Design Document:
Authoring App Version

Team 3 Members:
Eric Dao, 213551379
Dong Jae Lee, 214461560
Siddharth Bhardwaj, 21358439

EECS 2311 – Software Development Project
Instructor: Bill Tzerpos
Due: April 5, 2017

Table of Contents

1. Introduction 2

2. High-level organization of the Program 3

3.1 Description of RevisedApp 2

3.2 Description of HelpFrame and AuthorFrame 3

3.3 Description of AuthoringDisplay 3

3.4 Description of SectionNode 3

3.3 Description of AudioProgram 3

4. Description of Maintenance Scenario 4

5. Class Diagram 5

6. Sequence Diagrams 6

1. Introduction:

This part of the design document explains the choices made into the design of this program. The descriptions cover a high-level description of the organization of the system along with an explanation of design for the individual parts. Included in this document is a maintenance scenario to depict how to add additional functionality to the program or changing the display. Furthermore, at the end of this document are class and sequence diagrams to give an overview of how the program works and the connections that each module share with each other.

2. High-level organization of the Program:

The design of the program was to follow the MVC (model-view-controller) design pattern, because this program requires the use of user interface to create a scenario file. Although this current program does not implement a true MVC program, as the model, view and controller are not separate classes, but that some classes function both as the model and view or model and controller. The controller of the program is the RevisedApp class, which displays the main frame and menu items for the user to access. The RevisedApp also works in conjunction with the AuthoringDisplay class, which contains the actual buttons to give the user the ability to change the scenario file, in addition to seeing a view of the scenario file and the sections of the scenario file. The model and view part is the AuthoringDisplay class, as the Authoring display carries out all functions and interactions that the user wants when editing the scenario file, opening a scenario file or saving a scenario file. The AuthoringDisplay class uses the AudioRecording and SectionNode class to implement smoother functionalities for the user, and the AboutFrame and HelpFrame classes are used to provide an additional view to the user.

3.1 - Description of RevisedApp:

The design of the RevisedApp class was to create a main window for the user to interact with, which would have menu items to provide the user with access to creating a new scenario file, saving a scenario file or opening a different scenario files. Additional menu features include being able to open a sound recording along with creating new windows that describe how the commands and command inputs function.

3.2 Description of HelpFrame and AboutFrame:

The design for the HelpFrame and AboutFrame classes were to include additional non-essential features that help the user better understand the program. These classes were meant to provide additional information, and are not a main contributor of the overall program.

3.3 Description of AuthoringDisplay:

The design of the AuthoringDisplay was meant to be the model part of the program, but ended up also being the view and controller part as well. The user can interact with the buttons on the panel of the main window to manipulate and change the scenario file. The AuthoringDisplay then takes in these inputs and changes the scenario file accordingly, while displaying these changes to the user to show how those actions affected the scenario file. Ideally, the AuthoringDisplay should have implemented the strategy design pattern, as there are many buttons that the AuthoringDisplay class implements and therefore it would have been easier to have one abstract button to provide easier implementation. However, what was used instead were many if statements to represent what happens when the user chooses a variety of buttons. Although this pseudo-design produces a lot of lines of code to cover each choice and code smells since there are better implementations, it satisfies the customer's needs.

3.4 Description of SectionNode:

The design of the SectionNode was meant to follow the composite design pattern, where a tree can be made up of the SectionNode classes, and the combined tree structure represents the overall structure of the scenario file. The SectionNode contains all the lines for that specific part of the scenario file, and has access to child SectionNode classes to represent the flow of the scenario file, since the scenario file is read sequentially from top to bottom.

3.5 Description of AudioRecording:

The design of the AudioRecording class was to implement the function of allowing the user to record their voice speaking out a phrase, sentence or multiple sentences. The AudioRecording class is an additional feature to satisfy the customer's needs for the program.

