Susquehanna User Manual - v0.2.1 Cliffside

thetacola

May 25, 2025
Revision 6

Contents

Chapter 1

Introduction

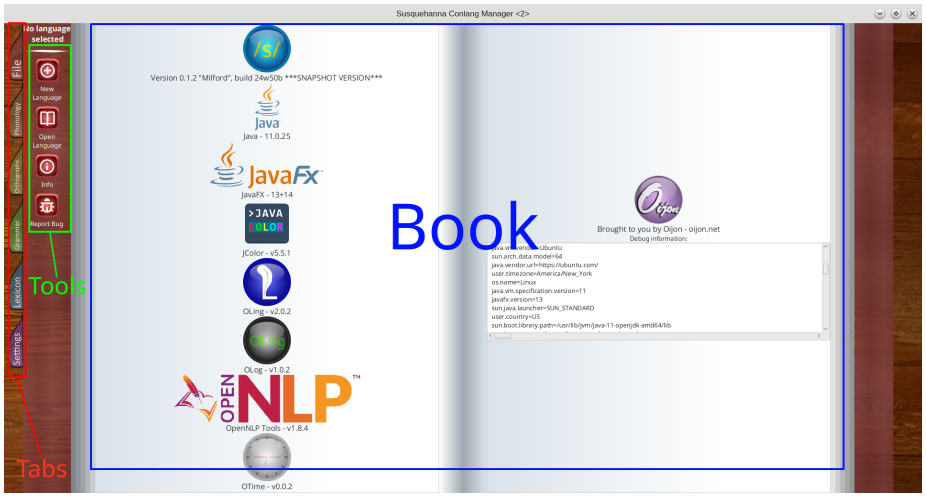

Figure 1.1: The three main components, labeled.

Susquehanna is a program made for the creation and management of conlangs. This application may also find use as a tool for the documentation of natural languages. Note the three main parts of the interface in Figure 1.1, those being the tabs, the tools, and the book. The tabs categorize tools into groups, the tools allow the user to switch between books, and the book allows the user to interact with the language. The following sections will allow you to become familiar with these parts of the interface. As of this version, Susquehanna autosaves all changes as soon as they are made, so one does not have to worry about data loss.

1.1 Tabs

Tabs can be found on the left-most edge of the application whenever it is open. These tabs categorize tools into categories, so that each tool is easier to find. These tabs are also color coded, and the color of the book background is decided by the currently selected tab. To check which tab is being used, the user can see which color book background matches which color tab.

There are six tabs, those being File, Phonology, Orthography, Grammar, Lexicon, and Settings. This manual covers each tab in detail, for more information on each tab please check the table of contents to find the page about a given tab. When using Susquehanna on low-resolution displays or small window sizes, the tabs can be scrolled through using the scroll wheel. Using the PgUp and PgDown keys on the number pad also works.

Figure 1.2: The six tabs in Susquehanna.

For developers: Adding a new tab to Susquehanna is fairly simple. This can be done by editing *net.oijon.susquehanna.gui.Navbox*. First, create an image for your new tab. A template for this can be found in *src/main/resources/img*. Save your image as {*name*}*-tab.png*. Then, add a new BinderTab instance in the Navbox class, next to the rest of the BinderTab instances. Make sure to set the name of the tab to the name you picked earlier for the image, minus the "-tab.png" bit. Otherwise, your image will be unable to link to this tab. Then, edit the line starting with "VBox navVBox" in *Navbox.Navbox()*, and add your tab to the end of the list. To make this tab functional, add {*tab variable name*}*.createTransferAction(*{*book ID*}*)* to *Navbox.createTransferActions()*. More information on book IDs can be found in Section 1.3. Once these steps are done, you should have a fully functional tab.

For developers: There is no limit set for the amount of tabs that can be loaded at a given time by Susquehanna, however performance and general usability will be impacted when adding several hundreds of tabs. Furthermore, as JavaFX parents can "only" contain *Integer.MAX_VALUE* (around 2^{31}) children, there's likely a limit of around 2^{31} tabs. It should be noted that Susquehanna is not built to handle this many tabs, and would likely crash from the amount of images needed to render before this point (each tab image following the template is around 2.81kB, $2.81 * 2^{31} \approx 6,034,429,051kB \approx$ 6TB, and the JVM is not happy handling anything more than 4TiB of memory at a given time). Scrolling through this amount of tabs trying to find the correct one would also likely not be a pleasant experience for the user, so tabs should be kept to a rather low amount if possible.

