

Redesigning the GUI for the HLsyn Speech Synthesizer

Samuel Rodberg

ECE-499 Capstone

Advisor: Professor Hanson

3/10/17

Report Summary

Speech synthesis refers to the artificial production of human speech. It is accomplished through either a mechanical, electronic, or software based system known as a speech synthesizer. For this capstone design project the focus will be on the HLSyn speech synthesizer. While the HLSyn software based speech synthesizer has not been updated in a long time, it does not mean that it does not have useful purposes in today's computing tasks.

Modernizing and optimizing HLSyn would allow for a useful tool to once again be used in both research and academic based learning situations. The new version of HLSyn will provide both new and updated features. One of these features is cross-platform support, specifically Mac OS in addition to Windows. Another feature that the updated HLSyn will have is an updated visual look to conform to the design standards laid out by Apple for Mac OS and Microsoft for Windows. Other major revisions will include a graphic-based input system for speech parameters, as well as a new method for inputting speech based parameters manually.

The main purpose of this report is to outline the design requirements for improving the speech synthesis software mentioned, and the proposed design alternatives, along with the suggested preliminary proposed solutions.

Table Of Contents

Table of Contents

Report Summary	1
Table of Contents	2
Table of Figures	3
Introduction.....	4
Background.....	6
Design Requirements	9
Design Alternatives	11
Preliminary Proposed Design.....	13
Proposed Project Timeline	18
Actual Project Time	20
Final Design and Implementation	21
Performance Estimates and Results	34
Cost Analysis	39
User Manual	40
Discussion, Conclusions, and Recommendations	43
Acknowledgments	45
References	47

Table Of Figures

Table of Figures

Figure 1: Design Decision Matrix with Functional Decomposition	11
Figure 2: Current Hlsyn Main User Interface	13
Figure 3: Proposed Hlsyn Main User Interface	14
Figure 4: Current Hlsyn Graphics Method	15
Figure 5: Proposed Graphics Input Method	15
Figure 6: Current Hlsyn Table Input Method	16
Figure 7: Proposed Hlsyn Table Input Method	17
Figure 8: Proposed Project Timetable	18
Figure 9: Actual Project Feature Implementation Timeline	20
Figure 10: Final Main Window	21
Figure 11: Main Window Menu Shortcuts	22
Figure 12: Revised Table Input Method	23
Figure 13: Hlsyn Installer Error	23
Figure 14: Inno Define Application Properties	24
Figure 15: Inno Installer Included Files	24
Figure 16: Hlsyn Installer	25
Figure 17: Hlsyn Running On Mac OS Using WINE	26
Figure 18: Wine Bottler Configuration Options	27
Figure 19: Wine Bottler Installer Error	28
Figure 20: Graphic Input Window Example	29
Figure 21: Setup Plot Graph Code Snippet	30
Figure 22: Starting Value Prompt	31
Figure 23: Connecting First Two Points	31
Figure 24: Log Output Of Plotted Points	32
Figure 25: Nmake Build Error	38

Introduction

Computer software is only as good as it is user friendly. No matter how incredible a feature set is, if it is not well designed then its user base will become frustrated. Software optimization takes many forms, ranging from optimized memory usage to power draw. With most modern software applications today using a Graphical User Interface (GUI), one of the most critical areas of focus in software design has become user interface design. A well designed user interface allows a user to intuitively navigate around the features they don't need, while at the same time utilizing the features they do need and applying them to their workflow.

While Hlsyn has an outdated user interface design, and lacks more advanced user interface functionality, it still retains a very powerful speech synthesis engine. The aim of this project is to optimize the GUI and functionality of Hlsyn in a manner that is helpful to the modern user.

To accomplish this, I will be updating Hlsyn to use a modernized layout that adheres to the latest design guidelines for the Microsoft Windows and Apple MacOS operating systems. Additionally, cross platform functionality will be enabled to allow for a wider use base. Furthermore, major features of Hlsyn will be updated to enable a smoother and easier use of the software, such as handling how speech parameters are entered. Updating the user interface and adding cross-platform support will allow Hlsyn to be used a modern tool for speech synthesis based research and discovery.

This report will lay out the path to this end goal by detailing the essential background information regarding not just speech synthesis but also approaches to optimizing the user

interface. It will then discuss initial design proposals, and the subsequent revisions. In the last section it will detail the test phase. The final section will be a reflection on the changes made and how they perform in comparison to the old version of Hlsyn.

Background

Creating artificial speech that can accurately replicate human speech has long been a goal of researchers around the world, even before the advent of modern electronics. During the late 1700's, a professor at St. Petersburg University by the name of Christian Kratzenstein created a mechanical device that could recreate each of the five primary vowel soundsⁱ. He accomplished this by using resonators with vibrating reeds to mimic the human vocal tract. Only a few years after Kratzenstein's creation, in the year 1791, a Hungarian inventor named Wolfgang von Kempelenⁱⁱ created an "Acoustic Mechanical Speech Machine" which was able to produce both single sounds and a few simple combinations of various vowel based sounds. In the middle of the 1800's, another inventor named Charles Wheatstone created a modified version of Wolfgang von Kempelen's speaking machine that could recreate the vowel sounds more clearly and most of the consonant soundsⁱⁱⁱ. Alexander Graham Bell found he was able to make his dog produce certain speech sounds by holding his dog and making him growl depending on the position in which he held his hands on the dog's mouth^{iv}. The era of electronic synthesizers, however, didn't begin until the year 1922 with J. Q. Stewart and his "electrical analogue of the vocal organs" device^v. It used an electrical buzzer to simulate the vocal folds, coupled with a pair of resonators to create resonances of the throat and mouth. The first device to properly be considered a speech synthesizer was exhibited by Homer Dudley at the 1939 World's Fair in New York City^{vi}. This device was called VODER (Voice Operating Demonstrator) and drew inspiration from the VOCODER (Voice Coder) developed by Bell Laboratories in the middle of the 1930's. The VODER used a wrist bar for choosing a voicing or noise source and then a foot pedal to control fundamental frequency. While the machine was

flawed, it drew the attention of the scientific community to the reality that accurate human speech could be produced artificially by a machine. While other improvements were gradually made, it wasn't until almost 40 years later at the beginning of the 1980's, that a cheap integrated circuit based synthesizer called Votrax arrived. This chip consisted of a cascade based formant synthesizer^{vii}, meaning the output of one of the resonators served as input into the next resonator. In this same time period basic implementations of software synthesizers modeled on hardware units began to come out. Apple included a basic speech synthesizer that could talk to the user called MacInTalk in 1984^{viii}. Microsoft did not include similar speech synthesis capabilities until Windows 2000. While Hlsyn allows for a variety of different types of synthesizer parameters, the main focus is high level speech synthesis parameters. Examples of high level speech synthesis parameters include acoustic frequencies that are present when a person speaks. Additional examples include parameters that model various aspects of the mouth and tongue.

One of the issues regarding the redesign is the economic impact. While the redesign itself will have not have any associated costs, the problem is the original Hlsyn program developed by Sensimetrics Cooperation was a commercial product. Charging a high fee for a new version of Hlsyn would severely limit who would have access to it and the environments in which it could be used to perform research. As a result, given that the product ceased development in the early part of the year 2000^{ix}, the hope is the company would give permission to release the new version of Hlsyn under one of several available open source licenses.

Another major issue with Hlsyn is its sustainability. In its current form Hlsyn has gone over 15 years without update or revision. A project that is based on Hlsyn that is no longer active is called The ProSynth Project. The ProSynth project developed an application called ProSynth2 which allows for users to import data generated from Hlsyn but requires a license for Hlsyn to be obtained first^x. The problem with this is that Hlsyn licenses are not available at all anymore. If the new version of Hlsyn is allowed to exist as an open source project, it would be hosted on a site like GitHub, where the source code could be maintained and built by the community. This would also allow it to be used in any market where a person or institution was interested in speech synthesis research applications.

Speech synthesis also raises an important ethical dilemma that must be considered. More and more applications nowadays are using voice recognition as a security tool to keep user information private and safe. The problem with speech synthesis is the potential to have a voice that is synthesized so accurately that it tricks the security software into believing it is the user's actual voice^{xi}. However, the reality of this happening is small as the software is not yet accurate enough to consistently replicate a human voice capable of bypassing a speech recognition security system.

Design Requirements

The new version of HLsyn will be an important update over the original version, as it will serve to optimize, as well as modernize, the user experience. The main requirements by the customer (Professor Hanson) were the following:

- Add a graphics-based input method for speech parameters, in addition to the current table based method
- Add cross-platform support to include Mac OS in addition to current support for Microsoft Windows
- Update user interface to conform to platform guidelines laid out by Apple and Microsoft for Mac OS and Windows, respectively
- Easily distributable software system with simple application installer

The first design requirement is a graphics-based input method for speech parameters as opposed to the current method of inputting parameters in a table. This would allow the user to quickly plot graphs that represent the various HLsyn speech parameters and how they are varied over time. The x-axis of the graph would correspond to the time at which the speech parameter is being manipulated (for example, manipulate the HL formant value f1 at time equal to 2 seconds), while the y-axis would correspond to the value of the speech parameter being manipulated (continuing with the earlier example, f1 at time equal to 2 seconds has a value of 300). The data drawn onto the graph would then be placed for the user to view, with the option to directly synthesize the sound generated from the drawn graphic.

The second design requirement is the need for cross-platform support. The last major revision of HLsyn was supported by Microsoft Windows 2000. It will need to be modified to

work on the latest version of Microsoft Windows, which is currently Windows 10. Additionally, it does not run on MacOS, which is the primary operating system used by my main customer. Support for Mac OS is also important because many academic institutions computing labs and personal computers use Apple's MacOS operating system.

One of the additional design requirements is an updated user interface. Currently, the Hlsyn user interface is very outdated. It is clunky and cumbersome to use. The goal is to create a modern user interface based on the design guidelines set forth by both Microsoft and Apple, depending on the platform that will be targeted in each version. Studies have shown that it is easier to use software when it has a familiar look and feel compared to other software programs that a user is already accustomed to^{xii}. Productivity and ease of workflow will also increase as result of a modernized layout.

