You're thinking I'm a good person

For two instruments and electronics (~15min)

Introductory notes

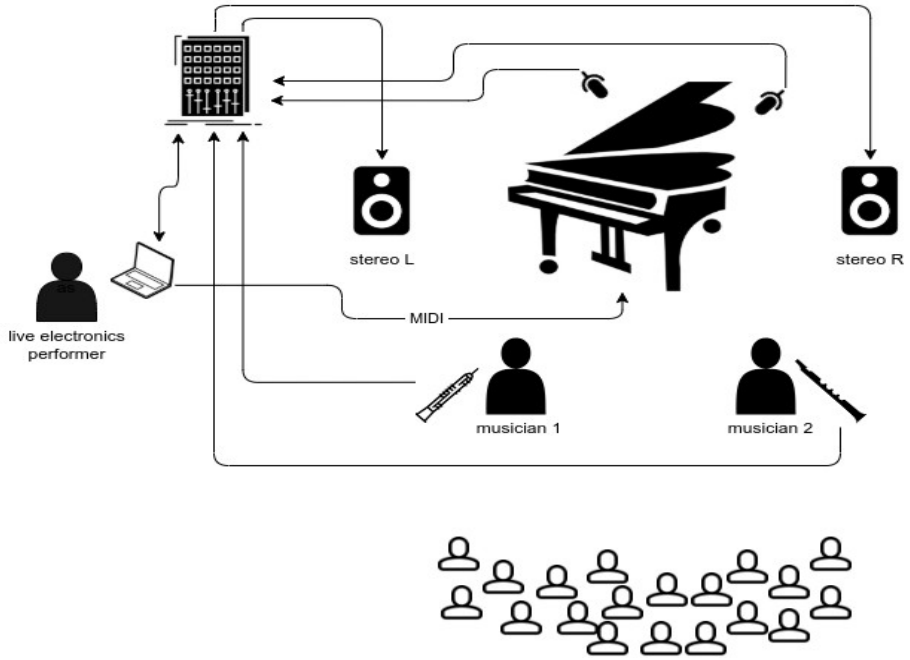
You're thinking I'm a good person is a 15-minute piece for two instruments, a player piano and live electronics. The theme of this piece is related to *social awareness*, more specifically to its most well-known feature: *empathy*. Here, two opposing currents are confronted against each other by algorithms in order to inspire the composition and the performance itself.

This piece is highly flexible in terms of instrumentation, as it is strongly based on a continuous sound approach together with some theatrical performance. Instruments, however, should be capable of producing a continuous sound in a great range with very little attack (i.e. using techniques such as circular breathing, continuous bowing, or sustained by digital effects).

Musicians with interest in free improvisation will certainly be a better match to this experience.

Scene

The figure below illustrates how the stage should be organized for the execution of this piece. The unnamed connections are all audio signals, except the MIDI communication coming from the computer to the *Disklavier*. Although not explicitly indicated in the diagram, the live electronics performer is encouraged to use an external audio device and a MIDI controller to manipulate the effects as described in the next section.



Setup requirements

Hardware

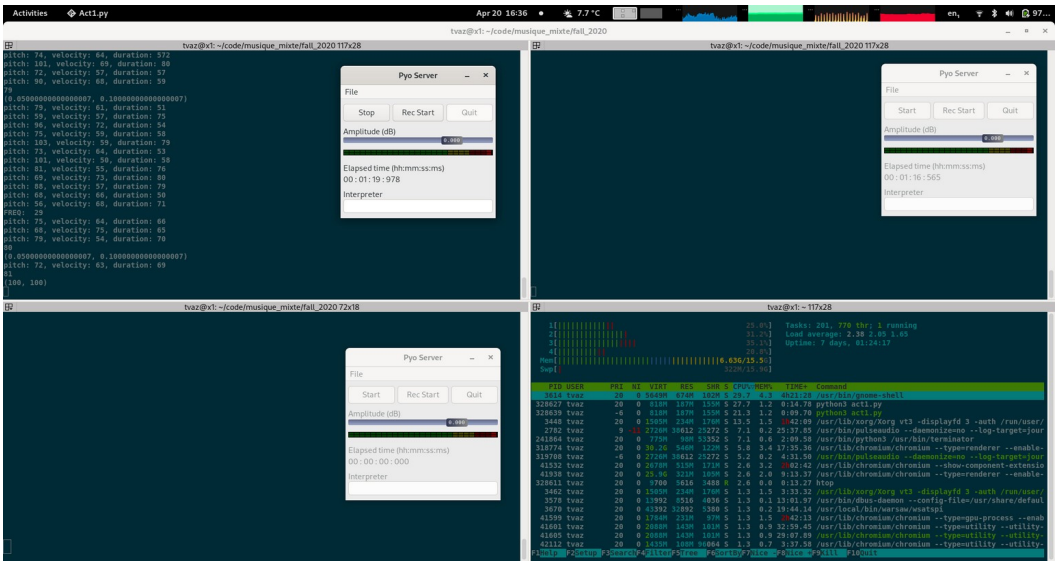
- A computer with an audio/MIDI interface
- A pair of speakers
- Stage spotlights
- A digital player piano
- Two microphones to amplify instruments 1 and 2
- Two microphones to amplify the player piano

