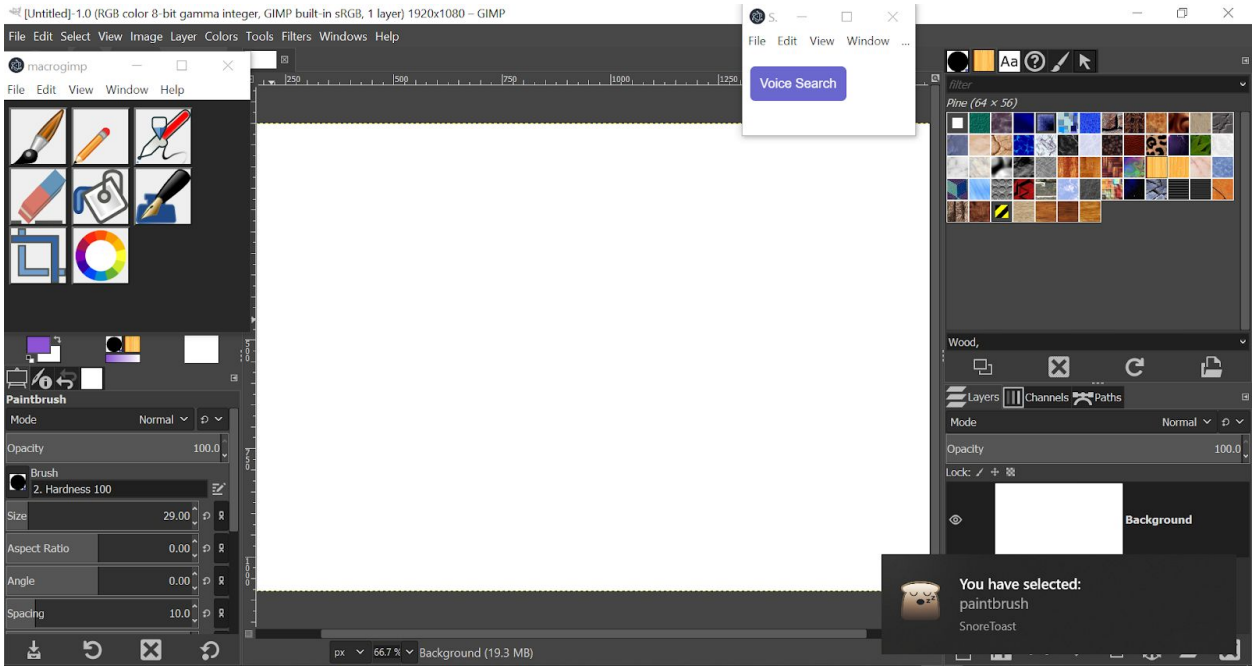


Whenever the user clicks on a button, an AutoHotKey file (.ahk) is opened. The macro file will then control the computer's mouse to click away from the Electron window and select the GIMP window. Then, depending on the tool the user selects, the macro will either use a keyboard shortcut to select a tool, or use the mouse to click a specific area of the screen.

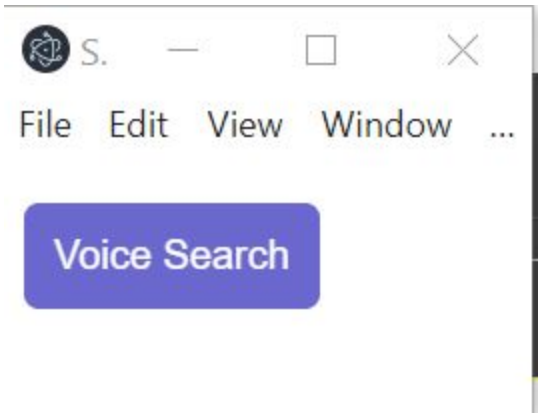


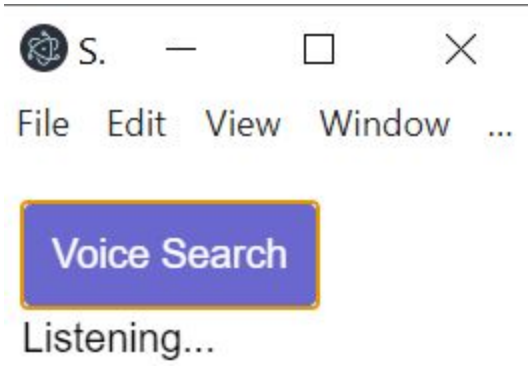
For example, selecting the "Change Color" tool will open a macro file that clicks on the widget that GIMP users normally click on to open the color wheel window.