An additional desired feature was a method of easily distributing the changed software package. Hlsyn does come with an installer package; however, the installer will not work on Windows 7 when tested. While one approach that was considered involved zipping up the necessary files and distributing the resulting zip file, this is not the most user friendly option. The most suitable approach would be to include an .exe based installer for Windows. For Mac OS a .dmg archive would be provided that the user could easily unpack and move Hlsyn to their applications folder.

Design Alternatives

	PyQt 5.x and Python 3.x	<i>PyQt 4.x and Python 2.7.x</i>	wxPython	GTKPython	Tkinter	Bitmap Graphics	<i>Vector Graphics</i>
Feature Set	4	4	3	4	1	2	4
Ease of Use	4	4	2	2	1	1	4
Compatibility	2	3	2	2	3	5	5
Total Score	10	11	7	8	5	8	12

Figure 1: Design Decision Matrix with Functional Decomposition

For the new version of Hlsyn the core of the redesign will focus around a modernized GUI toolkit. The GUI toolkit that was selected was Qt, which is currently developed by the Qt company^{xiii}. Specifically for this project, the Python language bindings for Qt known as PyQt by Riverbank Software^{xiv} will be used. Version 4.x of PyQt coupled with version 2.7.x of Python will be selected as the language version and binding version respectively. The alternatives for GUI toolkit and python version were as follows: while version 5.x of PyQt and python version 3.x both exist respectively, the older versions of both were chosen to maintain compatibility across all operating systems. Specifically, the way certain Qt graphically-based events are handled can differ greatly on Windows versus MacOS when using Python 3.x coupled with PyQt version 5.x. Python overall was chosen as the language due to its relatively high performance, but also for its ability to rapidly prototype applications and simplicity when debugging code problems. While Python has several available language bindings, most notably wxPython, PyGTK, and its own native Tkinter, all of these had more downsides than upsides compared to PyQt. While wxPython and PyGTK are both very capable GUI toolkits, they are not updated frequently and do not have the most up to date look and feel. For Python's native Tkinter the

GUI has a very unappealing look and feel and is also very lacking in feature set beyond very basic user interfaces.

The Graphical Input method will use vector based graphics to draw the line. While bitmap graphics were considered like the ones currently used in Hlsyn, bitmap graphics by nature would create less accurate representations of the speech parameters desired. As a result the vector graphics will allow for more accurate extrapolated data.

Proposed Preliminary Design

Before we can propose any design, we must first take a look at what the current version of Hlsyn looks like. The current layout of Hlsyn can be seen below in Figure 1:

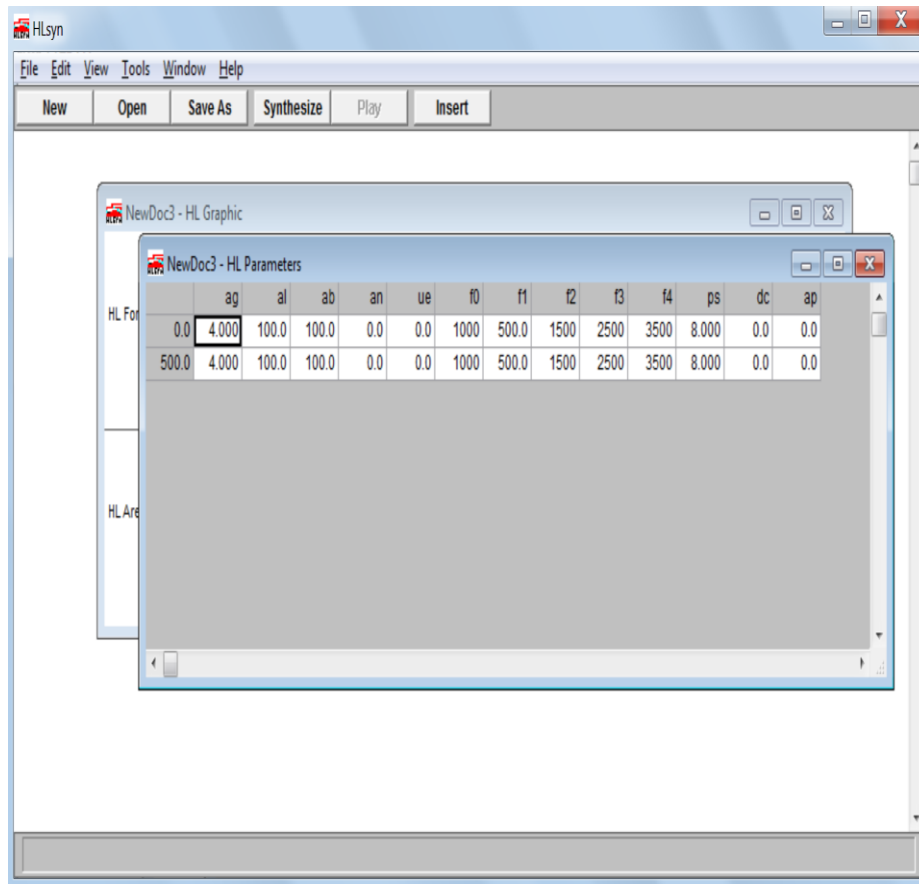


Figure 2: Current Hlsyn Main User Interface

Basic principles of user interface design state that an applications functions should be visually self-explanatory (for example, if you see a disk icon this should save the current file the user is working on), you should only show the user buttons they can actually click on (for example, do not show the user buttons that aren't selectable currently as seen in the "Play" button in Figure 2). Addition principles of user interface design state windows shouldn't overlap and hide information from the users. Taking these principles into account, the problems with the

current design of Hlsyn quickly become apparent. Immediately, it can be seen that there are overlapping windows, which violates usability principles. Also, the current layout does not allow for different to ways to view each of the program's windows at the same time. One example would be to have windows in a grid arrangement so they are not stacked on top of each other. Tweaking several types of speech parameters can quickly lead to window overload, and confuse, rather than aid, the user. To solve these problems a proposed modernized user interface based on PyQt can be seen below in Figure 3:

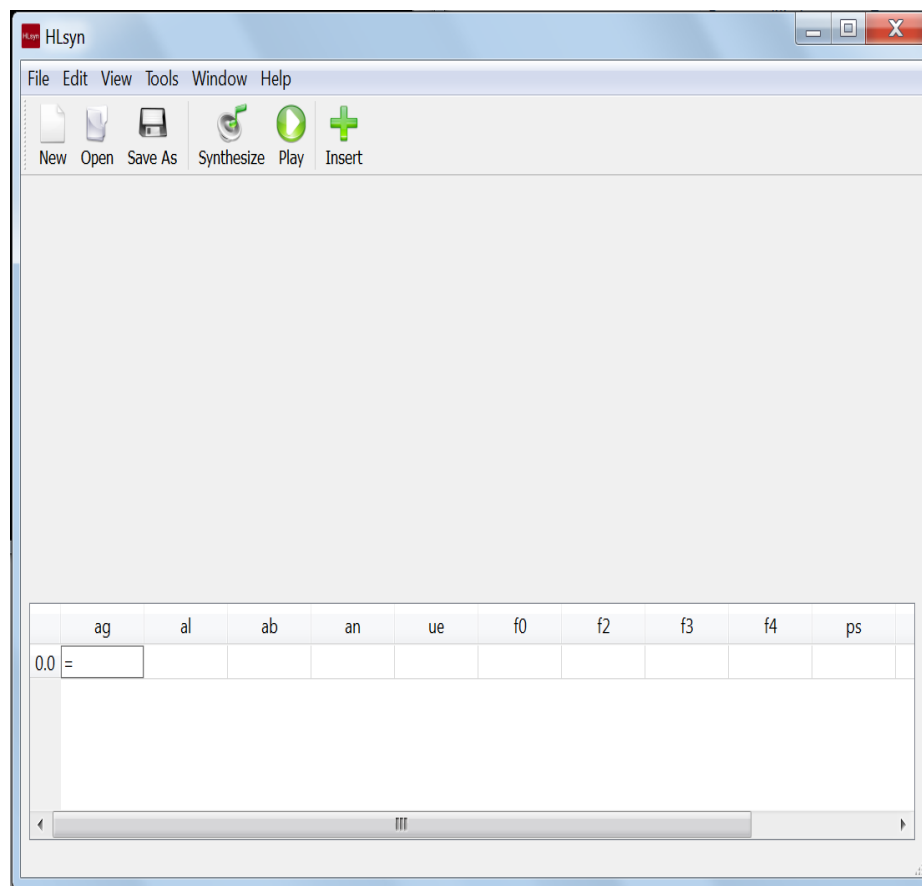


Figure 3: Proposed Hlsyn Main User Interface

This new version has cleaner visuals, which gives the user a better idea of what is going on and what the various buttons such as New, Open, and Save As do. Additionally, the toolbar is

movable to allow the user the ability to configure the layout in way that best suits their work flow.

For the graphic based input method for Hlsyn parameters, the current graphical display uses bitmap based graphics that can often make it difficult to discern exact data points. An example of this can be seen in Figure 4:

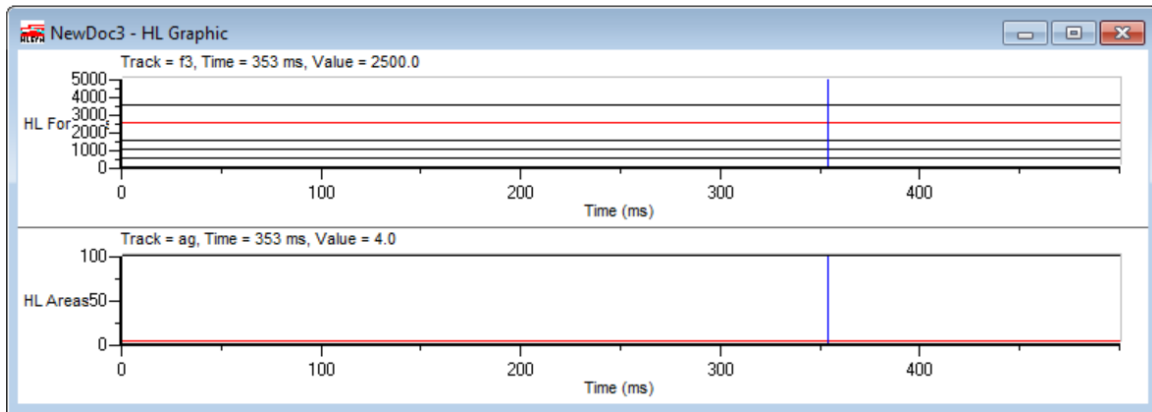


Figure 4: Current Hlsyn Graphics Method

While the current graphic methods displays a graphical representation of data inputted into the table, the goal is to allow the user the choice of either using the use of the table to input parameters by instead drawing points on the graph to represent the data. To accomplish this a system of drawn vector graphics would be utilized similar to the way they are drawn in the popular software applications Paint.net and Adobe Illustrator:

