Virtual Conductor

Gene Kim, Michael Wiktorek, Dustin Walde

Project Abstract

A conducting application that algorithmically generates music which can be directed through hand gestures. The application would be interfaced with the Oculus VR system to display a 3D environment containing instruments and a Leap Motion Controller to read hand gestures to manipulate the generation and playback of the music. This project builds on the work of real-time music generation with a novel interface that roughly simulates conducting.

Project Scenario and Goals

This project would be a leisure activity application for a novel way to interact with digital music. Since the generated music should be original for each use of the system and be able to interact with the user's motions, the music generator needs to be able to run in real-time and respond to user actions in a short enough timespan for the user to understand the interactions between his actions and the generated music. The generated music needs to support a wide array to instrumentations and handle both large and small musical groups in the composition process.

Design Strategy

There will be two primary modules in our project, with a fair amount of division between them. The first will be the "front end", which consists of the virtual environment and Leap Motion controls. This part of the project will generate the 3d graphics required by the Oculus Rift and will accept user input in the form of hand motions via the Leap controller. The "front end" will communicate with the "back end", the other module, which is responsible for actually generating the music that the user hears. This module will consist of an algorithmic composition engine and an API for the "front end", and will provide a constant stream of MIDI data to the front end for playback, modulated in response to user input from the "front end". Our current plan is to implement the composition engine in Java and interact with C# scripts in Unity via OSC.

Design Unknowns/Risks

The most significant unknown factor in our project is the algorithmic composition component. Implementing this will require a significant amount of research and experimentation on our part, both in finding preexisting examples of algorithmic composition and developing our own techniques. None of us have experience developing for the Oculus Rift or Leap Motion, but, while these are also unknowns, they will probably be easier to overcome, given the existence of a solid API that has been used by many developers. Integrating MIDI streams into Unity is also proving difficult.

Implementation Plan and Schedule

Algorithmic Music generation:

- Figure out how to generate MIDI with Java
- Write an API for the composition engine
- Write simple music generation examples
- Parameterize music generation for style and control
- Write more complex music generation code
- Figure out how to different instruments will interact

Leap Motion:

- Read the Leap API
- Determine the Leap's capabilities
- Write some example code for the Leap
- Implement Leap control for the 3d environment
- Implement Leap control for the composition engine API via the environment

Graphics:

- Build a virtual environment in Unity
- Generate and test the 3d environment on the Oculus
- Generate resources for the 3d environment (textures, shapes)
- Interface the graphics engine with Leap control and the back end API
- Send MIDI data to Unity and play it back as 3d audio

Timeline:

- Week 3 - Get a handle on how to generate MIDI, get the Oculus working
- Week 4 - Write composition API mark I, write something for the Oculus, figure out Leap API
- Week 5 - simple composition, simple leap control, simple Oculus environment
- Week 8 - More complex composition, multiple instruments, simple 3d Oculus environment with the Leap at least connected
- Week 10 - Leap control over 3d environment, communication with back end API, is fun to demo

Division of Labor:
Gene - get started on composition algorithms, Java + MIDI
Michael - get the Oculus working, work with Unity
Dustin - get started on the Leap API

Evaluation

Our final measure of success will be whether or not we get a working project. Does the algorithmic composition engine produce something that sounds even a little bit good? Does the front end correctly render the MIDI output of the back end into sound? And can the user's hand motions change the sound output of the program in the way we want? If the answers to any of these questions are "no", then we have more work to do. Once we've achieved these goals, we can begin working on stretch goals like improving the composition engine or improving the graphics.

Related Work
http://www.psfk.com/2014/05/touchscreen-robot-orchestra.html - Virtual conductor. This also uses the Leap Motion to read the conducting movements of the user and takes addition input through a touchscreen, but plays pre-composed (and pre-recorded?) music by Felix Mendelssohn rather than generationally composing the music. This system mixes the music in read time in reaction to the conducting of the user.
http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4561863 - Opposite of what we're doing, an automatic conductor that interprets the score and reacts to the choices of the musicians.
http://peterlangston.com/Papers/amc.pdf - Outline of methods for algorithmic composition.
http://arxiv.org/ftp/arxiv/papers/1402/1402.0585.pdf - Another survey of methods of algorithmic composition. Previous work in each method and successes.