Concerto Para Piano e Pandemia

Disponível em: https://www.youtube.com/user/malkabass

O Concerto Para Piano e Pandemia foi desenvolvido no âmbito das disciplinas de Projeto Interdisciplinar e Informática Musical do mestrado em Produções e Tecnologias do Som da ULHT. O projeto surgiu a partir de um ímpeto meu de estar artisticamente conectado com os acontecimentos do ano de 2020, como a pandemia de covid-19 e a entrada de Portugal em estado de emergência a partir do mês de março de 2020.

A obra é uma instalação sonora virtual que utiliza a API do Twitter para coletar em tempo real tweets que contenham o termo covid e convertê-las em sons que são transmitidos para o YouTube.

Para o desenvolvimento da obra utilizei a linguagem de programação python, o supercollider e scripts em Shell. A obra funciona a partir de um Raspberry Pi 4 e precisa apenas de uma fonte de energia e uma conexão com a internet para começar a operar, o que deixa a obra muito fácil de instalar, inclusive de forma remota.

As linguagens utilizadas no projeto não são de meu domínio e quase tudo que utilizei precisei pesquisar sobre e aprender como desenvolver. Assim acredito que o desenvolvimento dos softwares utilizados nesse projeto teve um caráter de descobertas, investigação e tentativas e erros.

Enquanto um instrumento musical tradicional precisa ser “dominado” pelo artista antes do mesmo começar a compor/criar nele, na informática musical se opera pelo oposto: a partir do ímpeto criativo/composicional o artista busca técnicas para poder desenvolver a performance ou obra.

A obra começou a ser desenvolvida pelo script de python que utilizei para coletar os tweets em tempo real. Nele precisei usar uma biblioteca que se comunicava com a API do Twitter e outra biblioteca para enviar mensagens OSC para o SuperCollider. Para a API usei a python-twitter e para as mensagens OSC a biblioteca pythonosc. Com elas pude coletar os tweets, converter cada caractere para números decimais e enviar uma lista desses números para o supercollider em forma de mensagem OSC.

No SuperCollider eu usei um OSCDef para interpretar as mensagens e disparar o sample de Piano para cada número decimal que representava um caractere do tweet original. Era necessário ler as mensagens OSC que chegavam ao servidor do SuperCollider; mapear as mesmas em uma distribuição exponencial, convertendo as, assim, em frequências; estruturar amplitudes, intervalos entre notas e distribuição panorâmica das notas relativos às alturas definidas; e, por fim, executar o sample de piano com as alterações estipuladas pelos argumentos anteriores.

Para as alturas de cada caractere, decidi considerar o valor decimal do caractere "espaço" como a nota mais grave da peça, dessa forma o valor 32 (que na tabela ASCII corresponde a um espaço na mensagem) correspondia a nota Lá de frequência 27,5hz (primeira tecla do piano). Essa definição do espaço como nota mais grave resultou numa maior incidência da nota lá na obra, dando a impressão de uma frequência onipresente na obra. Para contornar isso e reduzir a sensação de estabilidade auditiva, escolhi que para cada caractere que contivesse o valor 32, eu sortearia um valor entre 32 e 40 para ele, dessa forma quebraria com a frequência onipresente resultante.

Em seguida passei a definir os parâmetros que seriam consequência da altura das notas, como amplitude, intervalo entre uma nota e outra e posição no panorama estéreo do som. A amplitude ficou definida como 1 dividido pelo número decimal de cada caractere, o que resultou em uma distribuição de frequências análoga ao ruído rosa (1/f). Para escolher o intervalo de acionamento entre cada nota (e assim determinar a velocidade com que cada tweet é executado) usei a aleatoriedade para sortear um número entre o valor armazenado no caractere e o número total de caracteres contidos no tweet.

E para definir a distribuição espacial de cada nota usei uma máscara de tendência de probabilidades de maneira que a nota mais grave de todas (valor de caractere 32 que equivale a 27,5hz) esteja disposta ao centro do estéreo e as chances das notas estarem mais para as extremidades crescessem de acordo com a frequência. Dessa maneira as notas agudas soam no extremo esquerdo ou direito (sorteio entre -1 e 1) e as notas intermediárias possam soar em qualquer posição do panorama.

