**Grand Haven Musical Fountain:**

**Design Document**

4 February, 2015

Michael Kinkema

Nicholas Paquette

Raleigh Mumford

**Overview**

Our project is the Grand Haven Musical Fountain application. Our goal is to finish delivered program, testing code implemented by previous groups, and potentially implementing new features afterwards. In this design document, we will detail several aspects of our project and how we plan to implement those parts of the project.

**Programming Languages, Protocols, Frameworks, and Libraries**

We will be using Java exclusively to write our program. Java will allow the program to be supported across multiple environments. We will be enforcing Java API level 8 as our minimum supported runtime environment. For the GUI, the program will use JavaFX, a high level Java graphics library. JavaFX is the new standard for graphics within Java and will allow us to take advantage of drag and drop GUI design.

The program will generate a control (.ctl) file formatted to the Fountain Command Words (FCW) standard. This is a simple text based language that the fountain’s PLCs can read.

1. The FCW consists of a three digit address word follow by a three digit data word.
2. See the reference document “FCL-03.00.xx” for further definition. It along with sampel CTL files can be found at <https://github.com/ik1nky/GH-Fountain/blob/Final-Project/Starter_documents/Documents%20for%20next%20group/FCL-03.00.18.docx%20%28ask%20for%20latest%20revision%29.docx>

The only external library used by the Choreography software is the Simple JavaFX Player. This is a lightweight and open source audio player that provides an elegant and simple user interface.

**Testing**

Testing will be one of the key aspects of our project. Due to the state of the project as it has been delivered to us, we are unsure of which functions are bug-free, have minor bugs, or do not perform correctly. We will be documenting and fixing each bug systematically, starting with bugs that are frequent in the GUI. Performance will also need to be validated as this is a key concern to the client.

We have been provided with a suite of tools for testing the project,These tools include:

Sample .CTL files

Sample existing playback output

Map of fountain water and LED light modules with addresses

FCL-03.00.xx

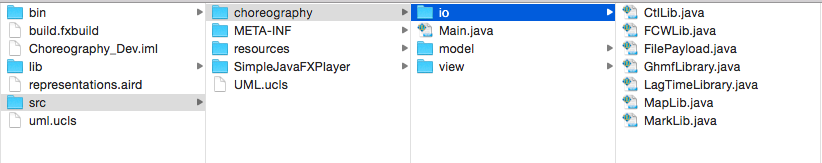
Original Choreography Simulation Software

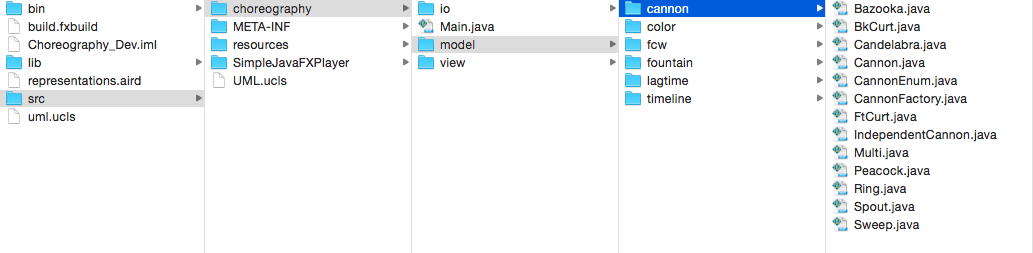
We are also working on getting a comprehensive CTL file that steps through each and every fountain operation. This will give visual proof that each function is running in the correct manner. Due to not having access to the fountain to run and test our code, the only methods available for testing our output is comparing it to the output of the old choreography software’s output as well as manual verification of each file.

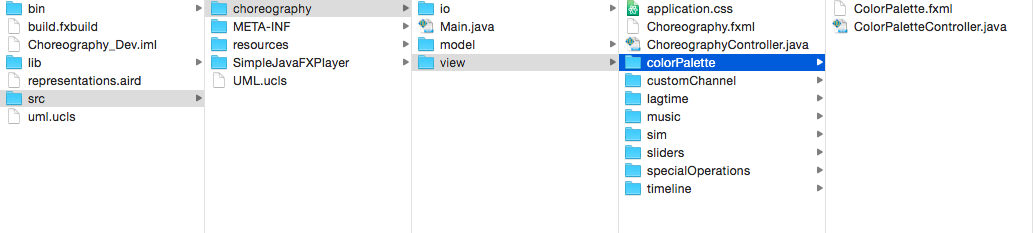
A testing feature that will need to be built will allow any CTL file to be tested, making sure it conforms FCL 3.00.x and that it does not contain any errors or malicious commands. Ultimately it will be up to the City of Grand Haven to decide what is run on the fountain, but this tool should verify that user submitted choreography is safe to run.

**Project Organization**

This is an example of our file structure given to us by the previous team.