1.2 Tools

Figure 1.3: An example of a tool button. This tool button is the Info button in the File tab.

Tools are various pages that allow for the editing of a language. These take the form of buttons to the right of tabs. The container that all of these buttons are in is called the toolbox. Each tab has a different toolbox corresponding to its category. For example, the phonology tab has the "View Phonology" and "Edit Phonology" tools, while the file tab has the "Info" and "Report Bug" tools. The page each tool takes the user to is also called a book. These tools are the main way different views are switched between in Susquehanna. Tool buttons are colored the same color as the tab they belong to as to distinguish between different selected tabs. The toolbox also serves the purpose of showing what language is selected at the top, however the language selection text itself is not a tool, and is not clickable. Each tool is described in detail throughout this manual, for a specific tool's functionality, look at the table of contents to find the tool you are looking for.

For developers: Adding a tool is quite simple in Susquehanna. To add a tool, first find the toolbox you want to add it to. These are located in *net.oijon.susquehanna.gui.toolboxes*. Then, create two files, one called {your tool ID}.png and the other called {your tool ID}-pressed.png. Your tool ID should follow the format of tab name.tool name. Notably, it should be the same ID as the page it is meant to point to. Put these files in *src/main/resources/img*. There is also a template button in this folder to use. Once that is done, create a new *ToolButton* object, using the ID as the string parameter. Like tabs, the name of the tool automatically links its image. Previously, developers had to manually call *createTransferAction()* for each button, but now this is done automatically via button ID. Once this is done, your button will appear in Susquehanna!

For developers: Adding a new toolbox is a bit more complicated than adding a tool. To add a toolbox, first create a new class in *net.oijon.susquehanna.gui.toolboxes*. Make sure this class extends Toolbox! Next, either choose a background from *net.oijon.susquehanna.gui.resources.Backgrouns*, or add a new background to that file. To add a background, create a *public static Background*, with the file name, x and y repeats, and background size as parameters. The two main BackgroundSizes are *BackgroundSize.DEFAULT* (provided by JavaFX) and *STRETCH_TO_FIT_SIZE* (provided by Susquehanna). You could also make your own BackgroundSize based off *STRETCH_TO_FIT_SIZE*. Once a background has been either selected or created, head back to your new class and, in your constructor, add *super(Backgrounds.{your background})* as the first line. Make sure to import the backgrounds class! Then, add a few ToolButtons using the tutorial above. Once done, go into *net.oijon.susquehanna.App*, and where your books are instantiated, set their toolbox to your new toolbox using *{book variable name}.setToolbox(new {your toolbox class}())*. This adds the toolbox to the given book. However, to access the toolbox, there must be at least one book with the given toolbox that can be accessed directly via a tab. To see how to create tabs, see Section 1.1.

1.3 Book

Books are the main display of Susquehanna, and are responsible for allowing the user to edit their language. Books display whatever is needed for their attached tool. Typically, books have two sections, those being the left and right pages. However, some books, such as the View Phonology book, only have one page that stretches out for the whole display. Books and tools are interlinked in that books are only accessible through tools, and each tool has a related book.

As such, the names of the books in this manual is the same as the name of the tool for said book. Each book is described in detail throughout this manual, for a specific book's functionality, look at the table of contents to find the book you are looking for.

FOR DEVELOPERS: Creating a new book is similar to creating a new toolbox. First, create a new class in *net.oijon.susquehanna.gui.scenes.**, where * is replaced with the tab the associated tool should be a part of. In your new class, make sure to extend Book! Then, create a public constructor, and initialize the superclass inside it. Then, set your ID and toolbox. Book IDs should follow the format of {*tab*}.{*tool name*}, as some functions depend on the tab being a part of the ID. Once that's set, create your various JavaFX components. To add them to the pages, use *addToLeft()* to add them to your left page, and *addToRight()* to add them to your right page. You may also want to change the font of labels so that it is standardized with the rest of the application, using the *OPENSANS* font in *net.oijon.susquehanna.gui.resources.Fonts* will make the font the same as other labels in Susquehanna. If your book allows data to be edited, create a *refresh()* function that will re-render various components when called. If you would like to utilize the embedded OLog in Susquehanna, you can access the log by using *App.getLog()*. After your book has been designed, add it to the books registry in *net.oijon.susquehanna.App*. This can be done via *books.add(new {your book class}())*. Set a tool and perhaps a tab to point to your book (see Sections 1.1 and 1.2), and you should have a functional book in Susquehanna!