Depois era necessário preparar a obra para ser automatizada, podendo ser reiniciada sempre que necessário, fazendo todos os softwares se configurarem sozinhos.

Para que isso funcionasse eu estabeleci diferentes etapas de inicializações:

1º - Ligar o computador usando o Jack como driver de áudio principal.

2º - Após inicializado a GUI do computador, iniciar o SuperCollider que executa o código do instrumento virtual na inicialização através do arquivo de inicialização (startup file).

3º - Inicializar o OBS já executando a transmissão de áudio e vídeo para o Youtube

4º - Conectar a saída de de áudio do SuperCollider à entrada de áudio do OBS usando o Jack

5º - Executar o script de python que se conecta ao Twitter e começa a enviar mensagens OSC para o SuperCollider e a exibir os tweets em uma janela do terminal que será transmitida através do OBS.

Para que isso acontecesse de maneira ordenada preparei alguns scripts de arquivos .desktop e os coloquei na pasta home/pi/.config/autostart. Dessa forma eles eram executados sempre que a GUI do Raspberry Pi finalizasse sua inicialização. Esses arquivos .desktop serviam para executar outros comandos do tipo Shell que então executam funções da lista.

Após preparar o computador para executar todos os sistemas sozinho, iniciei uma transmissão em fase de testes para perceber como a obra iria se comportar durante a transmissão. Durante esse teste de resistência percebi que depois de algum tempo (mais de uma hora) o áudio da obra começava a ficar truncado, com pouca definição e com alguma distorção digital. Para contornar esse problema adicionei uma última etapa à lista, uma reinicialização de hora em hora para que o computador recomeçasse a transmissão e não se sobrecarregasse com o passar do tempo.

A obra estreou no dia 22 de março de 2020 e está rodando no meu canal do YouTube desde então. Anexo a esse documento vão todos os códigos utilizados na obra.

Nikolas Gomes Ferranddis

Lisboa, 15 de Junho de 2020

Script Python

```
#!/usr/bin/python3

from twitter import *
import re
import time
from pythonosc import osc_message_builder
from pythonosc import udp_client

def read_tweets():
    #definição do termo a ser usado para o concerto
    TERMS = 'covid'

    #my credentials
    token = 'xxxxxxxxxxxxx'
    token_secret = 'xxxxxxxxxxxxxxxxx'
    consumer_key = 'xxxxxxxxxxxxxxxxxxxxxx'
    consumer_secret = 'xxxxxxxxxxxxxxxxx'

    counter = 0

    twitter_stream = TwitterStream(auth=OAuth(token, token_secret, consumer_key, consumer_secret))
    data = twitter_stream.statuses.filter(track=TERMS)
    client = udp_client.SimpleUDPClient('127.0.0.1', 57120)

    for tweet in data:
        print (tweet['text'])
        counter = counter + 1
        time.sleep(0.1)
        if counter < 25:
            continue
        if counter > 25:
            counter = 0
            continue
        var_array = [ord(c) for c in tweet['text']]
        var_length = len(var_array)
        n = 0
        msg = osc_message_builder.OscMessageBuilder(address = '/newmsg')
        msg.add_arg(var_length, arg_type = 'f')
        for ciclo in range (var_length):
            if n > var_length:
                continue
            msg.add_arg(var_array[n], arg_type='f')
            n = n + 1
        msg = msg.build()
        client.send(msg)

while True:
    try:
        read_tweets()
    except (KeyboardInterrupt, SystemExit):
        break
        raise
    except:
        print ("        TWITTER OVERFLOW")
print ("stoped")
```