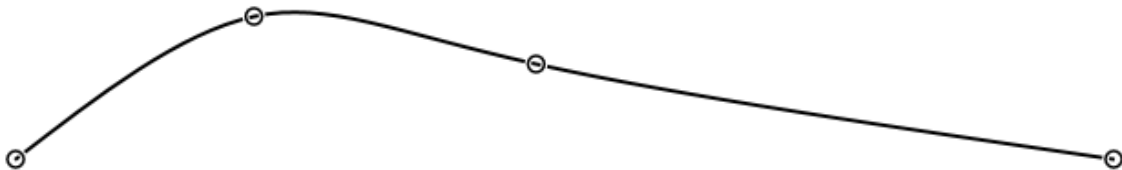
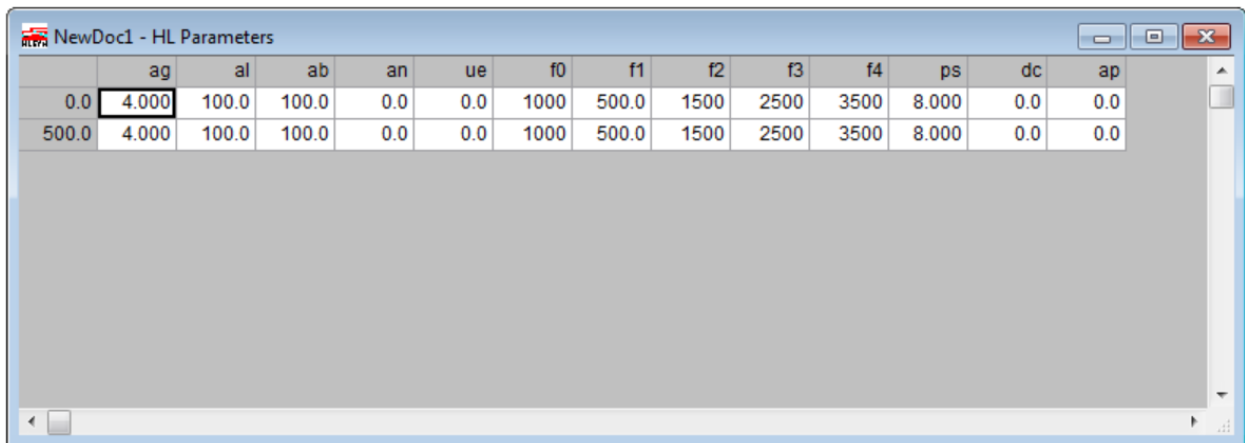


Figure 5: Proposed Graphics Input Method

The user would simply click on a graph that initially had no line segments, then they would click points and start sculpting the line segments to represent the parameter variation they want to represent. The x-axis of the graph would represent the time value at which the user wants to manipulate a speech parameter, while the y-axis would represent the value of the speech parameter that is being manipulated. The values from the graph would then automatically be entered into the table which the user could then use to synthesize the speech sounds.

Additionally, the user would be able to use the table to fine tune the data generated from the graphic they drew.

A final part of the proposed design is to redesign the table input method, which can be seen below in Figure 6:



	ag	al	ab	an	ue	f0	f1	f2	f3	f4	ps	dc	ap
0.0	4.000	100.0	100.0	0.0	0.0	1000	500.0	1500	2500	3500	8.000	0.0	0.0
500.0	4.000	100.0	100.0	0.0	0.0	1000	500.0	1500	2500	3500	8.000	0.0	0.0

Figure 6: Current Hlsyn Table Input Method

The current method has several flaws. One of these is the inability to change the time parameter once a row has been created (Seen as the 0.0 and 500.0 values in the left most table column in Figure 6 above) without deleting the entire row. Another proposed improvement is the way pivot points are created and handled by the table. Pivot points are where two time values can be selected in the table and the time values of the parameter in-between the two

selected times will be a linear interpolation. Pivot points are crucial to creating differently-shaped speech curves. In the current implementation, pivot points are selected by double clicking on a cell in the table or the parameter the user wishes to alter at a time specified. To address the issues mentioned above a different design can be seen in Figure 6:

	ag	al	ab	an	ue	f0	f1	f2	f3	f4	ps	dc	ap
	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figure 7: Proposed Hlsyn Table Input Method

As can be seen in Figure 7, the new input method uses a series of input boxes and check boxes to input parameter values, time values, and to mark pivot points. This allows for users to change values very easily and not have to delete entire rows or wonder why a pivot point suddenly was not acting as expected.

Proposed Project Timeline

The proposed week by week project timeline can be seen below in Figure 8:

Week	Milestone
1	Implement main window Obtain test feedback
2	Implement graphic input method Obtain test feedback
3	Implement revision to table input method Obtain test feedback
4	Bring in testers to test main components
5	Refine main components based on feedback
6	Comparison based testing
7	Continue comparison based testing
8	Implement feedback from comparison testing
9	Final testing and minor adjustments
10	Packaging and distribution phase

Figure 8: Proposed Project Timetable

In week 1, the goal was to implement the redesigned main window of Hlsyn as proposed in Figure 3. I also was going to test the implementation, and after obtaining feedback from my customer, apply the changes accordingly. In week 2, the graphic-based input method was to be prototyped and implemented. It was also going to be tested, and then changed further based on

the customer's feedback. It recognized that it may have potentially taken longer than one week for this feature to be fully implemented. In week 3, the focus was to be revising the current table based input method with the design proposed in figure 7. For this milestone the other goal was to be testing thoroughly, and obtaining feedback to apply from the customer. In weeks 4 and 5, the goal was to bring in a group of people to test the main revisions, and features that were implemented. This group of people just consisted of students, tasked with giving their initial design impressions and experiences using the features that were implemented. In weeks 6 and 7, comparison-based testing was to take place. This comparison testing was based around comparing the usage of the current version of Hlsyn to the updated version of Hlsyn that was worked on. Ideally, the testing was to consist of people who have used Hlsyn before. However, this was not possible, and the test group consisted of students performing exercises from the Hlsyn help documentation on both the current version of Hlsyn and the updated version of Hlsyn. In week 8, the feedback obtained from the comparison based testing was to be implemented, and all features that were implemented were to undergo more testing. During week 9, the focus was to be on minor testing and adjustments. These are any last minute bugs and glitches that may pop up and need to be ironed out. Finally, in week 10, the goal was to be packaging the updated version of Hlsyn into an .exe installer for windows and a .dmg archive for Mac OS. This will allow users to easily install and use the updated version of Hlsyn.

Actual Project Timeline

Week	Milestone
1	Re-design Main Window in Qt using C++
2	Implement revised table method Obtain test feedback
3	Research graphing libraries Continue examining backend code
4	Prototype Graphic Input Method Test prototype and obtained feedback
5	Implement Graphic Input Method
6	Continue Implementing Graphic Input Method
7	Determine how to connect backend and frontend code
8	Create new installer
9	Create cross platform support capability
10	Final testing and tweaking

Figure 9: Actual Project Feature Implementation Timeline

Figure 9 details the actual project schedule that was followed. This was mainly a result of revised design decisions, and issues in the implementation phase that were encountered. Full details and explanations of the project timeline can be found in the subsequent sections of the report.

Final Design and Implementation

The final design for the updated version of Hlsyn differed from the proposed preliminary design in a few ways. The major difference between the proposed design and the final design was a change in the windowing toolkit for the user interface. Originally, the user interface was going to be implemented in PyQt and Python. However, the final design settled on using the regular Qt framework as the windowing toolkit, which would be implemented in C++ instead. While I had very little C++ experience going into the project, it was selected for the final design, as most of the backend code that control the speech synthesizer engine of Hlsyn is written in either C or C++. If Python had been selected a wrapper layer would have had to be implemented between the C/C++ code to connect the backend and the frontend Python code. This wrapper layer would have also added an unnecessary level of complexity that would have hindered debugging later on. As a result of using Qt, the proposed main window design for the updated version of Hlsyn was implemented using the Qt Creator IDE's form builder. The final design that was created using form builder can be seen below in Figure 10:

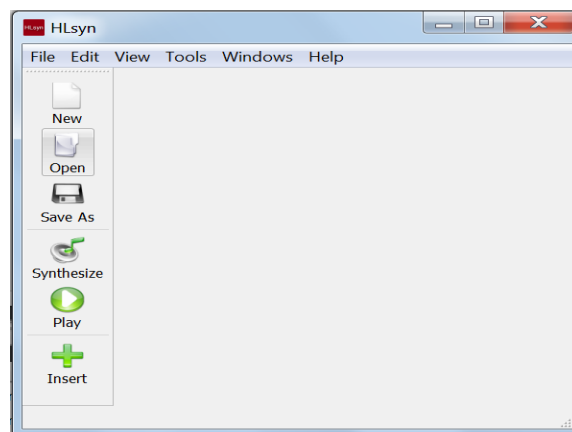


Figure 10: Final Main Window

While the final design is very similar to the proposed initial design, the final design does not show a table for inputting speech parameters in the main window as initially seen in Figure 2. Additionally, in Figure 8, it can be seen that the toolbar has been moved over to the left hand side of the main window, as opposed to the top of the main window in Figure 2. This is just to demonstrate that in the final version the user can indeed move the toolbar around to better fit their workflow. Also keyboard shortcuts have been added to the most common menu functionality as seen in Figure 11:

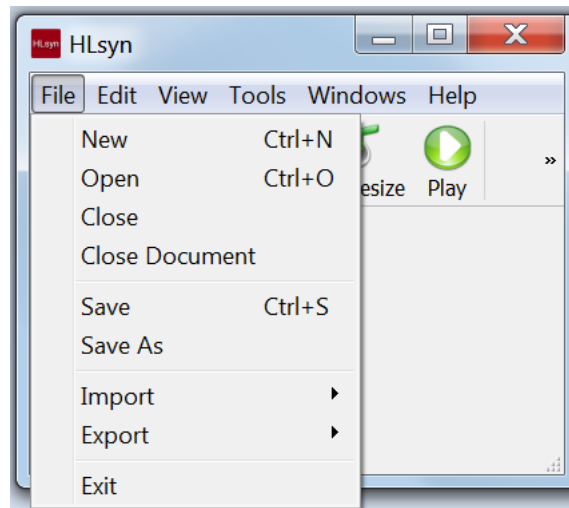


Figure 11: Main Window Menu Shortcuts

These shortcuts will allow the user to be more productive in their workflow by accessing commonly used commands such as New (to create a new document), Open (to open an existing document), and Save (to save the current document).