Chapter 2

The File Tab

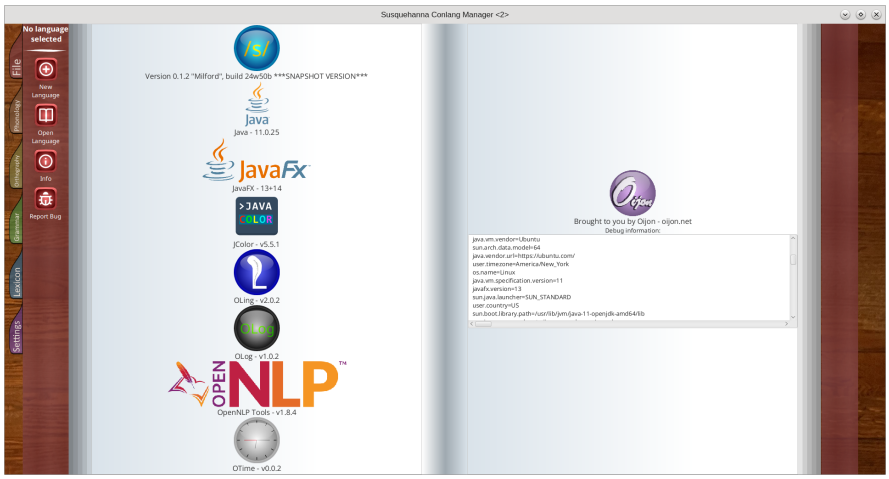

Figure 2.1: The File tab, when first clicked.

The first tab a user will encounter when starting up Susquehanna is the File tab. This tab is responsible for things such as selecting a language, seeing debug information, and sending bug reports. In short, the file tab allows the user to change what is being worked on, and report when anything goes wrong.

2.1 New Language

The New Language book allows the user to create a new language. This language can then be edited with other tools in Susquehanna. At the moment, the New Language book is a bit bare-bones. There are two text fields, that being the Language Name and the Language Autonym. Although the Language Name is listed as being unable to change, this will change in the near future. To create a language, simply type the information needed into this form, then click the "Create!" button. In the future, more information will be able to be added on this book, such as a language image and relations to other languages.

Figure 2.2: The New Language tool button.

2.2 Open Language

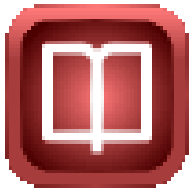

Figure 2.3: The Open Language tool button.

The Open Language book allows the user to open a previously created language and edit it. This book consists of a list of all languages available, with information such as the name, date created, and date last modified. From this page, the user can either select or delete languages. To select a language, simply click the "Select" button underneath the language. To delete a language, click the "Delete" button to the right of the "Select" button. A popup will appear asking if the user is sure they want to delete the given language. Selecting "Yes" on this popup will delete the language. There is also a "Refresh" button on this page. If the user has created a new language, and despite this the new language is not appearing in the list, they can click the "Refresh" button to refresh the list, showing the new language.

FOR DEVELOPERS: Language files are stored in your home directory. On Linux, this can be found at *~/Susquehanna/*, and on Windows it can be found at *C:\Users\{username}\Susquehanna* on a typical configuration. The directory is created in the *user.home* folder provided by Java, so if the folder cannot be located in these places, check your *user.home*. Files with the *.language* extension are those that store languages, and the file names correspond to the name given to them when first created via New Language. Other files, such as phonological systems, glossing systems, and logs can also be found in this directory.