Software

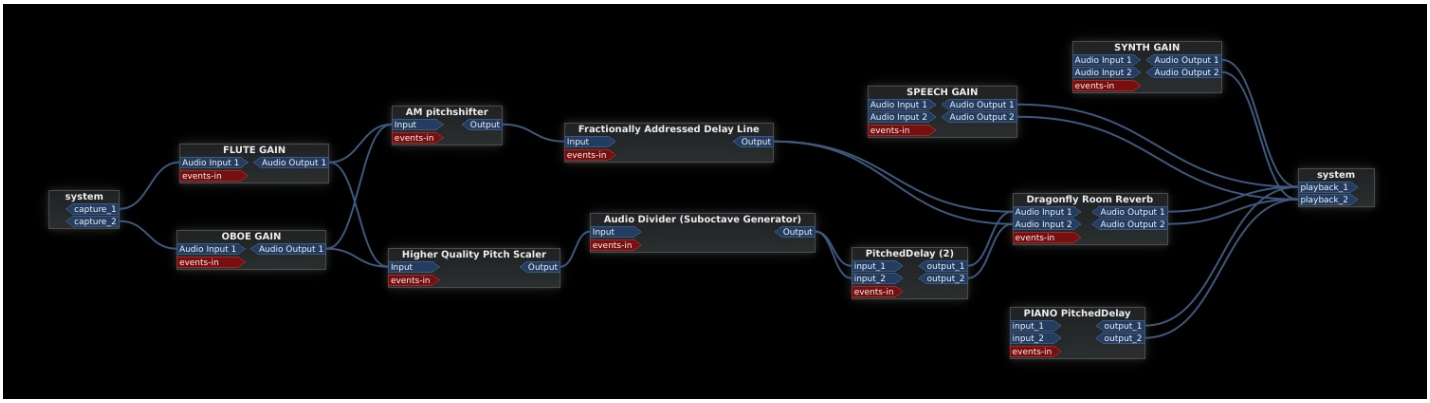
- GNU/Linux distribution (might also run on other systems)
- Python version 3 and a libraries installed as documented in the git repository located at https://github.com/tiagovaz/detaching
- Carla or similar audio plugins host with a similar effects chain as illustrated in the table bellow

Performance instructions

The piece is structured in 3 scenes. Each scene is controlled by a computer code to be executed by the live electronics performer operating the computer. It's recommended to open 3 terminals, one for each *Python* script. Each script can be run through the command "`python3 sceneX.py`" (where X is 1, 2 and 3). This command opens a simple interface in which the *Start/Stop* button should be called sequentially for each scene, as presented in the picture bellow:



The live electronics performer also controls the live effects applied to instruments and to the piano, as described in the attached score. A patch file containing the enchained audio objects and MIDI assignments is provided in the repository. This file (`carla_patch.carxp`) is to be used by *Carla audio plugin host* software. Any other similar software can be used once the following chain is respected (details of each of these effects are present in the table following the image) :



The instructions in the next lines are to be presented to – and assimilated by – all musicians and the lightning operator. Musicians 1 and 2 won't be provided with a score during the performance of scenes 1 and 2. Gesture-based cues by musician 3 (the live electronics performer in the picture) can be an option for them, if necessary. In Scene 3, a music sheet is provided for musicians 1 and 2 and should be followed as described bellow.

Note: as in May 2022, a video recording is available at the Music Faculty of Université de Montréal's Youtube channel and can be used as a reference for the reproduction of the piece: https://www.youtube.com/watch?v=47xVxk-pEM4

Scene 1

- Lights off. The TTS-driven voice introduces the piece by proclaiming the ***Speech #1***
- Musicians 1 and 2 stand side-by-side in front of the Disklavier. They remain quiet; no reaction is expected.
- ***Speech #2*** is played.
- After about 10 seconds, lights progressively on the piano spot.
- Piano plays an algorithmic music and go *crescendo* for about a minute. Musicians stay seated looking at void audience.
- ***Speech #3***

Scene 2

- ***Speech #2*** restarts: "Empathy works..."
- Regular lights on musicians during the speech.
- Same speech is presented 3 times and gets interrupted. After the third time (about 1min15s), musicians wait for a few seconds and start playing an air sound, as if they were trying to perform a proper sound. They play as if the other didn't exist.
- At some point (about 2min30s) the first empathy statement is presented. Musicians start to improvise, keeping no musical dialogue between them. They can play short, noisy, and contrasting sounds, as well as give some time to silence or degrading little phrases from classical repertoire, as if trying to succeed a performance to the audience.
- At some point, after about 5min, a loud piano note is be played. From this time on, musicians stand up and play more intensively, trying to overlap the other musicians' sound, as a subtile competition.
- Once voices (***Speech #4, #5 and #6)*** start overlapping each other, musicians stop playing and leave the scene.

Scene 3

- Low lights spot the musicians. After about 10 seconds, regular lights spot the piano.
- ***Speech #7***: “You must be so helpless, think of Mary...”
- When speech starts, musicians slowly bring their music stands with scores and place them backing each other (so that musicians face each other).
- After the speech, the piano starts playing short notes. Musicians start getting ready to play.
- Hight lights towards musicians as soon as they seem to be ready to play.
- At some point, the piano opens the sustain pedal. From that time on, musicians wait about 10 seconds, quickly look at each other, then start playing their score.
- Right after they finish playing their parts, they leave the scene, put their instruments aside, come back, take the score off the stand, take the stand with another hand and leave the scene with them. This whole process should suggest a total indifference to the music, and to each other; despite the fact that the music is kept playing due to the digital effects.
- ***Speech #8***
- About 10 seconds after musicians leave the scene, musicians lights slowly go off, followed by piano lights going off.
- After another 10 seconds, all sound goes off.