The next feature that differed slightly from the proposed design was the table-based input system. The final design of the table-based input system can be seen on the next page in Figure 12.

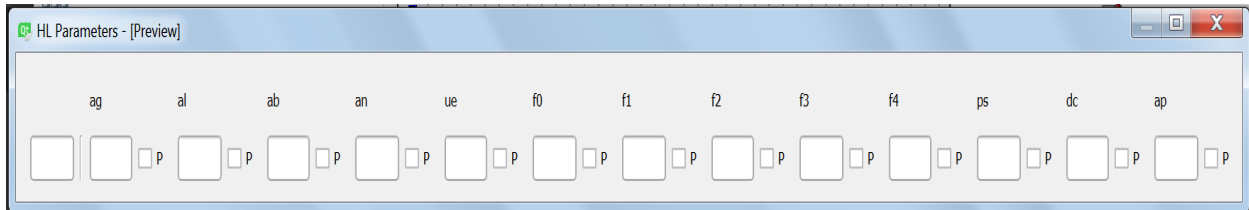


Figure 12: Revised Table Input Method

In Figure 12 we can see that the design differs from the design proposed in Figure 7 in a few subtle but important ways. The first difference is the number of rows that are present. In the proposed design the user is automatically presented with at least 2 rows. In the final version the user is presented with only a single row in the table and can then add more rows easily as needed. By having the minimal number of rows needed in the table, it allows the user to easily keep track of all the data they have entered (versus having blank rows in the table which can be visually distracting), and the window for the table takes up less space in a user's workspace. The final design maintained the idea that users would enter parameter values into boxes, time values in the left most box, and select pivot points by checking the boxes with a P next to them.

One of the other desired features that was not an initial customer requirement but is important was creating a new installer for HLSyn. The current version of HLSyn did come with an .exe based application setup installer. However as can be seen in Figure 13, it would not run:

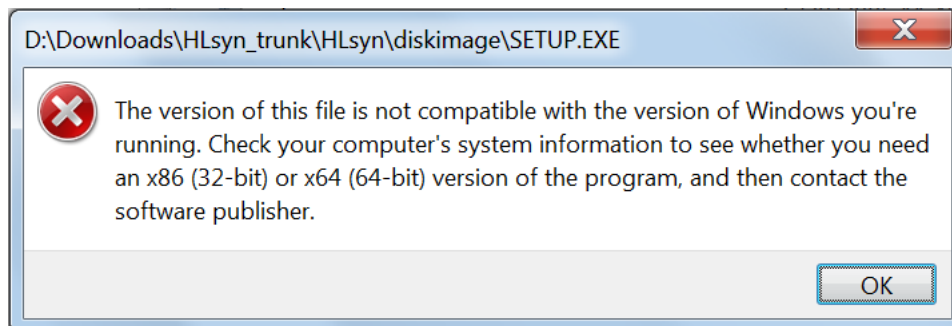


Figure 13: HLSyn Installer Error

As can be seen, the error message states that the version of Hlsyn is not compatible with the version of Windows I am running, which in the case of my laptop that I used for testing is Windows 7 64-bit. This error makes sense, considering that the most recent version of Hlsyn was intended to run on Microsoft Windows 2000. To solve this problem, a new installer was created using Inno Setup^{xv} which is a piece of freeware developed by jrsoftware to allow users to create .exe files that can install an application. The installer was created using Inno Setup's built in scripting language Inno Setup Script (ISS). A snippet of the code used appears in Figure 14 and Figure 15 below:

```
#define MyAppName "Hlsyn"  
#define MyAppVersion "1.0"  
#define MyAppPublisher "Rodberg and Hanson"  
#define MyAppURL "https://muse.union.edu/2017capstone-rodbergs/"  
#define MyAppExeName "hlsyn.exe"
```

Figure 14: Inno Define Application Properties

```
[Files]  
Source: "D:\Downloads\Hlsyn_trunk\Hlsyn\Dist\hlsyn.exe"; DestDir: "{app}"; Flags: ignoreversion  
Source: "D:\Downloads\Hlsyn_trunk\Hlsyn\Dist\aclgut16.dll"; DestDir: "{app}"; Flags: ignoreversion  
Source: "D:\Downloads\Hlsyn_trunk\Hlsyn\Dist\aclgut32.dll"; DestDir: "{app}"; Flags: ignoreversion  
Source: "D:\Downloads\Hlsyn_trunk\Hlsyn\Dist\allegro.ini"; DestDir: "{app}"; Flags: ignoreversion  
Source: "D:\Downloads\Hlsyn_trunk\Hlsyn\Dist\default.hld"; DestDir: "{app}"; Flags: ignoreversion  
Source: "D:\Downloads\Hlsyn_trunk\Hlsyn\Dist\default.kld"; DestDir: "{app}"; Flags: ignoreversion  
Source: "D:\Downloads\Hlsyn_trunk\Hlsyn\Dist\hlfileio.dll"; DestDir: "{app}"; Flags: ignoreversion  
Source: "D:\Downloads\Hlsyn_trunk\Hlsyn\Dist\hlsyn.bmp"; DestDir: "{app}"; Flags: ignoreversion  
Source: "D:\Downloads\Hlsyn_trunk\Hlsyn\Dist\hlsyn.img"; DestDir: "{app}"; Flags: ignoreversion  
Source: "D:\Downloads\Hlsyn_trunk\Hlsyn\Dist\hlsyn.ini"; DestDir: "{app}"; Flags: ignoreversion  
Source: "D:\Downloads\Hlsyn_trunk\Hlsyn\Dist\hlsyn32.dll"; DestDir: "{app}"; Flags: ignoreversion  
Source: "D:\Downloads\Hlsyn_trunk\Hlsyn\Dist\lisp.exe"; DestDir: "{app}"; Flags: ignoreversion
```

Figure 15: Inno Installer Included Files

As can be seen in Figure 14, the first step in creating the installer, was to define some basic information that will show up when the user installs Hlsyn. I made sure to include a link to my capstone project website so users would know where to go to for documentation. The second part of the installer was to include all of the files that Hlsyn needs to function, as seen in Figure

15. Upon installation, the installer will unpack the files listed to a directory specified by the user. A screen shot of the final installer can be seen below in Figure 16:

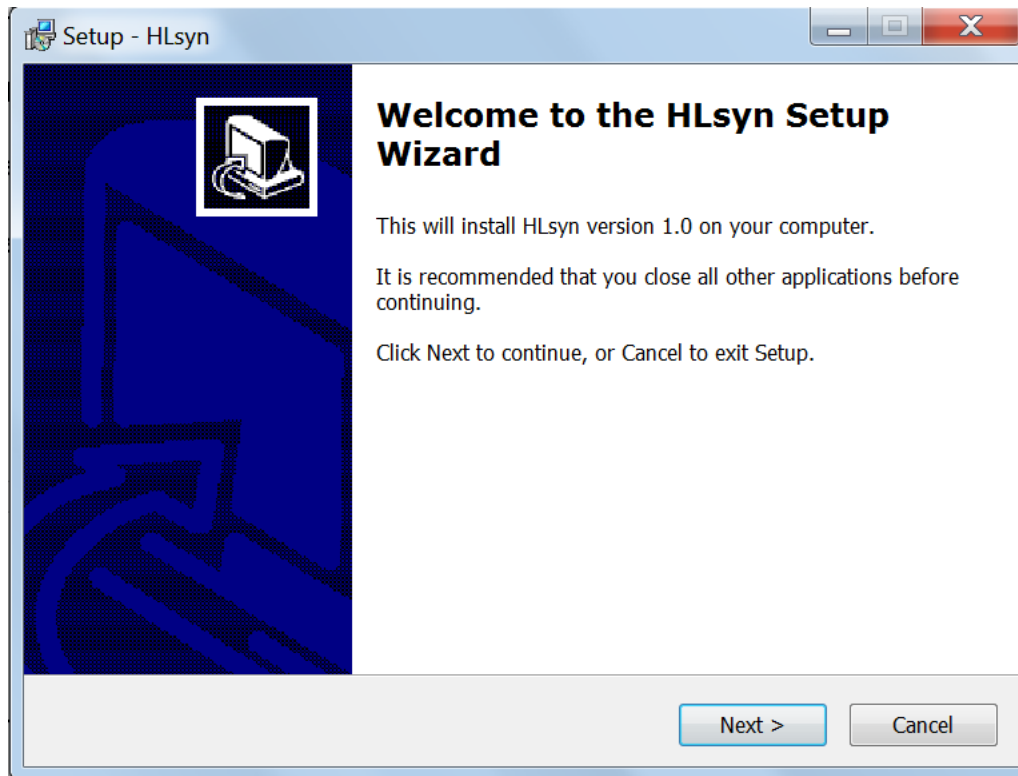


Figure 16: Hlsyn Installer

Once the installer has completed the installation, the user will be presented with the option of immediately running Hlsyn.

The next requirement was cross-platform support. This was a very important feature to the main customer, as she mainly works on Mac OS. The initial design proposal called for the new updated GUI to be connected to the backend of the current version of Hlsyn.

