Treasure Box Braille (TBB) Application

February 20th, 2018

By:

Risheed Malatombee

Yoni Fihrer

Steve Sohn

Daniel Santaguida

Table of Contents

Requirements

General Use:

Our application is designed to provide an environment that allows the kids and educators to create and edit simple text scenarios with braille prompts. The editing portion of the application allows the user to freely skip, repeat certain phrases or words in the text scenario. The application is designed to be fully functional by a visually impaired user with accessibility containing hover functionality to audibly read buttons and labels. Furthermore, the application also provides audio recording facilities to supplement any text/response with audio.

Use Cases:

The application will specifically allow for the following functions:

- Create a new text based scenario with customized prompts for each text entry.
- All text entries, referred to as nodes, can also have audio recording provided by the user
- A node can have multiple responses, allowing for multiple branch nodes based on which response was chosen. Multiple responses can also point to the same node.
- A pre-existing scenario may also be loaded by the user and edited, either removing, adding or modifying nodes and/or any responses to each node with respect to skipping and repeating functions.
- Each response may provide an audio cue recorded by the user most typically by hovering over the Java components, or by after entering a prompt on a text field.
- The application will also support playing of a scenario, allowing users to preview their scenarios.
- Audio recording is activated at command allowing the user to record and manipulate that recording whether textually, or audibly.

Acceptance Test Cases:

- Does the application provide a description in English of every element within the application?
- Does the application allow the user to create a scenario from scratch within the application itself?
- Does the application allow loading of a text file so that they can preview or edit their scenario?
- Does the application provide audio recording for every node or response created for each node?
- Does the application provide scenarios built to have a branching path or a linear path?

- Does the application accurately display braille as indicated by the user?

- Does the application accurately pick up audio recordings for modifications?
- Does the application allow for recording to textually manipulated?
- Does the application allow for freely maneuvering between creating, recording, editing nodes?
- Does the application demonstrate specified pins?

User Manual

Introduction:

To demonstrate to the user precisely how the application is to be installed, and typically used, below we have included this manual highlighting the installation process, as well as the major use cases and who this appeals to, as well as screenshots demonstrating how the application should behave and operate.

Rationalizing the installation, the installation is highlighted to simply show the user the how's, where's, and what to do when encountering the application icon.
The major use case simply outlines the can's, thus demonstrating to the user if this is right for them. Finally, the screenshots are shown to have a clear idea of what should happen, and how each interface connects to one another.

Installation:

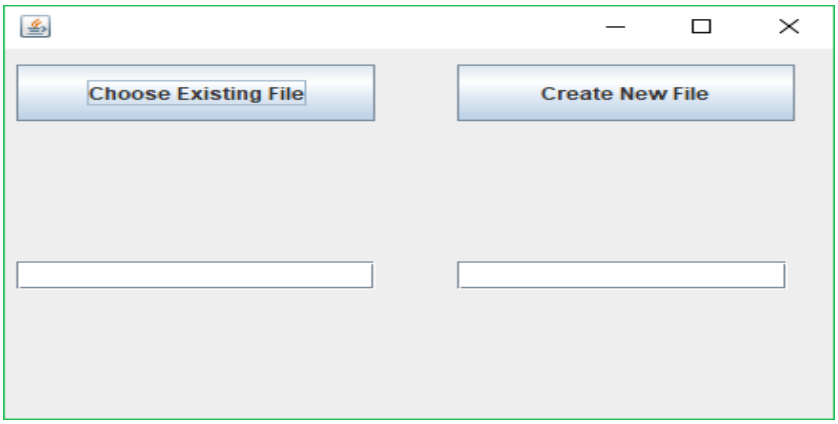
- Upon having the application found as a .jar file, double-click on the application icon, prompting you to choose a storage space for the application itself.
- Upon storing the application's destination for storage, the application will run with the designated application prompts.