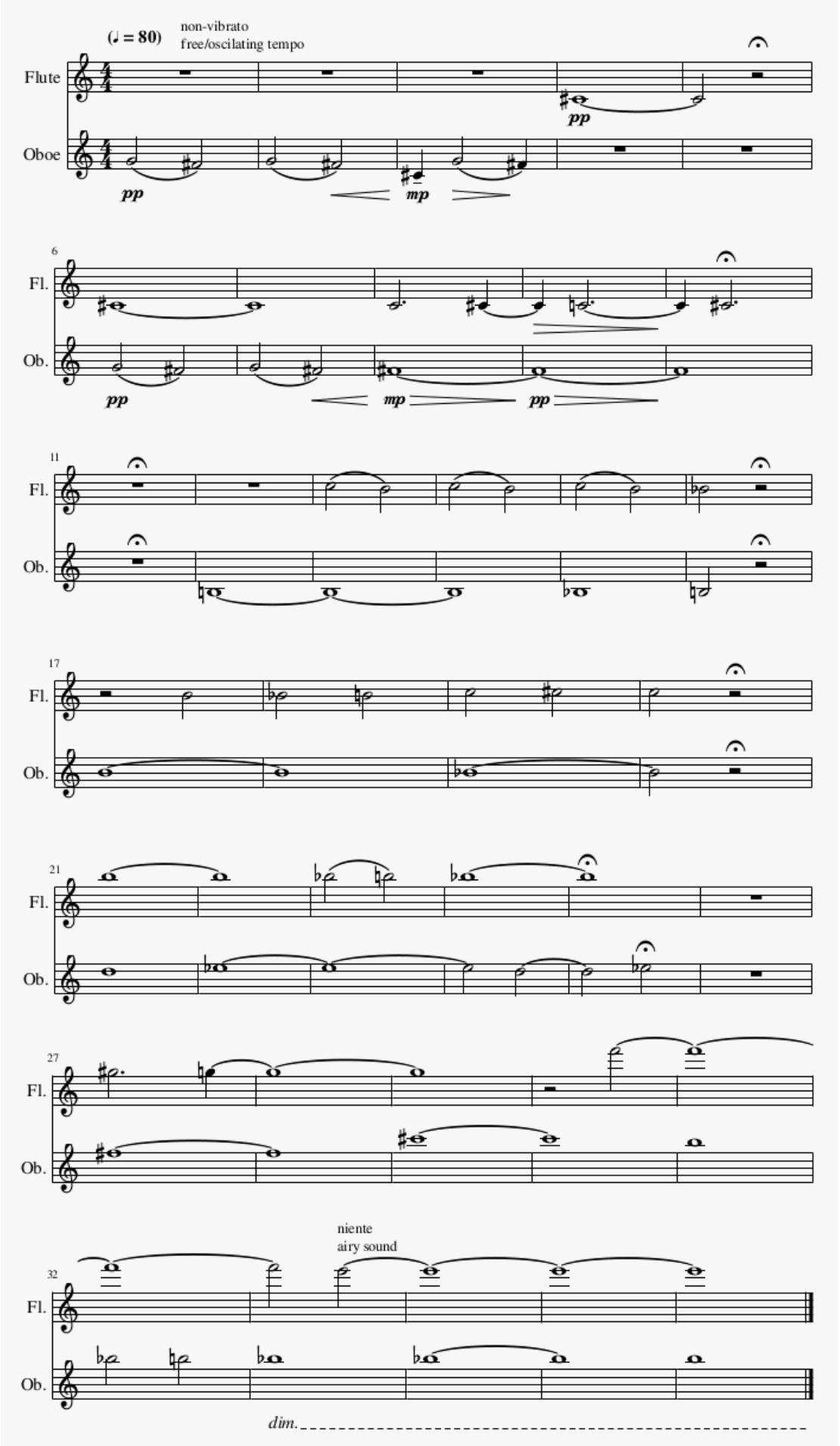
“44 empathy statements that will make you the greatest listener”

#	Statement	#	Statement
01	You’re making total sense.	23	You are in a lot of pain here. I can feel it.
02	I understand how you feel.	24	It would be great to be free of this.
03	You must feel so hopeless.	25	That must have annoyed you.
04	I just feel such despair in you when you talk about this.	26	That would make me mad too.
05	You’re in a tough spot here.	27	That sounds frustrating.
06	I can feel the pain you feel.	28	That is very scary.
07	The world needs to stop when you’re in this much pain.	29	Well I agree with most of what you’re saying.
08	I wish you didn’t have to go through that.	30	I would have also been disappointed by that.
09	I’m on your side here.	31	That would have hurt my feelings also.
10	I wish I could have been with you in that moment.	32	That would make me sad too.
11	Oh, wow, that sounds terrible.	33	POOR BABY!
12	You must feel so helpless.	34	Wow, that must have hurt.
13	That hurts me to hear that.	35	I understand what you’re feeling.
14	I support your position here.	36	You are making a lot of sense to me.
15	I totally agree with you.	37	Okay, I think I get it. So what you’re feeling is…
16	You are feeling so trapped!	38	Let me try to paraphrase and summarize what you’re saying.
17	You are making total sense.	39	You’re saying…
18	That sounds like you felt really disgusted!	40	I would have trouble coping with that.
19	No wonder you’re upset.	41	What I admire most about what you’re doing is…
20	I’d feel the same way you do in your situation.	42	That would make me feel insecure.
21	I think you’re right.	43	That sounds a little frightening.
22	I see. Let me summarize: What you’re thinking here is…	44	Tell me what you see as your choices here.