Unfortunately, this could not be accomplished due to difficulties recompiling the backend code as a result of an outdated build system (please see the Production Schedule Section and Conclusion section for more information). However, cross-platform support for Mac OS was still able to be achieved by utilizing WINE and Wine Bottler. What WINE does is act as a

compatibility layer for Mac OS. Instead of using a traditional virtual machine approach or emulation, it translates the Windows API calls into POSIX (Portable Operating System Interface) compatible instructions (Unix which Mac OS is based on, uses POSIX as its API standard). This allows the user to run HLsyn on Mac OS by first downloading a .zip file containing all of the files for HLsyn. Then when they click on the .exe file, WINE will run the .exe file for the user. When running HLsyn in WINE the program has the exact same functionality as a user would expect when using Windows:

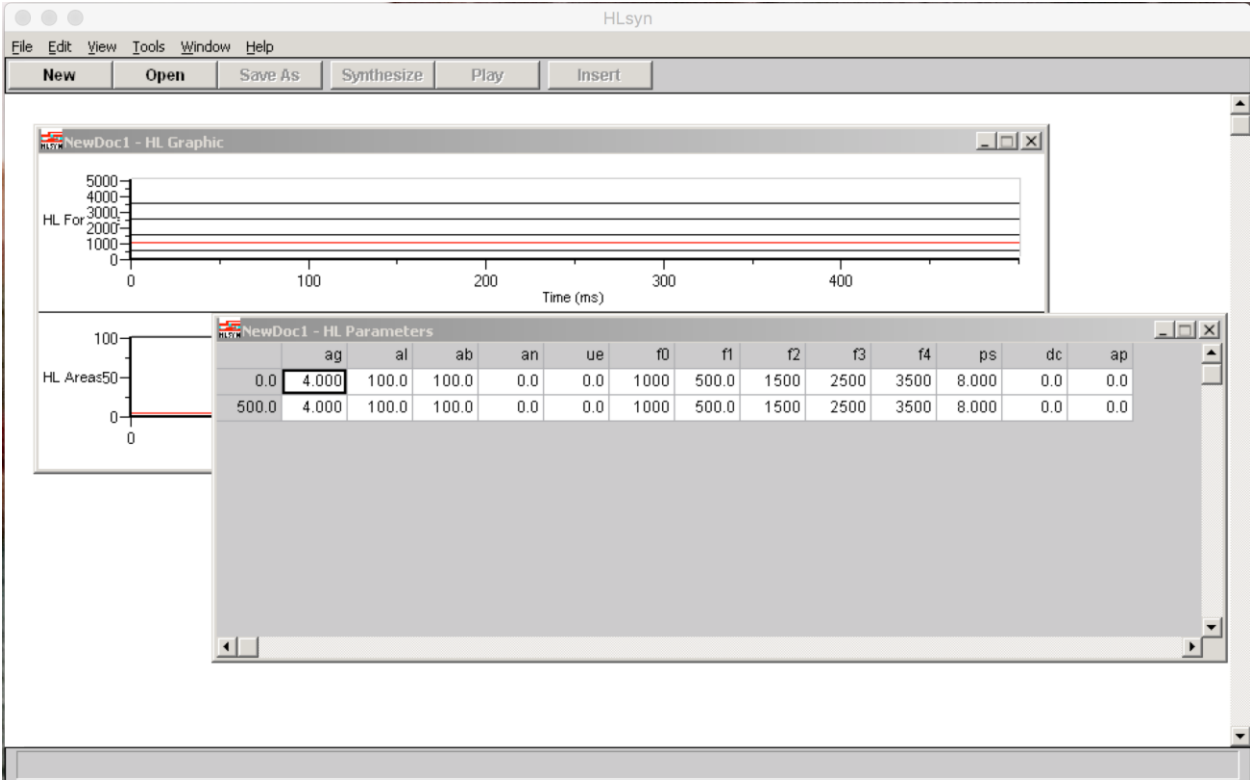


Figure 17: HLsyn Running On Mac OS Using WINE

As seen in Figure 17 above, HLsyn appears exactly the same as when run on Microsoft Windows. All user controls and application behavior are also the same as when run on Microsoft Windows. What Wine Bottler does is create an .app file which can run natively on Mac OS containing all the files needed to run HLsyn, as well as all of the files needed to run

WINE without needing to install WINE separately. Wine Bottler has many different configuration options that had to be explored:

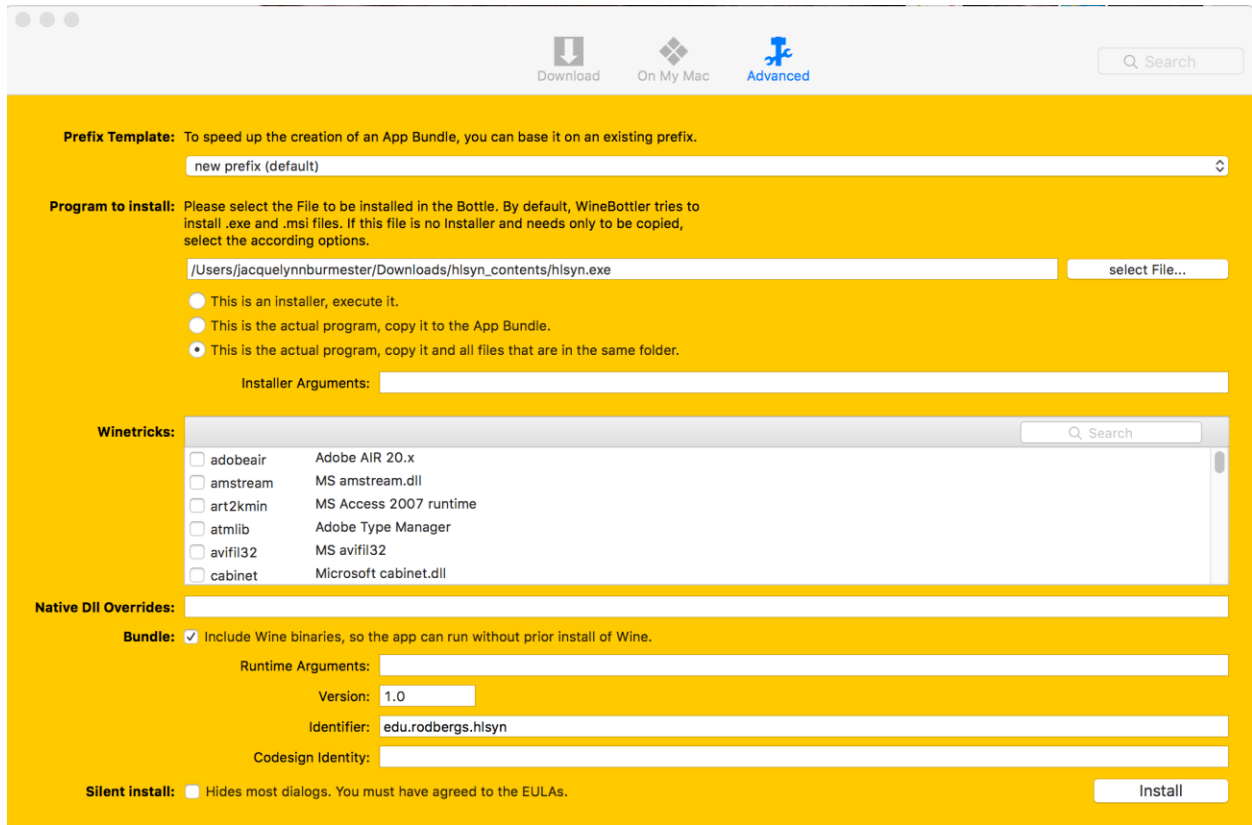


Figure 18: Wine Bottler Configuration Options

Though Wine Bottler has many more advanced configuration options, the main focus was on choosing how the application would be installed and executed. The first option was to have an .app file that would run the Hlsyn installer, allowing the user to install the application and then run it as seen in Figure 20 on the next page.

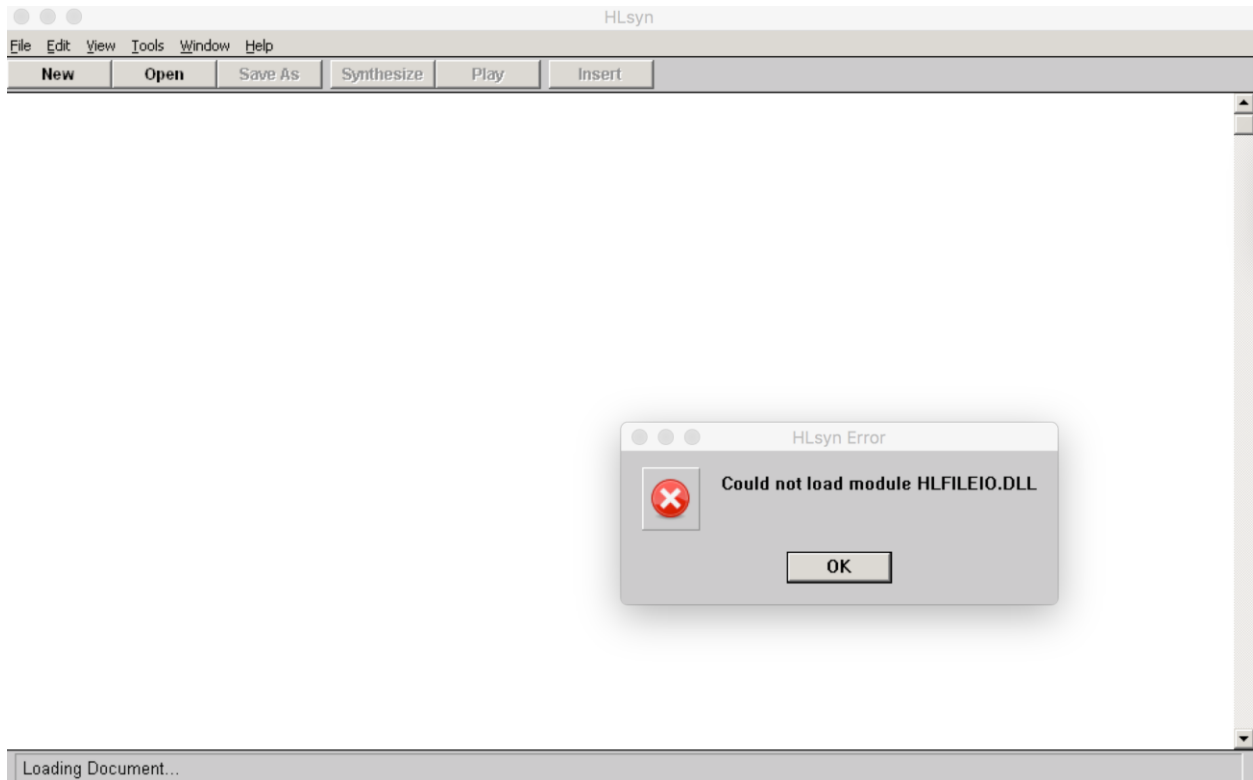


Figure 19: Wine Bottler Installer Error

The problem with creating an .app file of the installer that the user could execute is that the installer was not copying over all of the required files needed to run Hlsyn. This resulted in the error seen in Figure 19, referring to the fact that Hlsyn couldn't find the DLL file that is critical for a user, not only to be able to input speech parameters but also to synthesize sounds from the inputted speech parameters. The second option was to copy the main application, hlsyn.exe, to the Mac OS application folder. The problem that resulted from this option was that the Wine Bottler repeatedly failed during compilation to create an .app file. The third option was to copy hlsyn.exe and all its dependent files to a folder; then the user could navigate to the folder selected and directly run Hlsyn. While an .app file was successfully compiled by Wine Bottler, the .app file would launch, then quickly force close. Console messages did not prove successful in resolving this issue. It was therefore determined that cross-platform

support for Mac OS would be implemented by providing the user with a .zip file containing hlysn.exe and all the files needed to run it. The user would be directed when they download the .zip file to download and install WINE in order to then run HLsyn.