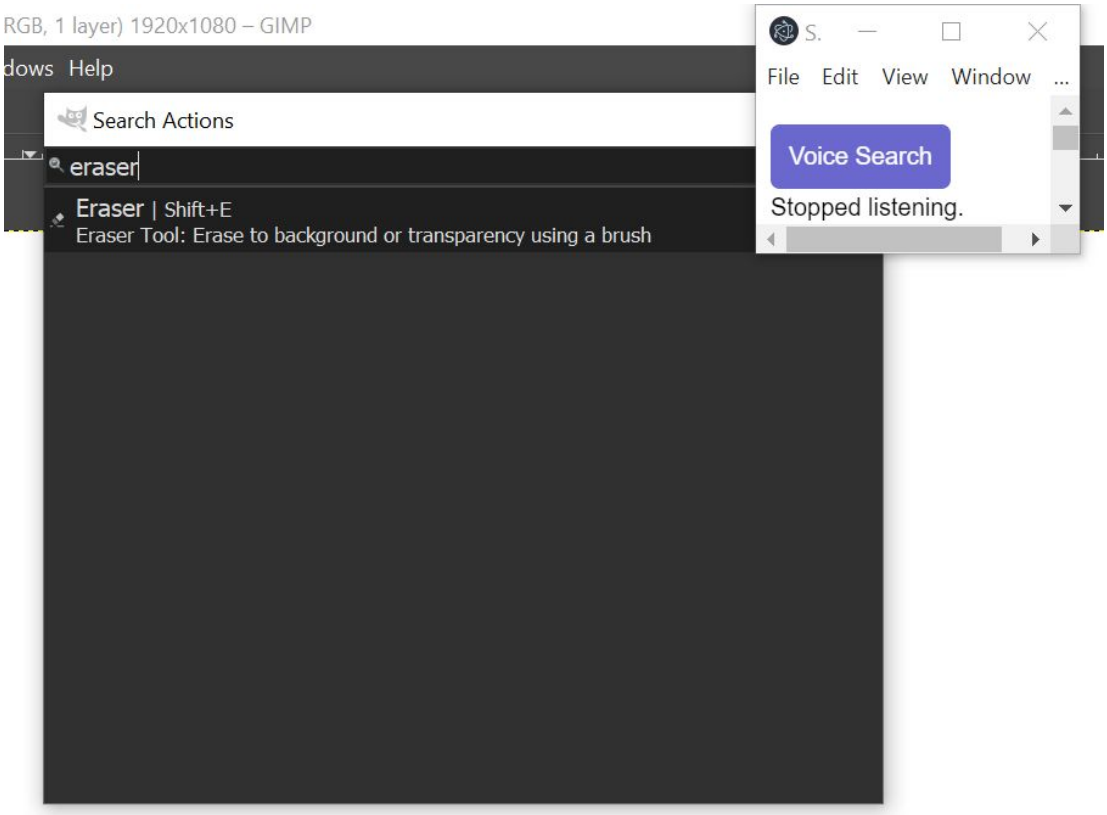
When a tool is selected, a desktop notification appears at the bottom right of the screen, confirming the user's selection.



The second window, which is the speech-to-text component of the application, contains a button that causes the app to start listening for voice input. The user can then talk into their microphone and their words will then be converted to text. The application will then copy and paste this text into GIMP's native search bar (that will be triggered by a macro file opened upon clicking the "Voice Search" button).







Afterwards, a list of tools matching the search word will appear under the search bar. The user can then select the tool they want.

3. Testing, Analysis and Performance

The way we proceeded with testing our prototype is to first analyze the specifications of our product in its current stage. The main specifications are mostly based on the user interface since the user experience is the main focus of our project. We must ensure that the RGB of the app are the same as expected and do not contain the color red. These specifications can all be tested by opening the application and examining the user interface. The button size must also be adequate.

There are also some aspects of the applications related to the user interactions that must be tested by simulating a digital drawing session in order to gather results on the user experience. In order to do that, we have asked several people to draw something on GIMP by using our MacroGimp application. We tested the amount of clicks it took for them to access a feature. This number should be minimal so the client does not get confused. We also observed the time it took for the average person to learn the application's basics. Furthermore, the application's specific features must also be tested with those simulations. These results are qualitative, and the only feedback we can get from the user is their general impression of the features.

Target Specification	Expected Results	Actual Results
RGB	(0-150) for all colours except black and white	(0-150) for all colours except black and white
Button Size	0-30% of screen occupied by buttons	6.25% of viewport width, 9% of viewport height
Amount of clicks to access a feature	1- 3 clicks	1 - 3 clicks
Time to learn basics of the product	1 hour at most and there should be no learning curve	0.5 hours at most and there should be no learning curve
Compatibility with the clients operating system	Yes	Yes

Size of executable file (.exe)	20mb	N/A (The application has not been converted to .exe yet. The current size of the code and the images is equal to 288mb)

Conclusion

The end goal of this project is to make an application that would make Digital Design more accessible and intuitive for our client, who has a fluctuating disability. The second prototype was presented on November 4th, 2020, during this meeting we received feedback from the client stating that the only necessary improvement to be made was changing the button colors to a more muted design and avoid the usage of red. Other than this issue, the client has responded positively to our user interface and the functionalities provided by our application(voice support, macro buttons). Although we've received very positive feedback from our client we plan to continue testing and making modifications. The testing that will be performed will include us measuring the speed and handling of the macros (scripts) in order to find any bugs preemptively, and to find ways that we can make our application smaller and easier to download as it currently requires Node js for Electron which is heavy. Testing the user interface and design will rely on communication with the client. As of now our product meets our target specification in all metrics except for file size. It is highly unlikely to meet our expectations for file size due to Node js but we will continue to further develop other aspects of the product such as the design, the next proptype will resolve the client's issue with button colors and there may be changes to layout or additions to improve our product.