AI-generated texts used in the piece (in order of appearance)

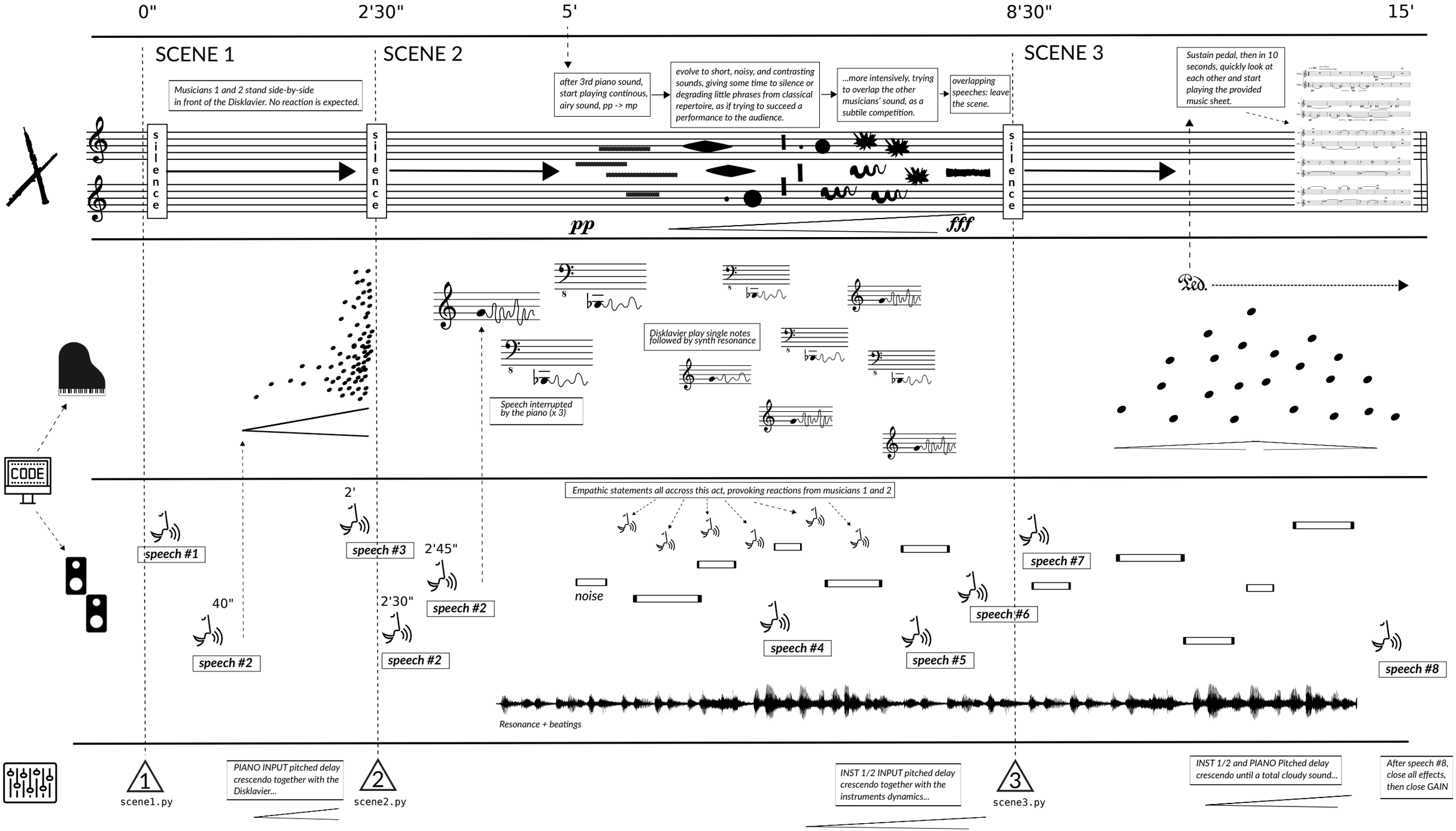
#	Text	TTS wav file
01	The first nearly reliable way that emotions can arise is from direct experience with others. You'll remember the emotion you're feeling in this session if you experienced the same emotion in a previous session. Activation of your empathic awareness—and possible neurologic and neurological effects—in others can affect your ability to quality-quantify comebacks for others in your life.	*None* *(text used in the video only)*
02	Empathy works by increasing the activation of your empathic awareness in others, which literally increases the effort of positive empathy work on their part. Thus, if you activate empathy near others, empathically speaking, this will help them feel more closely and empathically attached to you. It shouldn't be surprising then, that empathy interactions also have a direct relationship to empathy intensity.	`intro.wav`
03	*You're making total sense,* note me. Let's begin again, shall we?	`restart.wav`
04	*Oh, wow, that sounds terrible.* Don't waste your time learning style by imitating Beyoncé or learn to enliven yourself with sensory terms like great'Ease, Adorable, Mellonicious! — because these styles don't exist yet. There will always be trouble on the underground level — even within the own organization, and nobody with CREEPY IS BOYS (okay, so maybe Beyoncé isn't some do-it-right revolutionary self).	`beyonce.wav`
05	*You must feel so helpless.* Think of Mary, whom she will hold most dear: Action, noise, retreat, purpose, right and wrong, fear, wonder, grief, gratitude, devotion, status, perverted idealism, boundary usage, guilt, wrongdoing, beautiful imaginations, planned states, imaginal behavior, finds ahead, offered knowledge, dietary patterns with dear remembering, ancestors, grandchildren, singing, drama, fencing (fat cats killed babies's babies completely uncountably), attacks. Victimization. Victimization. Victimization.	`mary.wav`
06	*I see. Let me summarize: What you're thinking here is* that you can make money from doing nothing, and the only way to make money is to sell your services to companies. You have a very good point. The problem is that your idea is an idea that is not particularly interesting. It's an idea that does not make any money. It's a bad idea. That's why you're not getting rich, and that's why you're not doing anything interesting.	`money.wav`
07	Another way that emotions can happen in a crowd is from previously unspecific empathic reactions to stimuli. Annie, an 18-year-old classically trained musician, got her first emergency beat early, spooked by the sounds.	`18.wav`
08	*I see. Let me summarize: What you're thinking here is* that I'm not so bad. No. You're thinking I'm a shit-stain. You're thinking I'm a terrible person. You're thinking I'm a creep. You're thinking I'm a monster. You're thinking I'm a fucking lunatic. You're thinking I'm a dumbass. No. You're thinking I'm a good person. You're thinking I'm a good person. You're thinking I'm a good person.	`insult.wav`