2.3 Info

The Info book used to be the first book to appear when opening Susquehanna, but now is instead simply a way to view debug information. On the left page, this book contains version information for Susquehanna, along with each of its dependencies. On the right page, system debug information is present, and can be easily copied and pasted somewhere else. This page also displays if the currently used build is a snapshot, and if so its exact build ID.

Figure 2.4: The Info tool button.

> For developers: Make sure version information is correct before releasing a new version! If it's out of date, it can be changed in *net.oijon.susquehanna.SystemInfo*.

2.4 Report Bug

Figure 2.5: The Report Bug tool button.

The Report Bug book allows both suggestions and bug reports to be added from inside the application. This section requires a GitHub account, as GitHub does not support anonymous bug reports, so this page will prompt with a login. Once logged in, one will be brought directly to the new issue page. This page contains an embedded browser, so one can also browse GitHub from here. Before reporting a bug, it would be a good idea to check to make sure the bug in question has not been reported before. If it has, commenting on the issue saying it is also affecting you may help in fixing the bug.

2.5 Welcome

The Welcome book is the first book to appear when opening the application. On the left, one can find the version of Susquehanna running. On the right, one can find various commonly used tools that do not require a language open to use. Currently, the tools shown are New Language, Open Language, and Language.

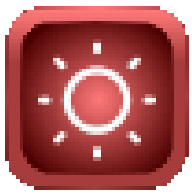

Figure 2.6: The Welcome tool button.

Chapter 3

The Phonology Tab

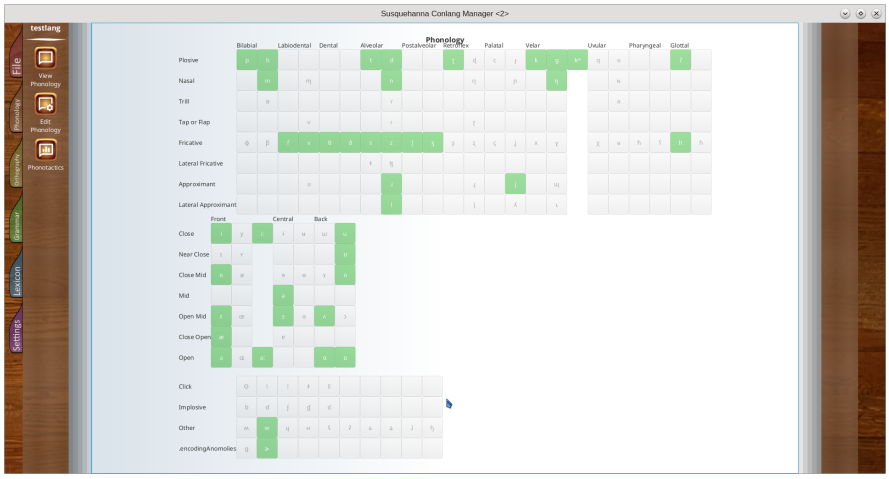

Figure 3.1: The Phonology tab, when first clicked.

The Phonology tab is all about the sounds of a language. This tab is primarily focused on changing what sounds appear in a language, and not much else.

3.1 View Phonology

Figure 3.2: The View Phonology tool button.

The View Phonology book is one of the few mono-paged books. This page allows the user to see the current phonology, without worrying about accidentally clicking on and editing something. For the default phonological system (International Phonetic Alphabet), this view is split into three tables, those being the consonants, the vowels, and the sounds that do not fit in a table in the IPA. Phonemes added to the language are marked in green. Phonemes with diacritics will appear next to their non-diacriticized counterparts.

FOR DEVELOPERS: Susquehanna is bundled with OLing, which has rather powerful phonological tools built-in. If the IPA is not the phonological system you want to use, you can create a new one in the {*user.home*}*/Susquehanna/phonoSystems* directory. Finding the Susquehanna directory is mentioned in Section 2.3. Susquehanna currently does not support switching between phonological systems inside the application itself, however the .language file for your language can be edited to contain the new system in it. Susquehanna will have no problem parsing a different system as long as it is syntactically correct!

3.2 Edit Phonology

Figure 3.3: The Edit Phonology tool button.