The final feature of the project was implementing the graphic-based parameter input method. Many of the concepts and theories behind the proposed preliminary design were used, such as a user being able to draw points, and using vector-based images to represent graphs drawn by the user. However, the ability for the user to interact with single points of data, and shape the speech curves in a manner similar to Figure 4, had to be changed. To create the graphs a user would draw a custom graphing library called QCustomPlot was used, which was selected due to its ease of use and relatively well documented feature set. The design that was implemented to allow the user to draw graphs can be seen below in Figure 20:

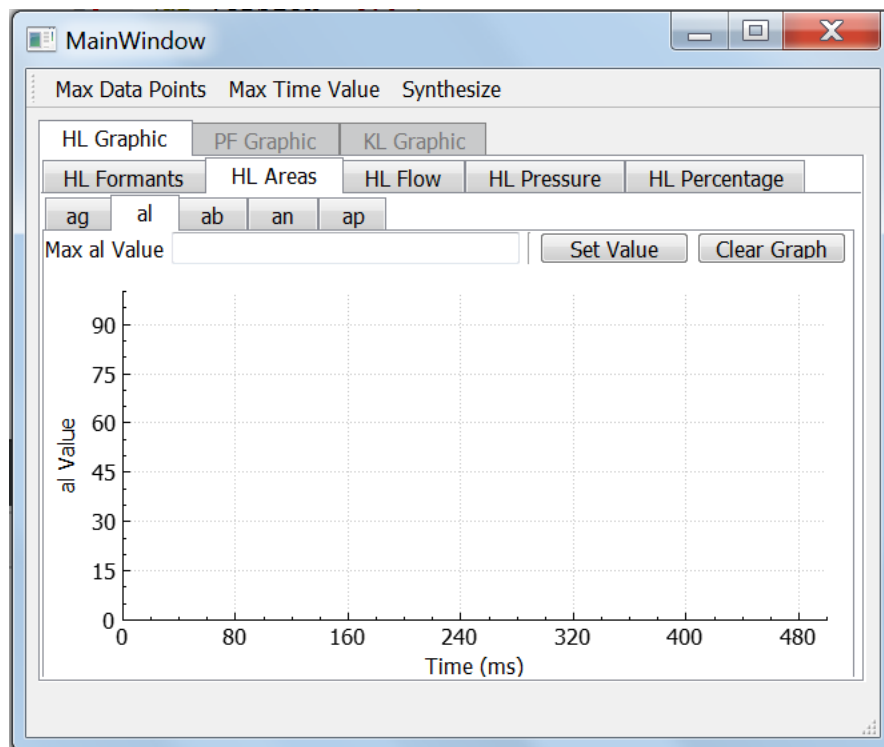


Figure 20: Graphic Input Window Example

The user clicks the tab of the HL Graphic speech parameter category. Then the user would click the tab of the specific parameter they would like to manipulate. Figure 22 below is a code snippet of how the graphs are setup for the user to draw points on:

```
void MainWindow::makePlots() {
    // Add HL Formants graphs
    // f0 Graph
    ui->f0_graph->addGraph(); // Add the graph to the screen
    // Set the data of the graph to be the vectors containing the points
    // a user clicks
    ui->f0_graph->graph(0)->setData(f0dataX, f0dataY);
    // Set the range of the xAxis to be from 0 to the maximum time value specified
    // by the user defaults to 500
    ui->f0_graph->xAxis->setRange(0, maxTimeValue);
    // Set the range of the yAxis to be from 0 to the maximum f0 start value
    // which is set as a constant 500
    ui->f0_graph->yAxis->setRange(0, F0_START_MAX);
    // Set the label of the xAxis to the string "Time (ms)"
    ui->f0_graph->xAxis->setLabel(TIME_LABEL);
    // Set the label of the yAxis
    ui->f0_graph->yAxis->setLabel("f0 Value");
    // Sets the color of the points that will be plotted by the user
    ui->f0_graph->graph(0)->setPen(QPen(QBrush(Qt::red), 2));
    // Redraw the graph to reflect the above changes
    ui->f0_graph->replot();
    qDebug() << "f0 graph has been plotted";
}
```

Figure 21: Setup Plot Graph Code Snippet

The same procedure is followed for each speech parameter graph. The graph is added to the main window, then the graphs x-axis and y-axis data are set to two vectors that will be filled up with data as the user clicks points on the graph. The first two points on the graph are determined by connecting the first point clicked on the graph to the point specified at time value 0 as seen in Figure 22. To get the parameter value at time value 0, the user is prompted through an input box as seen below in Figure 22:

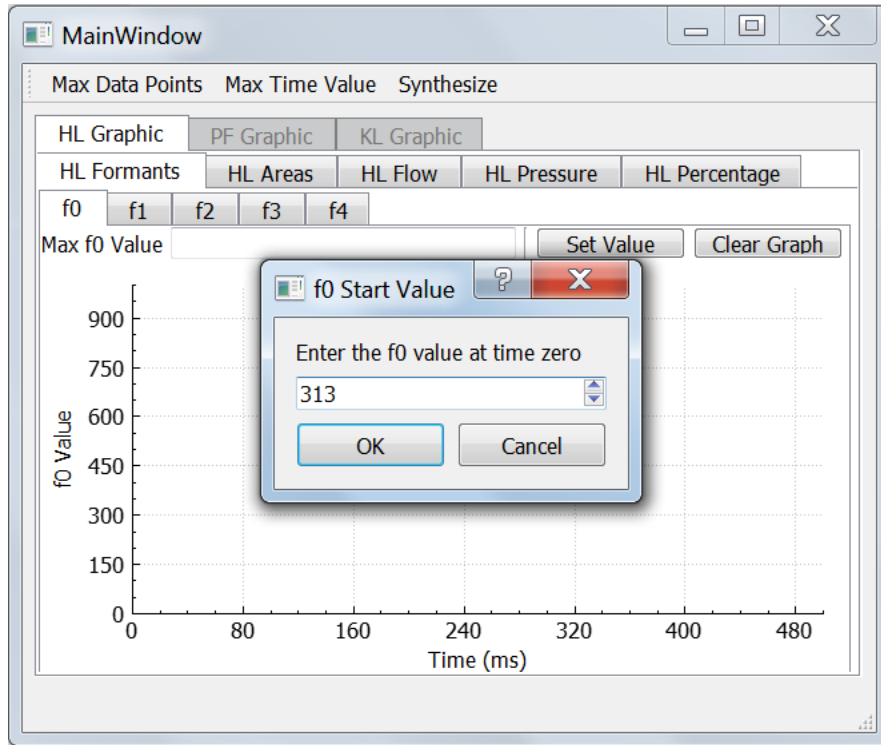


Figure 22: Starting Value Prompt

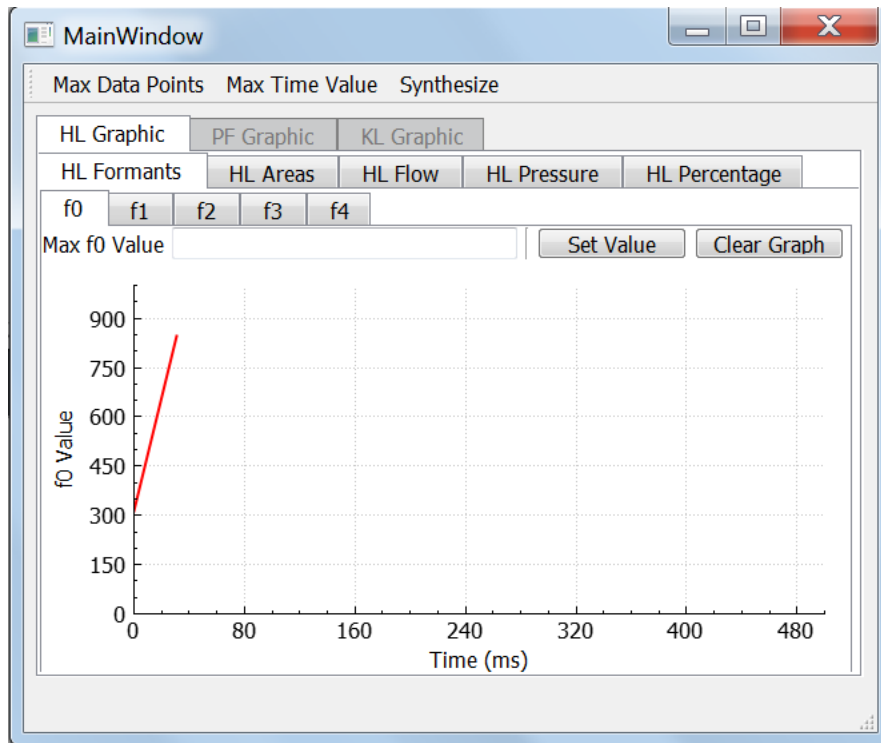


Figure 23: Connecting First Two Points

As the user plots points on the graph, the data are added into two vectors that represent the x coordinates (time), and y-coordinates (parameter value) of the point that was drawn. Each speech parameter has two of these vectors that represent the time at which the speech parameter is being manipulated and the value of the speech parameter.

