Nimbe

Video of the result: https://www.youtube.com/watch?v=MDxWTbbKq2M

(The video was made by simulating fake SMS messages.)

Introduction

Nimbe is a prototype for an interactive audiovisual performance with two main aspects – A visual one, projected at a screen, and an audio one for 8 channels, for an octophonic (a circle of 8 speakers) sound system. For this version, the sound is adapted to stereo.

The audience can send SMS messages during the performance to control a track that is part of the result. That track also has a visual counterpart, which shows as an arc. The users can then try to follow which arc and try they control.

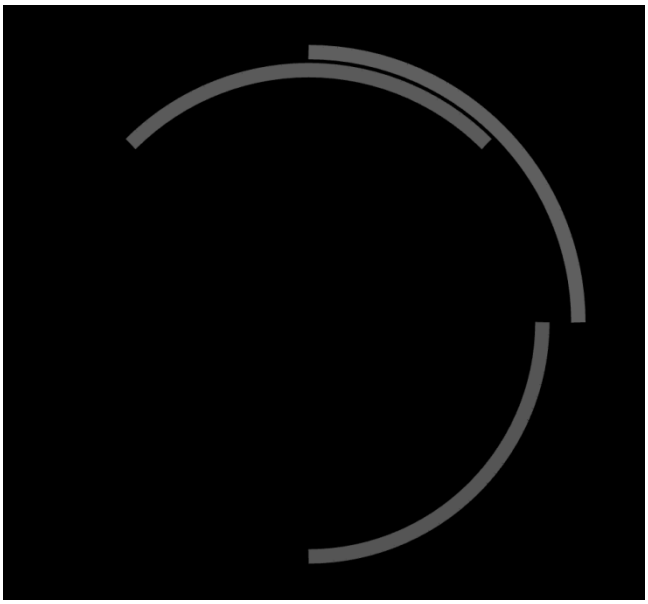


Figure 1 - 3 arcs controlled by users, each have their own associated track

The arc and the track

Each user controls a single arc that is associated to a track. There will be one for every member of the audience that chooses to take part in the performance by sending SMS messages.

The track is the sound that the user controls. Each arc has its own associated track that produces one constant sound at a specific frequency.

The track follows the movements of the arc. The further the arc is from the center, the higher the pitch of the track is. The origin of the sound from the arcs also follows their placement around the circle. If the arcs are located north, north-east and south-east, as shown in the picture, the sound from the tracks will come from those locations in the circle of speakers.

The tracks together are all part of a single chord. While users can control the pitch of the track, those pitches will always stay within a specific chord.

The SMS messages

These are the following messages that the users can send to control their track. (Any message containing one of these commands will also work).

On: Will open the track, either at the start, or after it has been turned off.

Off: Will close the track and mute its sound, until it is turned back on.

+: Will raise the pitch of the track, and move the arc accordingly.

-: Will lower the pitch of the track, and lower the arc accordingly.

North/South/East/West: Changes the origin of the sound of the track in the octophonic circle of speakers. The hybrid cardinal points, such as North-East, or South-West, will also work. Those cardinal points can be written in either French or English.

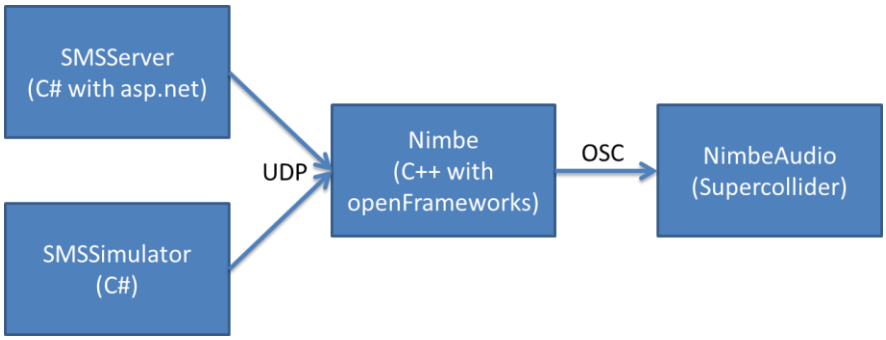
The performance

The performance lasts around 15 minutes. Over that time, different events will happen, changing visual and audio aspects of the performance. The color, shape and movements of the arcs will change overtime. The chords will change, as well as the tone of the sounds used to produce the audio. Those events are not controlled by the users, they have been written in advance in the code.

This has been performed only once, at the *Université de Montréal*, in May 2017, during the concert series *Ultrasons*. The program has since been improved.

How it works

Nimbe is formed by 3 programs that run at the same time, communicating with each other.



Receiving or generating the SMS

The First program is the one that receives the SMS from the users, SMSServer. It uses Nexmo as an API to receive SMS. It then sends the contents of those SMS to Nimbe using the UDP protocol.

Alternatively, SMSSimulator can be used. It is a simple C# program that simulates SMS, and then forwards them to Nimbe using UDP. It is used for testing.

Triggering events and drawing visuals

Nimbe is the core of the project, and by far the largest of those programs. Its main roles are to analyze the received SMS and do their requests, generate the visuals of the performance, and trigger the different events that make the performance evolve. It is also responsible for the tracks, their pitches, volume, pan, etc. While it does not generate the audio, it is the one who controls everything related to it. It then sends data to NimbeAudio using the OSC (Open Sound Control) protocol to NimbeAudio.

Nimbe is programmed in C++ using the openFrameworks toolkit. OpenFrameworks is an open source toolkit that is built on top of OpenGL and allows higher level functions to generate or process visuals and other utilities.

Generating the audio

The last program, used solely to generate the audio, is written using SuperCollider. SuperCollider is a high-level language used to generate and process sound. It receives commands in OSC from Nimbe, and triggers/transposes/moves the played samples accordingly.

How to run it

Set up the server

Option 1:

First, ngrok is used to host a local server that will receive the emails. Enter the following command line into ngrok:

ngrok http 8080 -host-header="localhost:8080"

Then, go on the Nexmo website. Log onto your account, and go into the settings. Change the callback for inbound SMS to the ngrok forwarding address, and add /SMS/Receive at the end of it.

Finally, launch the SMSServer c# application from Visual Studio. Before launching it, it is important to update appsettings.json to your own Nexmo api key, secret and number. Sending a few messages and making sure they are received properly is probably a good idea, they should write themselves in the output of the application.

Option 2:

Alternatively, run the SMSSimulator C# application in Visual Studio. After a wait of 20 seconds, this simple code will simulate SMS messages containing the different possible commands to control a track. They will keep on going until the application is closed.

Run the SuperCollider script

To run the supercollider script, first, it has to be opened in the SuperCollider IDE. Then, to execute the code, parts of it have to be selected and executed separately.

First, select the lines from “s = Server.local;” to “s.boot;”, and press shift-enter (lines 14 to 17). That should start the local sound server, wait until it has started properly.

Then, select the lines from 23 and everything that follows, and execute them as well, with shift+enter. The sound should be ready to play.

Run Nimbe

Finally, run the Nimbe application in Visual Studio (since there is no release yet). The application requires openFramworks to run. It has to be placed in “…/openframeworks/apps/myApps”. Set the window to either fullscreen or windowed mode in the main depending on what you want. Once the application has started, press “Spacebar” to start the performance. The SMS received before spacebar is pressed will be ignored.