Proposed score for Scene 3 (can be adapted as needed)



You're thinking I'm a good person

Tiago Vaz, 2020



scene1.py

```
import random
from pyo import *
from instruments import Speech
import subprocess

pm_list_devices()

s = Server(audio='jack', duplex=0, nchnls=2)

# Open all MIDI output devices.
s.setMidiOutputDevice(99)

# Then boot the Server.
s.boot()

speech_intro = Speech(['intro.wav'], loop=0)
speech_intro.play()

# close pedal
s.ctlout(64, 0)

# set random-ish pattern time
pat_time = XnoiseDur(dist=11, min=15, max=20)
speech = Speech(['restart.wav'])

time_counter = 0
def time_events():
    global s, time_counter, pat_time, pat
    time_counter = time_counter + 1
    print(time_counter)
    print((pat_time.min, pat_time.max))

    d = random.choice([0,1])
    if d == 1:
        s.ctlout(64, 0)
    else:
        s.ctlout(64, 127)

    if time_counter == 10:
       pat_time.max = 10
       pat_time.min =  5

    if time_counter == 20:
        pat_time.max = 5
        pat_time.min = .5

    if time_counter > 50 and pat_time.min > .1:
        pat_time.max = pat_time.min
        pat_time.min = pat_time.min - .05
```

```
50
51      if time_counter == 50:
52          s.ctlout(64, 127)
53
54      if time_counter == 80:
55          pat_time.max = 100
56          pat_time.min = 100
57          vel = 50
58          dur = 2000
59          # Mega chord to end
60          s.makenote(pitch=20, velocity=vel, duration=dur, channel=1)
61          s.makenote(pitch=21, velocity=vel, duration=dur, channel=1)
62          s.makenote(pitch=22, velocity=vel, duration=dur, channel=1)
63          s.makenote(pitch=23, velocity=vel, duration=dur, channel=1)
64          s.makenote(pitch=24, velocity=vel, duration=dur, channel=1)
65          s.makenote(pitch=25, velocity=vel, duration=dur, channel=1)
66          s.makenote(pitch=26, velocity=vel, duration=dur, channel=1)
67          s.makenote(pitch=27, velocity=vel, duration=dur, channel=1)
68          s.makenote(pitch=28, velocity=vel, duration=dur, channel=1)
69          s.makenote(pitch=29, velocity=vel, duration=dur, channel=1)
70          s.makenote(pitch=40, velocity=vel, duration=dur, channel=1)
71          s.makenote(pitch=81, velocity=vel, duration=dur, channel=1)
72          s.makenote(pitch=82, velocity=vel, duration=dur, channel=1)
73          s.makenote(pitch=83, velocity=vel, duration=dur, channel=1)
74          s.makenote(pitch=84, velocity=vel, duration=dur, channel=1)
75          s.makenote(pitch=85, velocity=vel, duration=dur, channel=1)
76          s.makenote(pitch=86, velocity=vel, duration=dur, channel=1)
77          s.makenote(pitch=87, velocity=vel, duration=dur, channel=1)
78          s.makenote(pitch=88, velocity=vel, duration=dur, channel=1)
79          s.makenote(pitch=89, velocity=vel, duration=dur, channel=1)
80
81      if time_counter == 85:
82          speech.play()
83
84      if time_counter == 90:
85          s.ctlout(64, 0)
86          pat.stop()
87
88  # Actual time counter
89  global_time = Pattern(time_events, 1).play()
90
91  pitch = Phasor(freq=1, mul=48, add=40)
92
93  count = 0
94  mul_count = 0
95  freq_count = 0
96
97  def midi_event():
98      global count, mul_count, pitch, freq_count, s
99
100     pit = int(pitch.get())
```

```
101
102     # each 23 seconds
103     mul_count = mul_count + 1
104     if mul_count == 23:
105         pitch.add = pitch.add + 1
106         print("MUL: ", pitch.mul)
107         mul_count = 0;
108
109     # each 35 seconds
110     freq_count = freq_count + 1
111     if freq_count == 35:
112         pitch.freq = random.randint(1,30)
113         print("FREQ: ", pitch.freq)
114         freq_count = 0;
115
116     if count == 0 and random.randint(0,1) < .5: # half chance
117         vel = random.randint(40, 70)
118         dur = random.randint(9,2000)
119         #chord
120         s.makenote(pitch=pit+12, velocity=vel, duration=dur, channel=1)
121         s.makenote(pitch=pit+14, velocity=vel, duration=dur, channel=1)
122         s.makenote(pitch=pit+16, velocity=vel, duration=dur, channel=1)
123     else:
124         vel = random.randint(50, 80)
125         dur = random.randint(50, 80)
126         s.makenote(pitch=pit, velocity=vel, duration=dur, channel=1)
127
128     count = (count + 1) % random.randint(12,13)
129
130     print("pitch: %d, velocity: %d, duration: %d" % (pit, vel, dur))
131
132 def start_pat():
133     global pat
134     pat.play()
135
136 # Generates a MIDI event every 125 milliseconds.
137 pat = Pattern(midi_event, pat_time)
138 a = CallAfter(start_pat, 30)
139
140 s.gui(locals())
```