4. Description of Maintenance Scenario:

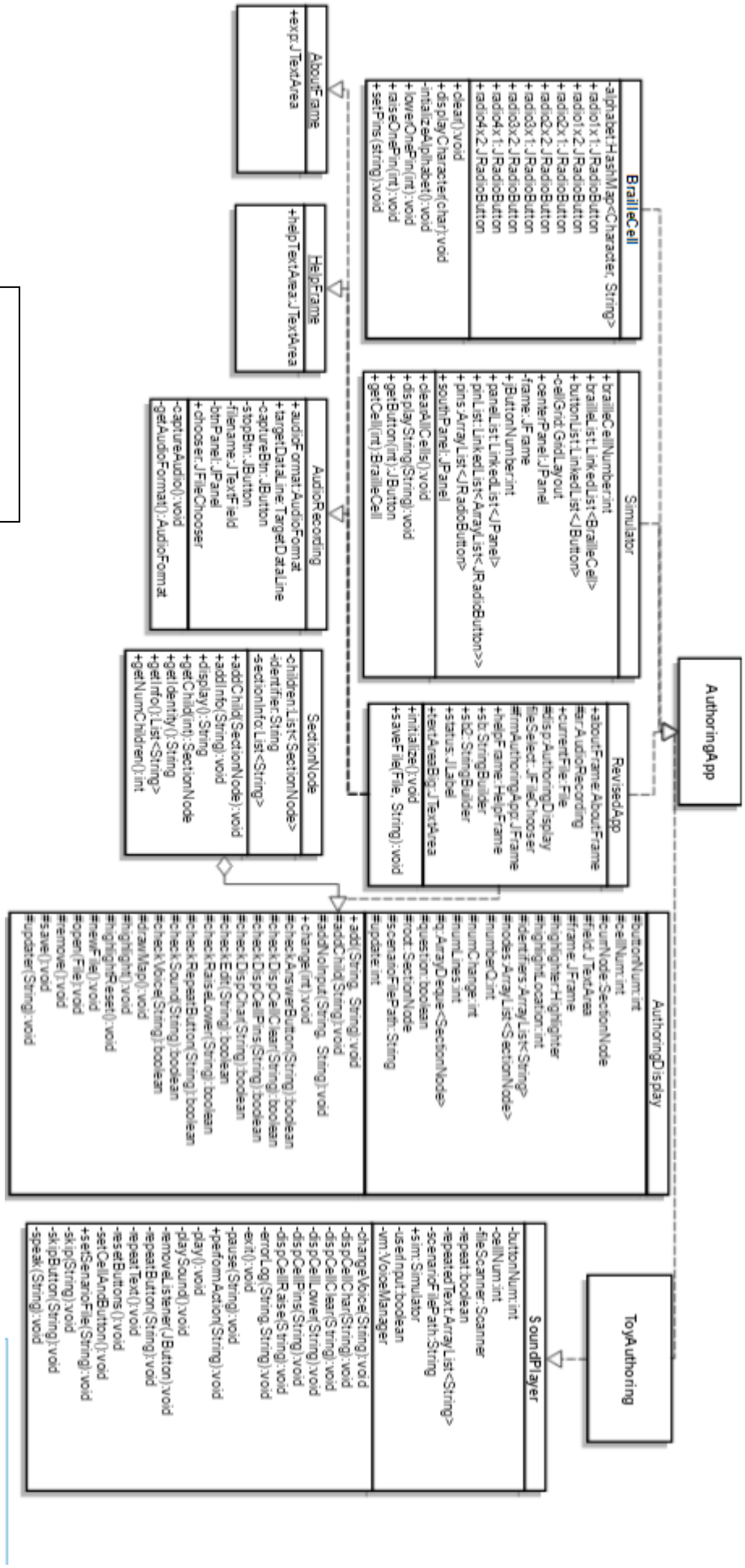
To maintain the program by additional additional features or updating old features, the main classes that should be considered are RevisedApp and AuthoringDisplay. For features that require adding more menu options, or adjusting the size of the main window or anything similar, then one would have to go to the RevisedApp class and add in their features to the initialize () method. This initialize method contains the creation of the main window along with the menu items and what happens when those menu items are pressed.