Major Use Cases:

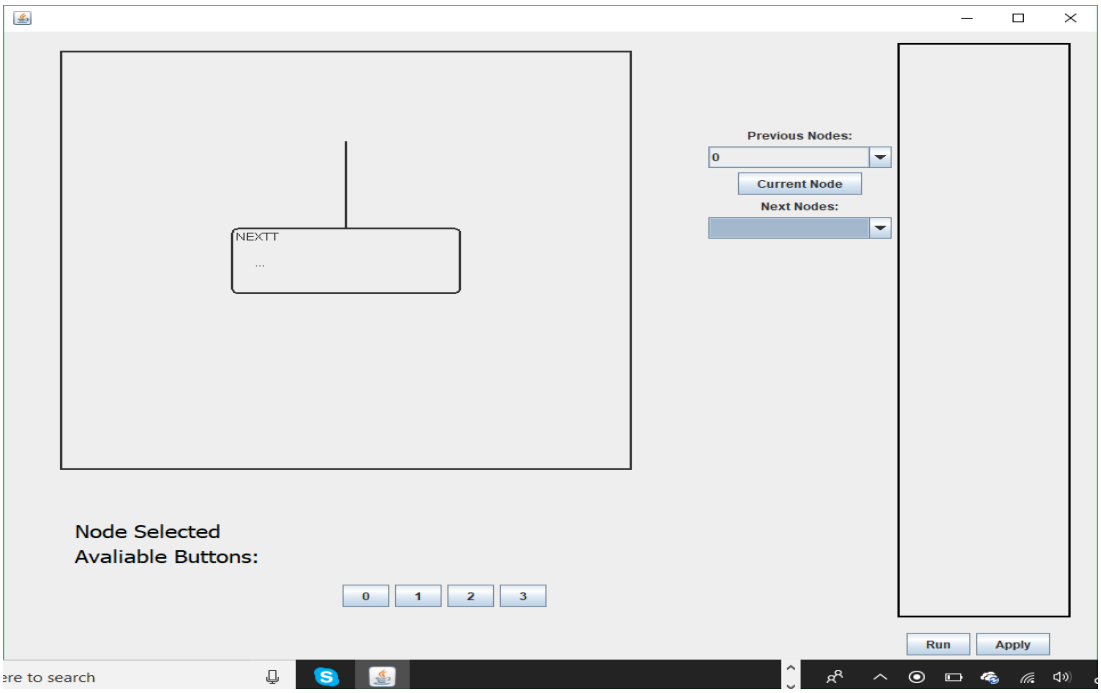
- Application is used to fulfill the task of reading documents, books and other textual representations using pins.

- Application to aid user in creating documents and other textual artifacts from scratch.
- Application is used to capture the user's voice and transmit that to a textual representation.
- Application is used to allow users to freely choose between creating, editing, and recording files.
- Application is used to allow user to preview they're text before creating a file, or editing one.