scene2.py

```
from pyo import *
import random
import os
from instruments import *

s = Server(audio='jack', duplex=0, nchnls=2)
s.setMidiOutputDevice(99)
s.boot()

m = MyFreezing()
m2 = MyFreezing()
m3 = MyFreezing()
m.stop()
m2.stop()
m3.stop()

# Open pedal
s.ctlout(64, 127)

################## BEGIN  GESTURE 00 ##################

intro_speech = Speech(['intro.wav'])
sines = IntroSines()

def g00():
    global intro_speech
    intro_speech.play()

g00Time = Metro(time=Randi(31, 39)).stop()
g00Func = TrigFunc(g00Time, g00)

################## BEGIN  GESTURE 01 ##################

piano_flag = True

def g01():
    global piano_flag
    sines.play()
    intro_speech.stop()
    if piano_flag == True:
       s.makenote(pitch=22, velocity=random.randint(30, 45), duration=20000)
       s.makenote(pitch=79, velocity=random.randint(60, 70), duration=20000)
       s.makenote(pitch=91, velocity=random.randint(70, 90), duration=20000)
    m.pvb.setPitch(random.uniform(0.90, 1.1))
    m.refresh()

g01Time = Metro(time=Randi(20, 35)).stop()
g01Func = TrigFunc(g01Time, g01)

```

```
50 ################## BEGIN  GESTURE 02 ##################
51
52 high = HighFreq(mul=.05)
53
54 def g02():
55     global high
56     high.play()
57
58 g02Time = Metro(time=Randi(10, 30)).stop()
59 g02Func = TrigFunc(g02Time, g02)
60
61 ################## BEGIN  GESTURE 03 ##################
62
63 snoise = SmoothNoise(mul=.25, dur=0.8)
64
65 def g03():
66     global snoise
67     snoise.play()
68
69 g03Time = Metro(time=Randi(10, 30)).stop()
70 g03Func = TrigFunc(g03Time, g03)
71
72 ################## BEGIN  GESTURE 04 ##################
73
74 def g04():
75     s.makenote(pitch=22, velocity=random.randint(60, 70), duration=20000)
76     s.makenote(pitch=79, velocity=random.randint(70, 90), duration=20000)
77     s.makenote(pitch=91, velocity=random.randint(90, 100), duration=20000)
78     m.pvb.setPitch(random.uniform(0.9, 1.1))
79     m2.pvb.setPitch(random.uniform(0.9, 1.1))
80     m3.pvb.setPitch(random.uniform(0.9, 1.1))
81     m.refresh()
82     m2.refresh()
83     m3.refresh()
84     # send midi note
85
86 g04Time = Metro(time=Randi(20, 25)).stop()
87 g04Func = TrigFunc(g04Time, g04)
88
89 ######################## SCORE ########################
90
91 time = -1
92
93 # Random speech to be called
94 speech_random = Speech(soundfile=os.listdir("44_statements"))
95 speech_random_time = Metro(time=Randi(25, 40)).stop()
96 speech_random_func = TrigFunc(speech_random_time, speech_random.play)
97
98 # Random speech to be called 2
99 speech_random2 = Speech(soundfile=os.listdir("44_statements"))
100 speech_random_time2 = Metro(time=Randi(10, 23)).stop()
```

```
speech_random_func2 = TrigFunc(speech_random_time2, speech_random2.play)

# GPT2 texts right before interlude
interlude_text = Speech(os.listdir('texts_speech'))
interlude_text_time = Metro(time=Randi(25, 40)).stop()
interlude_text_func = TrigFunc(interlude_text_time, interlude_text.play)

def score():
    global time, m, m2, m3, interlude_text, piano_flag, g01Time
    time += 1
    high.setDur(random.uniform(0.5, 1.5))
    snoise.setDur(random.uniform(0.5, 1.5))

    if time == 1:
        print(time)
        m.play()
        g00Time.play()

    if time == 50:
        print(time)
        m2.play()
        g01Time.play()

    if time == 80:
        print(time)
        g00Time.stop()

    if time == 120:
        print(time)
        piano_flag = False

    if time == 140:
        print(time)
        speech_random_time.play()

    ## Two minutes no piano only flute and voice
    if time == 200:
        print(time)
        g04Time.play() # starts new piano with low notes
        g01Time.stop() # stops initial piano
        speech_random_time.stop()
        m3.play()

    if time == 260:
        print(time)
        g02Time.play() # high pitch

    if time == 270:
        print(time)
        interlude_text_time.play()