The Edit Phonology book allows the user to change what phonemes are in their language, edit existing phonemes to add diacritics, and delete phonemes in their entirety. Like the View Phonology book, it is a mono-paged book. To add a phoneme, simply click on the phoneme, then click the green button on the right of the button to add it. Adding diacritics to a phoneme (such as in the case of /p_h/ for example) is done by clicking the phoneme, then clicking the yellow edit button. From here, a keyboard will pop up with every possible character in your set phonological system. Add your diacritics, then click "Submit" or press the enter key to add diacritics to your phoneme. To delete a phoneme, click the red delete button on the right of the phoneme button. There is no confirmation for this, but if a phoneme is accidentally deleted, it can be rather simply added back.

3.3 Phonotactics

Figure 3.4: The Phonotactics tool button.

The Phonotactics book is currently a placeholder, and has yet to be added. Clicking on the tool will simply give a blank book.

Chapter 4

The Orthography Tab

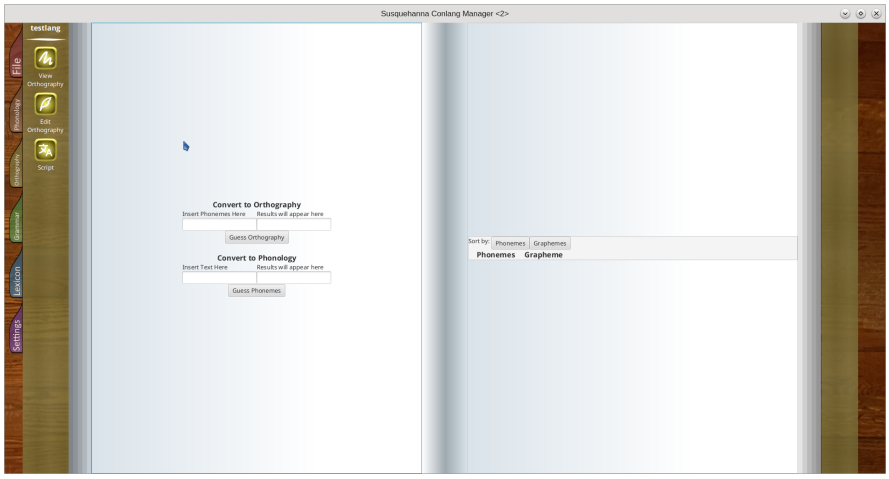

Figure 4.1: The Orthography tab, when first clicked.

The Orthography tab is all about how the language is written. This tab is also one of the less-developed tabs, and most of it is a major work-in-progress.

4.1 View Orthography

The View Orthography book allows the user to see what phonemes correspond to what graphemes. On the left, there are converters between phonemes and graphemes, allowing the user to see how a given text would likely be pronounced, or how a string of phonemes would be spelled. On the right page, all orthographic pairs are listed. Users can sort this table either by phoneme, or by grapheme. In the future, more will be added to this page, for example the ability to view the orthography in a given script.

Figure 4.2: The View Orthography tool button.

4.2 Edit Orthography

Figure 4.3: The Edit Orthography tool button.

The Edit Orthography book allows the user to define new pairs of phonemes and graphemes. This page is a proof-of-concept at the moment, and is not necessarily ready to be used. On the left page, pairs are defined with phonemes going in the left input and graphemes going in the right input. Orthography is also viewable in this book on the right page, just as it is in the View Orthography book.

4.3 Script

The Script book is a placeholder for a planned feature. Clicking on the Script tool simply returns a blank book.

Figure 4.4: The Script tool button.

Chapter 5

The Grammar Tab

The grammar tab is a placeholder tab for a planned update. Currently, the grammar tab does not contain any usable tools.