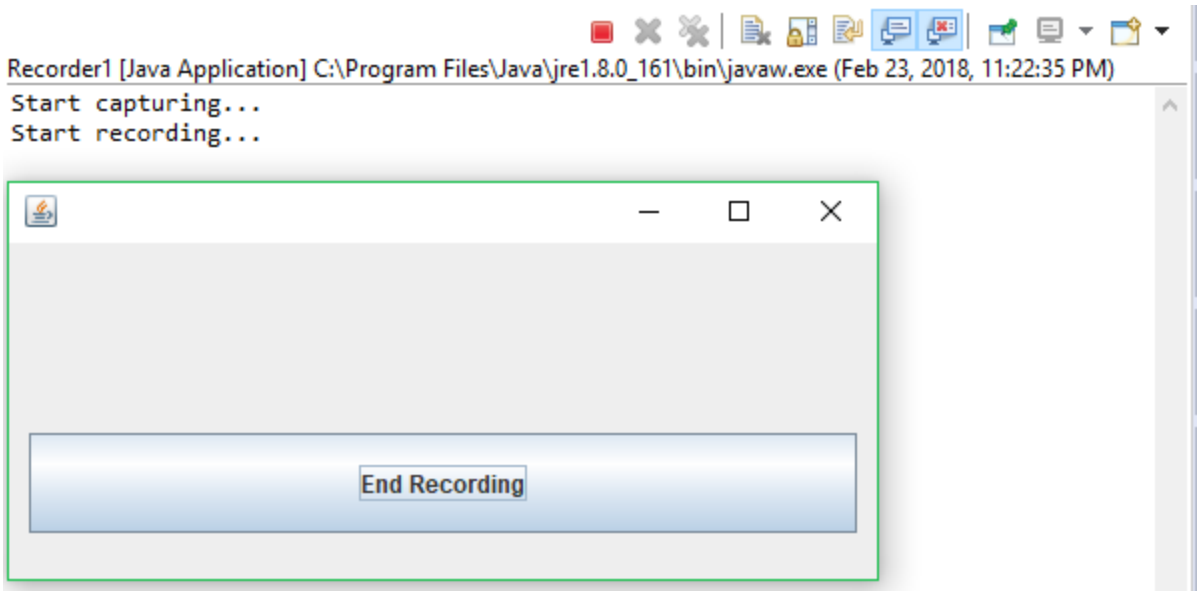
Screenshots + Video Links:



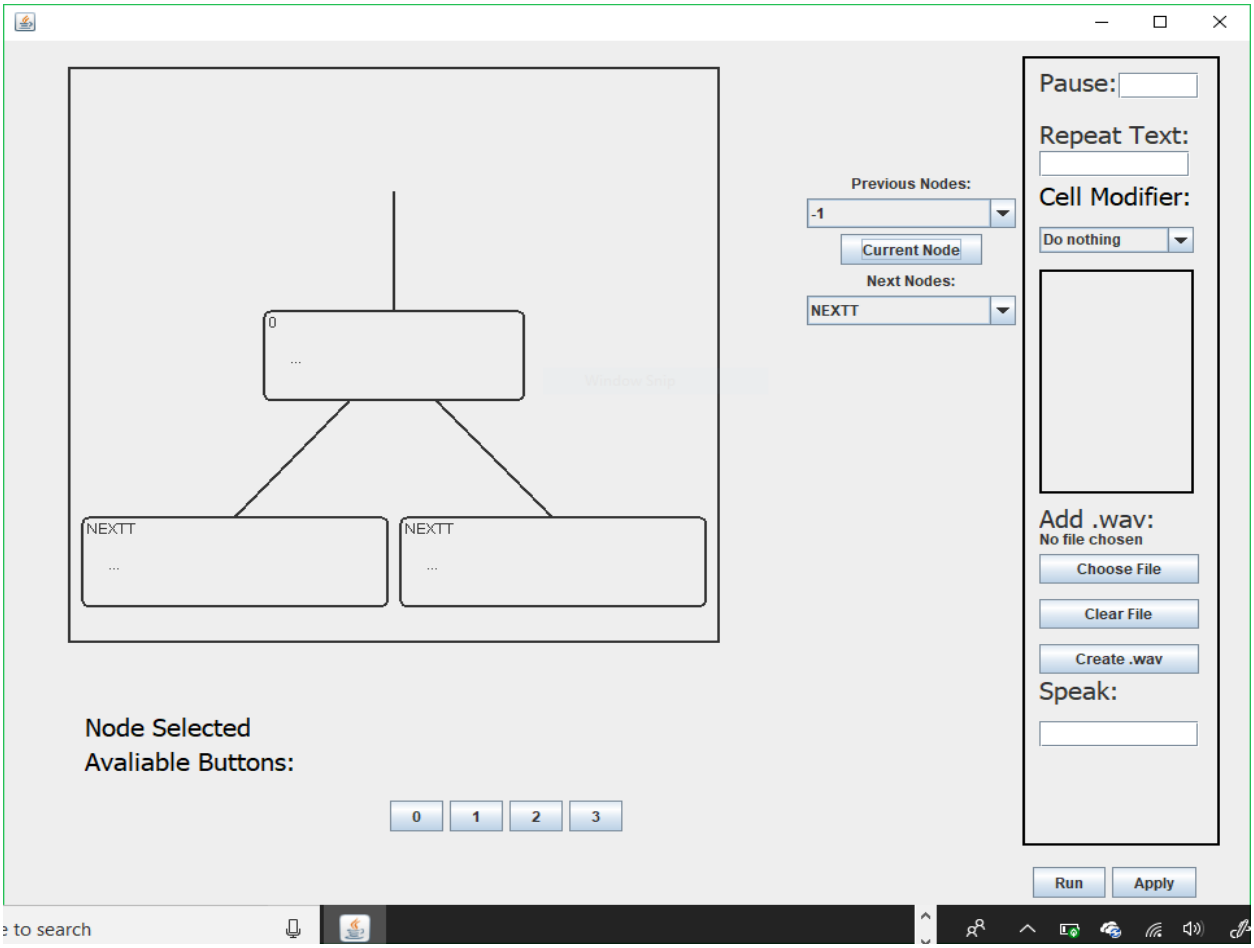
Start Screen with option to choose file or create one.