When the user clicks on a point on the graph to draw in a data point, this is handled by a click event handler. Once a user has plotted all of the data points they want to plot, a log file will be generated showing the data points that were plotted and the time at which they were plotted:

```
f0 plot clicked
The "0" is "0" , "700"
The "1" is "80.1158" , "743.902"
The "2" is "108.108" , "670.732"
The "3" is "149.614" , "756.098"
The "4" is "221.042" , "626.016"
The "5" is "271.236" , "735.772"
The "6" is "305.019" , "467.48"
The "7" is "373.552" , "813.008"
The "8" is "479.73" , "825.203"
The "9" is "500" , "900"
f0 replotted
```

Figure 24: Log Output Of Plotted Points

From the outputted log of points plotted, a user can simply enter these points into the current version of Hlsyn table, from which they can synthesize and play the sound that would be generated. However, since the GUI is not connected to the backend yet, the user has to enter the output points manually into the table in Hlsyn in order to synthesize the sound. Users also have the option of erasing a graph's data by clicking on the "Clear" button. This will clear the

two vectors representing the time and speech parameter values of the graph that was being drawn. The user can then draw in new points on the graph, which will be stored in the two vectors.

Performance Estimates and Results

In this section of the report the performance estimates of the preliminary design will be compared to the results of the final implemented design. The testing methods for each of the features implemented will also be discussed.

Due to the nature of this project which centered around the user experience, many of the performance estimates were based on qualitative criteria instead of quantitative measurements that had to be precisely achieved. One of the first areas of performance that had to be addressed was updating the visual appearance and increasing the ease of use of the main GUI. For this, the method of testing was very qualitative in nature with the criteria being clear visuals, clearer indication of what all the buttons and functions of the program did, and how intuitive the program was for a user to just pick up and use. To accomplish this, I recruited a group of eight of my friends to provide me with their thoughts and impressions on the above mentioned criteria. The group of eight was divided evenly into two groups with group A testing the current version of Hlsyn and group B testing the updated version of Hlsyn. To ensure accurate results, I did not tell the test subjects which was the current version and which was the updated version. The feedback from the tests showed that 3 out of the 4 members of group A felt that the current version's user interface was not satisfactory. The comments from group A range from the current interface was "ugly" and had "harsh intimidating visuals", or the test subject "felt lost by what the buttons did". For group B on the other hand, all four test subjects felt that the updated version had satisfactory visuals. Comments for group B ranged from "Buttons make sense" to "movable toolbar is helpful". These tests, coupled with the comparison to design guidelines laid out by Microsoft and Apple, show that the ease of use and

intuitive nature of the main GUI had been improved. The only negative result was the inability to conduct testing of the updated GUI with the current version of HLsyn backend code to fully test typical user interactions and workflow.

For the revised table input method, the proposed design called for the ease of use to improve over the current table method. Visually, the updated table method was an improvement, and several test subjects noted they preferred the ability to select pivot points using check boxes instead of double clicking on a table's cell, as well as the ability to edit time data after a row has been created. Limited testing and results were able to be achieved beyond this for the revised table input once again due to the inability to connect the updated GUI of HLsyn to the current backend of HLsyn. More details will be provided on the difficulties of connecting the updated Frontend code to the current version of HLsyn backend code later in this section.

The third feature that needed testing was a new installer for HLsyn. For this, the performance criteria compared to the proposed design was that it needed to allow the user to easily install the application. Considering that the current version of HLsyn installer is broken, it was relatively straightforward to achieve the desired performance goal. To ensure the installer was fully working, all functions of the installer were successfully tested including installing HLsyn, changing the directory the user can install the program into, and uninstalling the program if a user wishes to remove the program from their computer.

The fourth feature was cross-platform support. The original performance goal for this was to have a version of HLsyn that could run natively as an .app file on the latest version of Mac OS. For my testing platform, I used a friend's computer running Mac OS Sierra. Since it

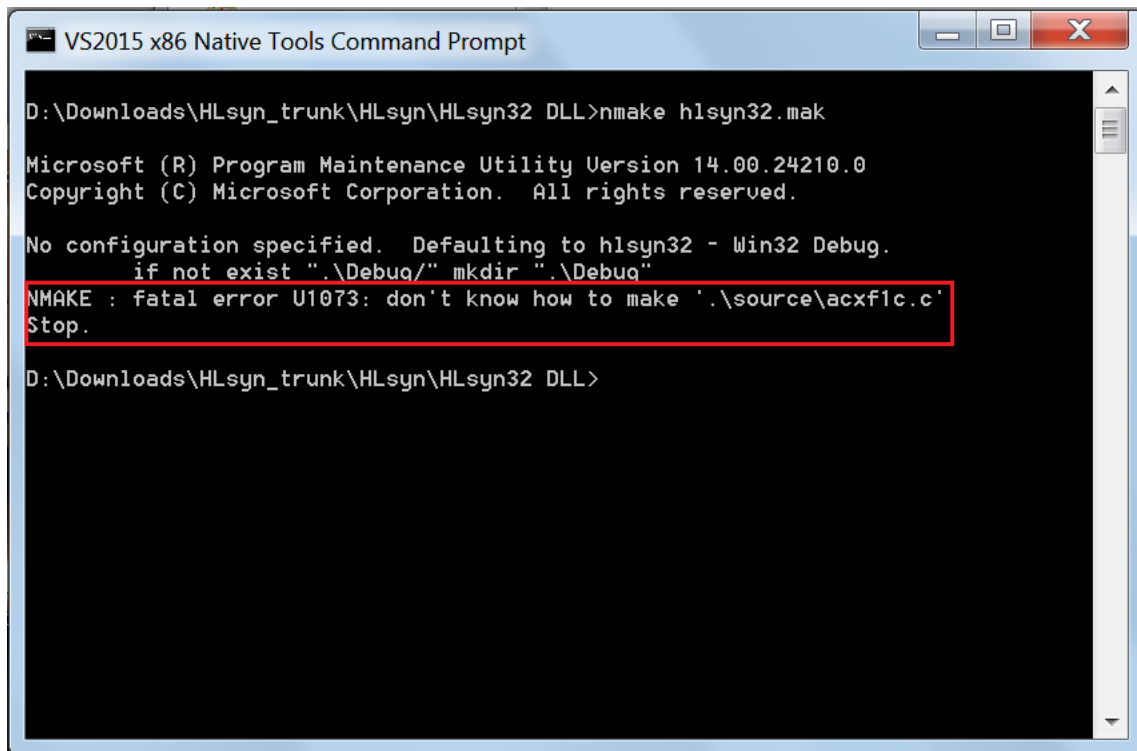
was impossible to create a native application for Mac OS due to issues connecting the current backend of Hlsyn to the updated GUI, the performance criteria of the proposed design had to be met in a different way. Running hlsyn.exe was only successful when it was run using the WINE compatibility layer. While Wine Bottler was able to generate a native .app file, these .app files either failed to execute, or when they did execute, generated errors denoting missing file dependencies needed to run Hlsyn .exe.

The last feature was implementing a graphic-based input method. The performance criteria for this was that the user needed to be able to synthesize and then play sounds generated from data plotted on graphs. Additional performance criteria were that users should be able to interact with a single point on the graph to make changes to the data or delete any point without having to start over. During testing, drawing data points on the graph proved successful; however, there were some glitches and issues. One of these issues was that due to limitations in the QCustomPlot library that was used to generate the graphs, there is no way for a user to click on a single point on the graph and adjust its value after it has initially been plotted. Instead, the user must completely clear the graph they are plotting points on if a mistake in plotting is made. The same can be said about deleting single points. Once again, a user must press the “Clear” button on the graph they are working on, which will delete all points on the currently selected graph. There were additional issues that arose during the testing of the graph. One of these issues was that there would be occasional glitches, where if a user plots a point on the graph that is close to the maximum time value or maximum speech parameter value that the user specified when they first setup the graph (please see the user manual for complete step by step directions on how setup drawing graphs). These glitches

appear as line segments on the graph that have a very steep slope. An additional glitch that happens occasionally is when generating a console log of the points that were plotted. What will occur, is that points plotted when time is equal to 0 and the point that is plotted as the time where the second to last point on the graph is plotted, will seem to swap vector value. Overall, however, the testing showed that the performance criteria for the graphic input method was mostly met.

One of the biggest challenges of this project was connecting the backend of the current version of Hlsyn to the updated GUI. This was one of the performance criteria that was not able to be met. The main reason for this was that normally, I would be able to import the libraries that control the functionality of the current version of Hlsyn as a series of .dll files into the Qt Creator IDE. The problem, however, was that the .dll files that were provided in the Hlsyn source code did not include the additional needed .lib files. Without the .lib file there is no way to resolve any file dependencies the .dll file may have. Normally, in software development, while this would be a headache, it would not mean this issue could not be resolved. Usually what could be done is to take the provided C and C++ Hlsyn source code files and recompile them from scratch to generate not only updated .dll library files, but also the necessary .lib file. However, the difficulty with this method is that since Hlsyn is so outdated, so are the build tools used to compile the code from scratch. Hlsyn uses Microsoft's nmake build system to compile the code. To build the current version of Hlsyn nmake version 2.00. The problem with this is that nmake version 2.00 has long been discontinued by Microsoft and is no longer available to download. The current version of nmake is version 14.00, which, when attempting to build the files needed to compile Hlsyn, generates the following error as seen in

Figure 25:



```
VS2015 x86 Native Tools Command Prompt
D:\Downloads\HLsyn_trunk\HLsyn\HLsyn32 DLL>nmake hlsyn32.mak
Microsoft (R) Program Maintenance Utility Version 14.00.24210.0
Copyright (C) Microsoft Corporation. All rights reserved.

No configuration specified. Defaulting to hlsyn32 - Win32 Debug.
if not exist ".\Debuq/" mkdir ".\Debuq"
NMAKE : fatal error U1073: don't know how to make '.\source\acxf1c.c'
Stop.

D:\Downloads\HLsyn_trunk\HLsyn\HLsyn32 DLL>
```

Figure 25: Nmake Build Error

This build error refers to two critical errors. The first is that the build system doesn't recognize the instructions of the make file that is being used. The second issue was that it was looking for a path that was hardcoded to the build system used by the Sensimetrics Corporation when they were compiling the code. While it may seem trivial to recreate the build system, it is not. Even if nmake knew how to interpret the make files instructions, it would be necessary to go through hundreds of lines of code to recreate the Sensimetrics Corporation's build system file structure.