Chapter 6

The Lexicon Tab

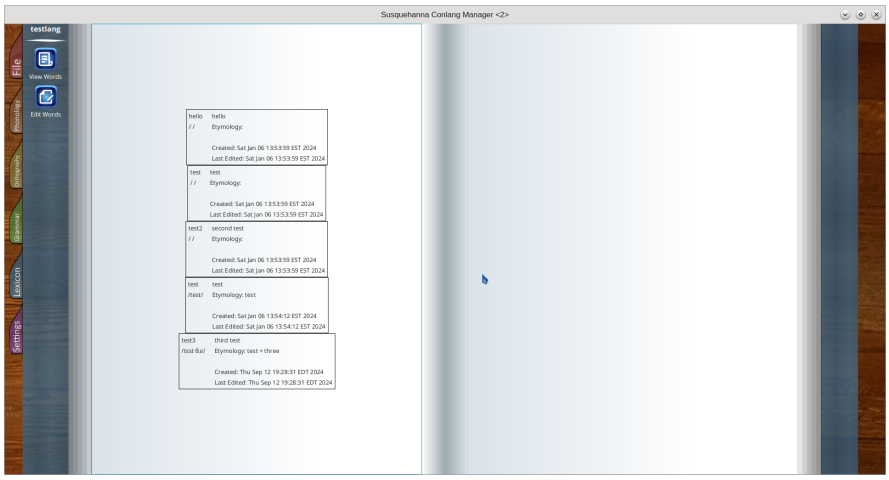

Figure 6.1: The Lexicon tab, when first clicked.

The Lexicon tab is mainly about the words of a language. This tab has tools for adding and finding words, and will have more tools as updates continue.

6.1 View Words

Figure 6.2: The View Words tool button.

The View Words book contains all of the words in the selected language. On the left page, every word available is shown, alongside its pronunciation, etymology, synonyms, homonyms, creation date, and last edit date. This book is somewhat of a proof-of-concept, and is likely to change in the future. One thing that will most certainly change is the display of words, as the boxes are occasionally unaligned. As Susquehanna is still in a pre-release state, this is to be expected.

FOR DEVELOPERS: In the past, words could have source languages attached to them. This was somewhat broken, and has been removed. It will come back in a future update.

6.2 Edit Words

The Edit Words book allows the user to add new words, and to check to make sure they are not adding accidental duplicates. On the left page, a form with the word, meaning, pronunciation, etymology, and source language as options allows the user to add words. On the right page, a simplified word view allows for the user to make sure they are not adding accidental duplicates. Adding a word on this page automatically checks the dictionary for any synonyms and homonyms, and will display as such on the View Words book.

Figure 6.3: The Edit Words tool button.

Chapter 7

The Settings Tab

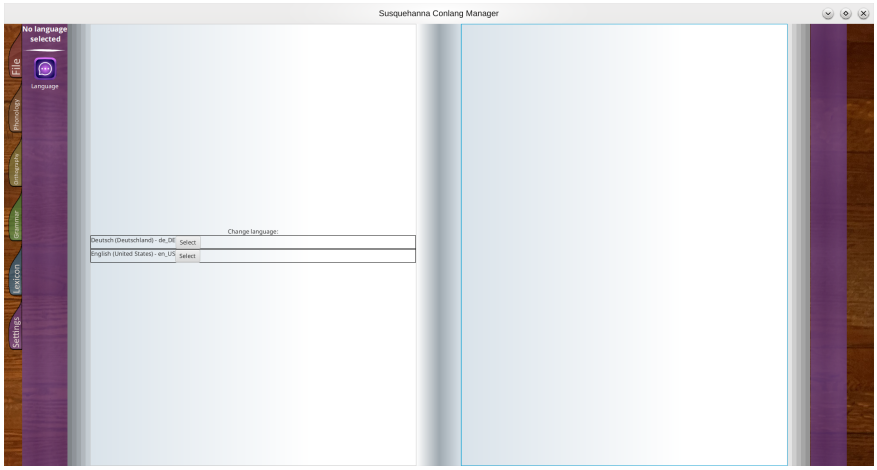

Figure 7.1: The Settings tab, when first clicked.

The settings tab will be the place to change how Susquehanna works. Currently, the only setting that can be changed here is the language, however this is planned to be changed in the future.

7.1 Language

Figure 7.2: The Language tool button.

The Language book allows one to change the language Susquehanna shows text in. Currently, there are two built-in languages, those being English (US) and German (Germany). These packs can be found in the Susquehanna directory under *localizationPacks*. Localization packs added to this directory should automatically appear in the language selection menu. Each localization file is simply a list of text, so translating Susquehanna should be relatively simple. In the future, these files may contain things such as date formats, currency formats, and other things dependent on locale. Furthermore, the ability to create locale files for conlangs is planned for a future update.