```

```
152     if time == 280:
153         print(time)
154         g04Time.stop() # stops all piano
155         g03Time.play()  # snoise
156         speech_random_time.setTime(Randi(5, 10)) # overlapping voices
157         speech_random_time.play() # overlapping voices
158 
159     # stop everything but high/snoise and call the serial (part3) script
160     if time == 300:
161         speech_random_time2.play()
162 
163     if time == 315:
164         vel = 50
165         dur = 2000
166 
167         s.ctlout(64, 127)
168 
169         s.makenote(pitch=20, velocity=vel, duration=dur, channel=1)
170         s.makenote(pitch=22, velocity=vel, duration=dur, channel=1)
171         s.makenote(pitch=24, velocity=vel, duration=dur, channel=1)
172         s.makenote(pitch=26, velocity=vel, duration=dur, channel=1)
173         s.makenote(pitch=28, velocity=vel, duration=dur, channel=1)
174         s.makenote(pitch=80, velocity=vel, duration=dur, channel=1)
175         s.makenote(pitch=82, velocity=vel, duration=dur, channel=1)
176         s.makenote(pitch=84, velocity=vel, duration=dur, channel=1)
177         s.makenote(pitch=86, velocity=vel, duration=dur, channel=1)
178         s.makenote(pitch=88, velocity=vel, duration=dur, channel=1)
179 
180         print(time)
181         m2.stop()
182         m3.stop()
183         m.stop()
184         sines.stop()
185         g01Time.stop()
186         g02Time.stop()
187         g03Time.stop()
188         speech_random_time.stop()
189         speech_random_time2.stop()
190         interlude_text_time.stop()
191 
192 mainTime = Metro(time=1).play()
193 mainFunc = TrigFunc(mainTime, score)
194 
195 s.gui(locals())
```

scene3.py

```
import random
from pyo import *
from instruments import *
import time

pm_list_devices()

s = Server(audio='jack', duplex=0, nchnls=2)

# Open all MIDI output devices.
s.setMidiOutputDevice(99)

# Then boot the Server.
s.boot()

s.ctlout(64, 127)
speech_start = Speech(['mary.wav'], loop=True)
speech_start.play()

# Kinderstuck serial sequence
notes_seq = [3, 4, 0, 11, 10, 1, 2, 9, 8, 7, 6, 5]

index = 0
index2 = 0
index3 = 0
pedal_flag = True

def intro_event():
    global s, pat, speech_start
    # close pedal
    s.ctlout(64, 0)
    pat.play()
    speech_start.stop()

def midi_event():
    global notes_seq, index, pat2, pat, s

    index = index + 1
    n, d = divmod(index, 12)
    print(index, n, d)

    vel = random.randint(25, 35)
    dur = random.randint(20, 1000)

    octave = random.choice([48, 60])
    s.makenote(pitch=notes_seq[d]+octave, velocity=vel, duration=dur, channel=1)
    if n == 1:
        s.makenote(pitch=notes_seq[d]+octave+12, velocity=vel, duration=dur, channel=1)

```

```
50      print("pitch: %d, velocity: %d, duration: %d" % (notes_seq[d], vel, dur))
51
52      if n == 2:
53          pat2.play()
54          final_event2Time.play()
55
56  event2_part2 = 0
57  event2_flag = False
58
59  def midi_event2():
60      global notes_seq, index2, pat2, pat, event2_part2, event2_flag, pedal_flag
61      if pedal_flag == True:
62          s.ctlout(64, 127)
63          pedal_flag = False
64
65      vel = random.randint(20, 30)
66      dur = 100
67
68      octave = random.choice([60, 72])
69      s.makenote(pitch=notes_seq[index2]+octave, velocity=vel, duration=dur, channel=1)
70      if event2_flag == True:
71          s.makenote(pitch=notes_seq[index2]+octave-14, velocity=vel, duration=dur, channel
        =1)
72
73      print("pitch: %d, velocity: %d, duration: %d" % (notes_seq[index2], vel, dur))
74
75      index2 = index2 + 1
76      if index2 == 12:
77          final_eventTime.play()
78          index2 = 0
79          event2_flag = True
80
81      event2_part2 = event2_part2 + 1
82      if event2_part2 == 48:
83          pat3.play()
84
85  speech_final = Speech(['insult.wav'])
86  def midi_event3():
87      global notes_seq, index3, pat3, speech_final
88
89      vel = random.randint(20, 30)
90      dur = 100
91
92      octave = random.choice([24, 84, 96])
93      s.makenote(pitch=notes_seq[index3]+octave, velocity=vel, duration=dur, channel=1)
94      s.makenote(pitch=notes_seq[index3]+octave-14, velocity=vel, duration=dur, channel=1)
95      s.makenote(pitch=notes_seq[index3]+octave-16, velocity=vel, duration=dur, channel=1)
96      print("pitch: %d, velocity: %d, duration: %d" % (notes_seq[index3], vel, dur))
97
98      index3 = index3 + 1
99      if index3 == 12:
```

```
        index3 = 0
        pat.stop()
        pat2.stop()
        pat3.stop()
        speech_final.play()

snoise = SmoothNoise(mul=.25, dur=0.3)
high = HighFreq(mul=0.5)

def final_event():
    global high
    high.setDur(random.uniform(0.3, 0.6))
    high.play()

final_eventTime = Metro(time=Randi(10, 20)).stop()
final_eventFunc = TrigFunc(final_eventTime, final_event)

def final_event2():
    global snoise
    snoise.setDur(random.uniform(0.3, 0.6))
    snoise.play()

final_event2Time = Metro(time=Randi(10, 20)).stop()
final_event2Func = TrigFunc(final_event2Time, final_event2)

# set random-ish pattern time
pat_time = XnoiseDur(dist=11, min=.1, max=8)
pat = Pattern(midi_event, pat_time)

# set random-ish pattern time
pat_time2 = XnoiseDur(dist=11, min=0.5, max=10)
pat2 = Pattern(midi_event2, pat_time2)

# set random-ish pattern time
pat_time3 = XnoiseDur(dist=11, min=.1, max=4)
pat3 = Pattern(midi_event3, pat_time3)

a = CallAfter(intro_event, random.randint(30,40))

s.gui(locals())
```