As a whole, the performance criteria for the project laid out in the initial design was mostly met. While the occasional limitations and glitches of the graphic based input method coupled with the issues recompiling the code were frustrating, it did not stop me from implementing a working prototype of the desired system.

Cost Analysis

It is difficult to gauge what the cost of this project was, because no money was actually spent during the course of the project. The source code was provided to me for free by my main customer, Professor Hanson, which was provided to her from the Sensimetrics Corporation. As I am a student and the sole person working on this project for capstone, there was no associated labor costs in developing the software. The real issue with cost is that, while it is no longer being sold, HLsyn is a commercial application that required users to obtain licenses. In order to publicly release our changes we would need the permission of the Sensimetrics Corporation. The goal would be to release the changes to the software and bundle the original software free of charge under the General Public License (GPL). However, as HLsyn is still the intellectual property of the Sensimetrics Corporation, it is their prerogative to determine the cost of using the updated application.

User Manual

The following section describes how to setup Hlsyn for both Windows and MAC OS, as well as a tutorial on how to use the graphic input method.

Installing Hlsyn on Windows

1. Download the Hlsyn installer from my capstone website located at:
<https://muse.union.edu/2017capstone-rodbergs/>
2. Using Windows Explorer navigate to the folder to where you downloaded the setup file which should be named: "hlsyn_setup.exe"
3. Run "hlsyn_setup.exe" and follow the on screen installation instructions.
4. Once the installation has completed you will be prompted to run Hlsyn.

Installing Hlsyn on Mac OS

1. Download the Hlsyn .zip package from my capstone website located at:
<https://muse.union.edu/2017capstone-rodbergs/>
2. Download WINE from the following website:
<https://www.winehq.org/>
3. Follow the setup instructions provided with WINE to install it.
4. Using Finder, navigate to the folder where you downloaded the .zip package.
5. Extract the contents of the .zip package
6. Using Finder, navigate to the directory that contains the unzipped contents of the .zip package.
7. Double click on "hlsyn.exe" and you will be prompted by WINE to run the application.
8. Allow Wine to run the application and Hlsyn will now start.

Using the Graphic Input

1. Start the Hlsyn companion app for drawing graphics.
2. Once the application has started, select the tab of the category of speech parameter you would like to manipulate, for Example, HL Graphic.
3. Next select the tab of the type of speech parameter you would like to manipulate, for example HL formants.
4. Then select the tab of the specific speech parameter you would like to manipulate for example f0.
5. Click on the “Max Data Points” button on the toolbar and enter in the number of data points you plan to plot for each graph, for example 10.
6. Click on the “Max Time Value” button on the toolbar and enter in the latest time value you would like to manipulate speech parameters at, for example 500.
7. To start plotting data click on any point on the graph, you will then be prompted to entering the starting speech parameter at time value 0.
8. Next a line segment will appear between the first point you plotted on the graph and the starting value you entered.
9. From here plot all of the points of data you wish to plot up to the maximum number of data points you specified in step 5.
10. When you are plotting your second to last data point you will promoted to enter the speech parameter value at the maximum time value you specified in step 6.

11. After you have plotted all points, a log output will be generated showing the points that you plotted on the graph. These points then can be manually entered into the HLsyn table to synthesize the sound and then play it back.

Discussion, Conclusions, and Recommendations

The prototype that was developed did mostly meet the design requirements and criteria laid out by the main customer, Professor Hanson. Hlsyn is an old application that, while its speech synthesis engine is still very powerful and applicable to academia and research, its outdated GUI held it back for use in modern times. To remedy this issue, an updated user interface was designed that featured modern visuals and increased the ease of use.

Features already included in Hlsyn were updated and revised to further increase the user's flexibility and workflow productivity when using the software. New methods of data input were developed that allowed users an alternative to entering values into a table by instead plotting points on a graph to represent speech parameter values and their changes over time. Hlsyn gained cross-platform support to allow not only the customer, but other users, to use Hlsyn on the platform that they not only have access to but are most familiar with. It also gained an easy to use installer that makes it extremely simple for the user to setup Hlsyn and start using it immediately.

Despite the occasional glitch and issue with the table-based input method, and issues with recompiling the Hlsyn source code, the prototype turned out reasonably successfully. However, there are several recommendations for future work that can be made. One of the areas of future work is solving the issues with recompiling the source code of Hlsyn to allow the current backend to be connected to the updated GUI. One possible method for solving this is to use the latest version of Microsoft Visual Studio to import the source code and generate a new make file. A second area of future work is to have a table that is generated for the user after they have drawn all of their data points

graph. Additionally, the log file that is outputted to the console is only available with the Qt Creator IDE and it would be useful to have a user be able to open a console window in the final release version. The final area of future work is that once the source code has been successfully compiled for Hlsyn it will be possible to use the Qt framework to create an .app file that can run natively on Mac OS without the dependencies of needing to install WINE first.

This project was a great learning experience for me as it gave me an avenue to learn both C++ and user interface development skills that I have wanted to learn for a while. It also increased my interest in speech synthesis, and gave me a greater appreciation for the complexities of human speech. Being able to complete a capstone design project has been a great experience.

Acknowledgments

Professor Helen Hanson: Thank you for allowing me to work with you on the project and increasing my interest in speech synthesis. While our schedules for this term were both demanding and tiring you always were willing to meet with me and discuss the project which I greatly appreciate.

Professor Matt Anderson: Thank you for the moral support as well as the help with understanding the build system used to compile the Hlsyn source code. Also thanks for putting up with my workflow style and putting aside your distaste for Windows to help me.

Gene Davison: Thank you for taking the time to print my capstone poster.

My Girlfriend, Family and Friends: Thank you Meaghan for putting up with me taking over the desk in your room to work until all hours of the night, and for listening to my frustration with the problems that arose during the project. Thank you to my friends, for the emotional support they provided me and the many times where I was too busy to hang out with them or I looked like a zombie from sleep deprivation. And mostly importantly thank you to my incredible parents and family for the constant support and reassurance at all hours of the night and day that I was headed in the right direction and could complete the project. An especially big thanks to my father for inspiring me to love technology with the passion that I have today, as well as to think outside the box. And thank you to my Opa for providing inspiration, and teaching me the valuable lesson of never giving up no matter how hard a problem may seem.

References

- ⁱ "History and Development of Speech Synthesis." History and Development of Speech Synthesis. N.p., n.d. Web. 26 Nov. 2016. <http://research.spa.aalto.fi/publications/theses/lemmetty_mst/chap2.html>.
- ⁱⁱ "History of Computers and Computing, Automata, Wolfgang Von Kempelen." History of Computers and Computing, Automata, Wolfgang Von Kempelen. N.p., n.d. Web. 26 Nov. 2016. <<http://history-computer.com/Dreamers/Kempelen.html>>.
- ⁱⁱⁱ "Von Kempelen's Talking Machine." Von Kempelen's Talking Machine. N.p., n.d. Web. 26 Nov. 2016. <<http://www.haskins.yale.edu/featured/heads/SIMULACRA/kempelen.html>>.
- ^{iv} "Before Inventing The Telephone, Alexander Graham Bell Tried To Teach His Dog To Talk." The Dodo - For Animal People. N.p., 30 Mar. 2014. Web. 11 Mar. 2017. <<https://www.thedodo.com/before-inventing-the-telephone-489117573.html>>.
- ^v "Stewart's Electrical Analog." Stewart's Electrical Analog. N.p., n.d. Web. 26 Nov. 2016. <<http://www.haskins.yale.edu/featured/heads/SIMULACRA/stewart.html>>.
- ^{vi} "The 'Voder' & 'Vocoder' Homer Dudley, USA,1940." 120 Years of Electronic Music. N.p., 17 Dec. 2013. Web. 26 Nov. 2016. <<http://120years.net/the-voder-vocoderhomer-dudleyusa1940/>>.
- ^{vii} Grover, Dale. "SC-01A Speech Synthesizer and Related ICs." SC-01A. N.p., n.d. Web. 26 Nov. 2016. <<http://www.redcedar.com/sc01.htm>>.
- ^{viii} "Speech Synthesis." Speech Synthesis. N.p., n.d. Web. 11 Mar. 2017. <https://www.cs.mcgill.ca/~rwest/link-suggestion/wpcd_2008-09_augmented/wp/s/Speech_synthesis.htm>.
- ^{ix} "About." Sensimetrics Corporation. N.p., n.d. Web. 26 Nov. 2016. <<http://www.sens.com/about/#history>>.
- ^x "Project Deliverables." ProSynth Project Deliverables Home Page. N.p., n.d. Web. 26 Nov. 2016. <<http://www.phon.ucl.ac.uk/project/prosynth/>>.
- ^{xi} "Right to Ones Voice?" The Research Center on Computing and Society. N.p., n.d. Web. 26 Nov. 2016. <<http://rccs.southernct.edu/right-to-ones-voice/>>.

^{xii} Edison, Thomas Wilfred, Dr., and R. Ponmurugan, Dr. "A Study on Influence of Software Familiarity Over Software Consumption Among Non-Institutional Consumers in Tamilnadu." *Journal of Business Management & Social Sciences Research* November 1.2 (2012): 60-69. [Http://citeseerx.ist.psu.edu](http://citeseerx.ist.psu.edu). Web. 11 Mar. 2017. <<http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=33758912A6024D10186F62756D3EADB8?doi=10.1.1.403.4771&rep=rep1&type=pdf>>.

^{xiii} @qtproject. "The Future Is Written with Qt: Cross-platform Software Development for Embedded & Desktop." Qt. N.p., n.d. Web. 26 Nov. 2016. <<https://www.qt.io/>>.

^{xiv} "What Is PyQt?" Riverbank | Software | PyQt | What Is PyQt? N.p., n.d. Web. 26 Nov. 2016. <<https://riverbankcomputing.com/software/pyqt/intro>>.

^{xv} "Inno Setup." Inno Setup. N.p., n.d. Web. 12 Mar. 2017. <<http://www.jrsoftware.org/isinfo.php>>.