MUS 470/470L Project Proposal and Specifications

Fall 2018

Due date: Friday, November 30.

This project proposal and list of specifications is for the course project to be completed in MUS 471L. Each project should exhibit significant work in each of the following three areas:

1. algorithm implementation
2. user interface with parameters
3. audio engine components

Your report should consist of four pages, one page devoted to an overview of the project, and one page for each of the three areas listed above.

More details:

1. In the overview, include details about how much of the work you have set out to do is material that you have significant experience with, and how much is very new to you. For example, if you plan to do an algorithm implementation which is a variation on the reverb project in MAT 321, this would qualify as something you have some experience with. Perhaps you want to implement an ambisonics encoder/decoder, which might be something completely new, etc. If you have very little experience with UI, or you plan to take on a new API, then this should be stated. Also state which of the three areas you plan to put the most time into, based on your proposal.
2. Your algorithm implementation should be, at a minimum, on the order of complexity which is similar to the reverb project in MAT 321, or the Plucked String project in MAT 320, or an ambisonics encoder/decoder, or a spline signal modeler, for example. This requirement cannot be skipped or minimized.
3. The UI is an essential part of the project and is basically self-explanatory. In order to demo the project you should have a nice set of sliders which give the user an intuitive means to explore your work and the various parameters that illustrate your implementation.
4. The audio engine components can vary, but one example would be to implement a plugin which is used in an audio processing graph along with an existing audio engine. For example, if one was doing a reverb project then a plugin would be a nice way to make the reverb real-time. It is not necessary to write an entire stand-alone audio engine, but at least some of the typical functions should be managed in your code. For example, if you implement your own call-back, or manage your own audio buffers, or write a plugin, any of these would suffice.