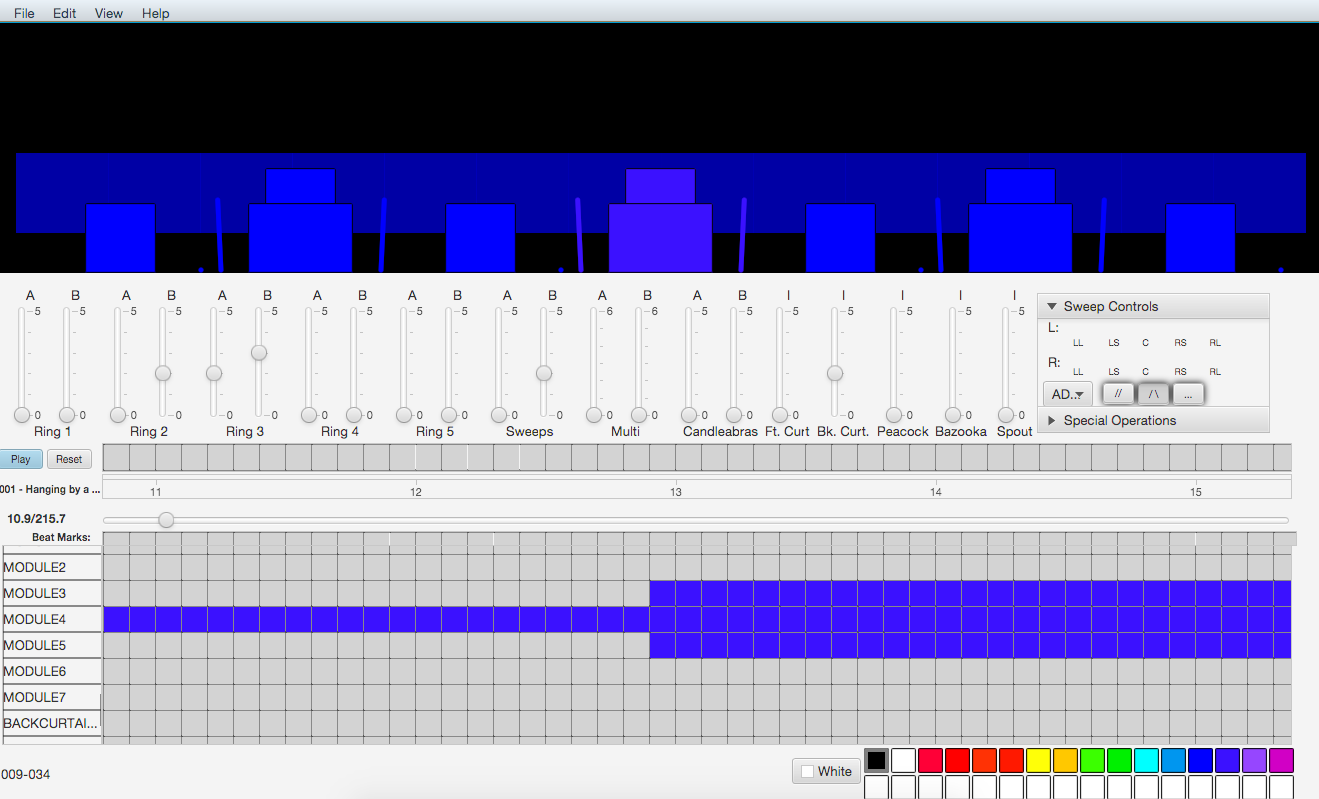






**User Interface**

As said above, our user interface will be created using JavaFX. This graphics library was chosen for its ease of use and drag and drop design. The GUI provides a simulation to test how the fountains will work given a .wav file. The simulation shows both how the fountains will act and what color the lights will be for each fountain.



At the top you see a graphical representation of each fountain along with sweeps, curtains and peacocks. The height or power of each piece of water are specified by the sliders which go from level 0 (off) to level 5 which is the highest. The colors are also displayed by the graphics, for this example everything is colored blue. Below the graphics and sliders is the timeline allows the user to change the colors of the fountains, represented by module 1-7 (each fountain), the curtains, sweeps and peacock (not shown).

**Division of Labour**

Each member of the group will be responsible for testing parts of the code implemented by the previous group. There are a lot of bugs in the program and much of our time in the beginning will have to be dedicated to making sure everything works before trying to implement new code. After the code has been sufficiently tested and is free of bugs, we will each work together to implement new code for the program, testing it along the way. This project will be more about evenly working together as a group than dividing labor amongst specialized team members.

**Design Methodology**

We will be following the Agile development methodology. The Agile manifesto states the values we will be following the best:

*Individuals and interactions over processes and tools*

*Working software over comprehensive documentation*

*Customer collaboration over contract negotiation*

*Responding to change over following a plan*

The only point we will contest is the working software over comprehensive documentation as there is no real reason we cannot have both. A project of this scale will need comprehensive documentation when it requires updates and new features which will be implemented by future developers.

Not only will we be in contact with the client, but we will also have the possibility of contacting and consulting former students and Dr. Adams who worked on this project previously. This is a large project, and with some guidance from them, the scope of the project should be well within our ability.

**Alternate Possibilities**

As we received the Grand Haven Musical Fountain project after significant progress had already been made on it, we mostly were not able to consider alternate possibilities for the project. We were asked by one of the project managers, however, if java was the correct choice for such a project, and if not, which language would be best for this program. We decided that java was the correct language due to its portability and its performance.