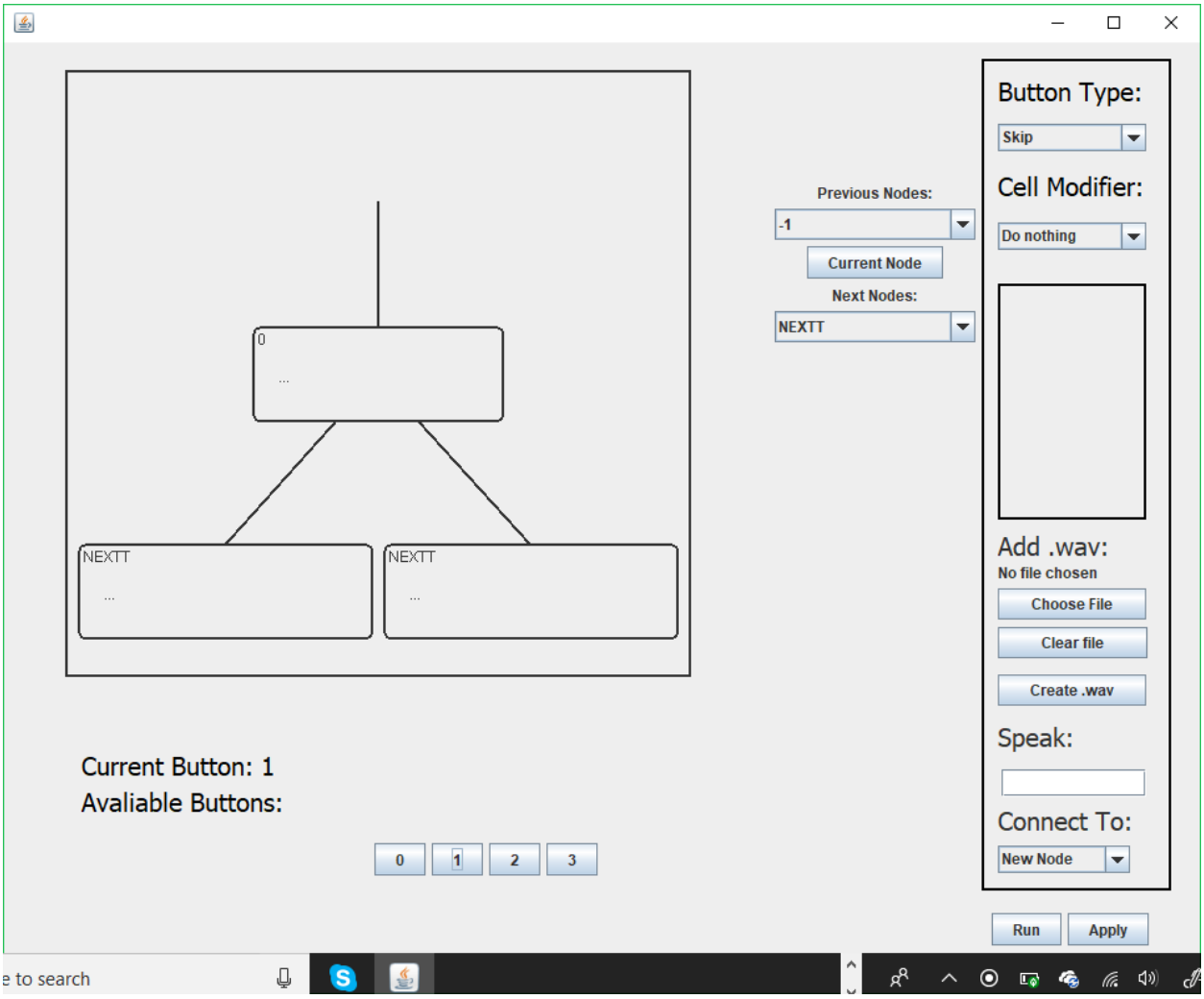
Editing Screen upon start up after the Start Screen