For adding additional features that affect the scenario file, then the class AuthoringDisplay is the one to add any additional features. For example, if a programmer wanted to add another button that allowed the user to enter a play .mp4 file into the scenario file, then the programmer would have to create a new button and add it to the constructor of AuthoringDisplay. Once they add the button to the panel there, they would then have to go to the add () method and add another branch to the if statement, where that branch deals with adding the mp4 playing line to the scenario file. For changing the display of the sections of the scenario file onto the text area, the programmer would have to change the display () method in the SectionNode class.

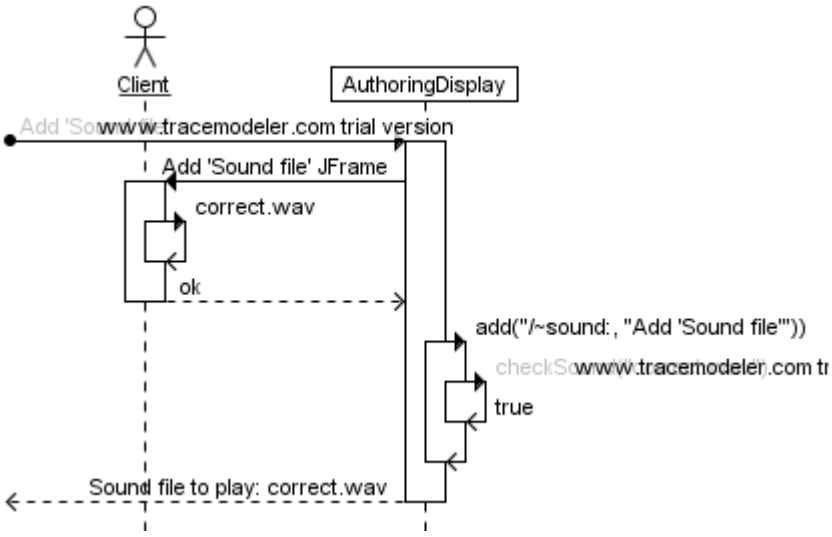
5. Class Diagram:

For clarity's sake, you can view the image following this link:

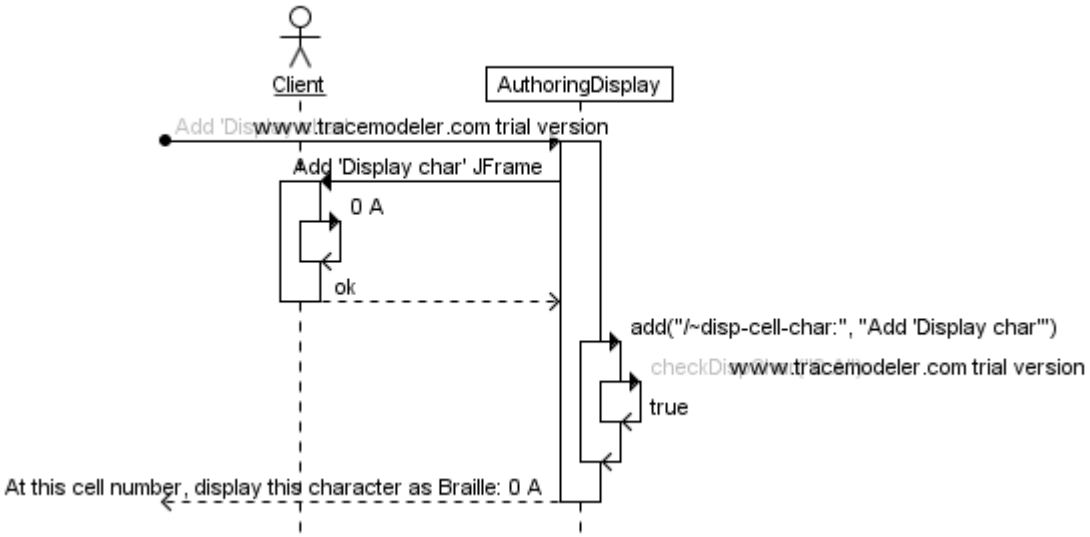
https://www.gliffy.com/go/publish/11917356



6. Sequence Diagrams:



The above sequence class diagram explains what happens when you press the add sound file button.



The above sequence diagram explains what happens when you press the add display cell character button.