SuperCollider

```
(
s.options.memSize = 1 * 1024 * 1024;s.options.bindAddress_("0.0.0.0");s.boot;
s.waitForBoot{
    2.wait;
    (
        SynthDef(\piano1, {
            arg out = 0, bufnum = 0,rate=1, amp=1, pan = 0;            var sig;
            sig = Pan2.ar(PlayBuf.ar(bufnum.numChannels, bufnum, rate, doneAction:3), pan);
            Out.ar(out, sig*amp);
        }).add;
        ~p4 = Buffer.read(s,"/home/pi/Desktop/pianoA3.wav");
    );
    3.wait;
    (
        OSCdef(\msgfrompython, {arg msg;
            "----------------".postln;"new msg with ".post;msg[1].post;" notes".postln;
            for (0, msg[1]+1,
                { arg i;
                    if(i < 2) {
                        ~pitches = Array.new(msg[1]);
                        ~amp = Array.new(msg[1]);
                        ~pan = Array.new(msg[1]);
                        ~dur = Array.new(msg[1]);
                    } {
                        if(msg[i]>191) {
                            "jump ".post;
                            msg[i].postln;
                        } {
                            if (msg[i]<32) {//jump
                                "jump ".post;
                                msg[i].postln;
                            }
                            {   if (msg[i] == 32) {msg[i]=rrand(32,40);"32 becomes ".post;msg[i].postln;}{};
                                ~pitches.add(msg[i]);
                                ~amp.add(16*(1/msg[i].pow(0.9)));
                                ~dur.add(75/rrand(msg[i],msg[1]));
                                ~pan.add(
                                    if(msg[i] < 85) {
                                        //primeiro triangulo
                                        ~coin = [-1,1];
                                        ~probLO = msg[i].linlin(32,142,0,-1);
                                        ~probLO.rand*~coin.choose;
                                    }{ //se eh maior que 85
                                        ~coin = [-1,1];
                                        ~probLO = msg[i].linlin(32,142,0,-1);
                                        ~probLI = msg[i].linlin(85,142,0,-1);
                                        rrand(~probLI,~probLO)*~coin.choose;};)};};};);
            k = Pbind(
                \instrument, \piano1,
                \bufnum, ~p4,
                \dur, Pseq(~dur),
                \rate, Pseq(~pitches.linexp(32, 191, 0.125, 19.0273185)),
                \amp, Pseq(~amp*0.5),
                \pan, Pseq(~pan)
            ).play;
}, "/newmsg"););};
)
```

- arquivos .desktop

- Scripts Shell

StartJack.desktop

```
[Desktop Entry]
Name=autojack
Exec=/home/pi/.config/autostart/InitJack.sh
Type=Application
```

InitJack.sh

```
#!/bin/bash
jackd -d alsa -d hw:0 -D &
```

StartSuperCollider.desktop

```
[Desktop Entry]
Name=SuperCollider
Exec=scide %F
Type=application
```

StartObs.desktop

```
[Desktop Entry]
Name=ObsAuto
Exec=/home/pi/.config/autostart/InitObs.sh
Type=Application
```

InitObs.sh

```
#!/bin/bash
MESA_GL_VERSION_OVERRIDE=3.3
obs --startstreaming &
```

StartConnectJack.desktop

```
[Desktop Entry]
Name=ConnectJack
Exec=/home/pi/.config/autostart/InitConnectJack.sh
Type=Application
```

InitConnectJack.sh

```
#!/bin/bash
Sleep 35
Jack_connect SuperCollider:out_1 ObsIn:in_1
Jack_connect SuperCollider:out_2 ObsIn:in_2
```

StartPython.desktop

```
[Desktop Entry]
Type=Application
Name=TerminalOpen
Exec=/home/pi/.config/autostart/InitPython.sh
Hidden=false
Terminal=true
```

InitPython.sh

```
#!/bin/bash
sleep 20
cd /home/pi/.config/autostart
lxterminal -e ./twtcovsc22.py
```

StartReboot.desktop

```
[Desktop Entry]
Name=autoreboot
Exec=/home/pi/.config/autostart/InitReboot.sh
Type=Application
```

InitReboot.sh

```
#!/bin/bash
sleep 3600
sudo reboot
```