Recording screen highlighting start and end of recording button.



Editing Screen with use of the current node button



Editing Screen with use of available buttons

Testing

Introduction:

To ensure accuracy and satisfactory performance of the application, we have outlined what functions need to be tested. It is especially important to test that every braille prompt is correct in its representation. We also test to ensure the application follows the correct path, particularly in the case of branching (i.e., non-linear layout).

In our preliminary launch of the application, we do not expect to have all testing in place. Much of the application lies in the graphical user interface, which in our determination, is best tested visually by running the application.

Implemented Tests:

Test	Expected	Actual
Braille character representation	Each letter of the English alphabet is correctly represented in Braille	Only lower-case letters of the English alphabet so far correctly depicted
Graphical structure integrity	Each time a button or node is added, it is visually and audibly updated by the application	Graph demonstrates the button and node action
Button prompts are followed	Scenario is followed correctly as Directed by user	Scenario is followed correctly as directed by user
Screen reader reliability	Every graphical element is relayed audibly to the user	Graphical elements have accessibility contexts read aloud

Rationale and Sufficiency:

It would not be difficult for non-braille users (in this case, the developers) to overlook any incorrect braille representations. As such, it would be particularly important to test and ensure the accuracy of any braille represented in our application.

The application also incorporates a small number of menu prompts, leading to different “areas” of the program. For example, on the main menu screen upon loading of the application, the user has the option to create a new scenario file. We would ensure that upon selecting that option (or any option for that matter), the application updates the text and buttons on the screen to be consistent with the part of the program that facilitates the creating of a new scenario.

As such, with the above tested for and the scenario being adhered to as expected, the basic functionality of the application would be achieved.

Furthermore, we test for correct audio cues being played when a prompt is selected, such as a "correct ding" audio file being played when the correct response to a question is selected.

Lastly, accessibility context is provided for all visual graphical elements that are present on the screen. As such, anytime a graphical component (such as a node, or the text within the node) is updated, added, or removed, this information is relayed audibly to the user. We wish to ensure that every graphical element relays this audible element.

Test Coverage:

Due to some parts of the application still being under construction, we are unable to reliably reach a high percentage of coverage. Some classes are unable to be run, and thus tested for. For those methods that can be run, we mostly rely on running the program manually rather than JUnit testing as much of it is graphical in nature.

Future and Improvement:

We are fully aware of the incomplete state of the application, which heavily impacts our ability

to create a test suite that checks all aspects as desired. As such, we rely on manually checking the reliability of our application as opposed to the more ubiquitous method of using a testing suite.