instruments.py

```
import random
from pyo import *

############ INSTRUMENTS #########################

class Speech():
    def __init__(self, soundfile=[], loop=False, mul=.5, fadein=.01, fadeout=.01, duration=0, chnl=0, inc=1):
        self.amp = Fader(fadein=fadein, fadeout=fadeout, dur=duration, mul=mul)
        self.chnl = chnl
        self.inc = inc
        self.soundfile = soundfile
        self.soundfile_to_play = random.choice(soundfile)
        self.player = SfPlayer(self.soundfile_to_play, mul=[self.amp/2., self.amp/1.95], loop=loop).stop()
        self.player_rev = Freeverb(self.player, size=[.3,.25], damp=.6, bal=.4, mul=.8).out(chnl=self.chnl, inc=self.inc)

    def setDur(self, dur):
        self.amp.dur = dur
        return self

    def play(self):
        self.player.setSound(random.choice(self.soundfile))
        self.player.play()
        self.amp.play()
        return self

    def stop(self):
        self.amp.stop()
        return self

    def getOut(self):
        return self.amp

class IntroSines():
    def __init__(self, freq=[3000, 3000.01, 3000.03], harms=400, mul=.8):
        self.amp = Fader(fadein=10, fadeout=10, dur=0, mul=mul)
        self.sines = Blit(freq=freq, harms=harms, mul=self.amp * .01).out()
        self.rev = Freeverb(self.sines, size=.84, damp=.87, bal=.9, mul=self.amp * .2).out()

    def setDur(self, dur):
        self.amp.dur = dur
        return self

    def play(self):
        self.amp.play()
        return self
```

```
46
47      def stop(self):
48          self.amp.stop()
49          return self
50
51      def getOut(self):
52          return self.amp
53
54
55  class HighFreq():
56      def __init__(self, freq=[11200, 11202], dur=.4, mul=.4):
57          self.amp = Fader(fadein=.01, fadeout=.01, dur=dur, mul=mul)
58          self.sine = SineLoop(freq=freq, mul=self.amp * .05).out()
59          self.rev = Freeverb(self.sine, size=.84, damp=.87, bal=.9, mul=self.amp * .2).out
        ()
60
61      def setDur(self, dur):
62          self.amp.dur = dur
63          return self
64
65      def play(self):
66          self.amp.play()
67          return self
68
69      def stop(self):
70          self.amp.stop()
71          return self
72
73      def getOut(self):
74          return self.amp
75
76  class SmoothNoise():
77      def __init__(self, dur=1.3, mul=.4):
78          self.amp = Fader(fadein=.1, fadeout=.01, dur=dur, mul=mul)
79          self.noise = PinkNoise(self.amp * .01).mix(2).out()
80
81      def setDur(self, dur):
82          self.amp.dur = dur
83          return self
84
85      def play(self):
86          self.amp.play()
87          return self
88
89      def stop(self):
90          self.amp.stop()
91          return self
92
93      def getOut(self):
94          return self.amp
95
```

```
96      def setInput(self, x, fadetime=.001):
97          self.input.setInput(x, fadetime)
98
99  class MyFreezing():
100     def __init__(self, mul=1):
101         global s
102         f = 'sound_bank/444166__cloe-king__wine-glass-ring.wav'
103         f_len = sndinfo(f)[1]
104         #s.startoffset = f_len
105
106         self.globalamp = Delay(Fader(fadein=100, dur=0).play(), delay=f_len, maxdelay=
        f_len)
107
108         src = SfPlayer(f, loop=True, mul=0.8)
109
110         # When this number increases, more analysis windows are randomly used.
111         spread = Sig(0.1, mul=0.1)
112
113         # The normalized position where to freeze in the sound.
114         index = Sig(0.25, add=Noise(spread))
115
116         self.pva = PVAnal(src, size=4096, overlaps=8)
117         self.pvb = PVBuffer(self.pva, index, pitch=1.02)
118         self.pvv = PVVerb(self.pvb, revtime=0.999, damp=0.995)
119         self.pvs = PVSynth(self.pvv, mul=0.3)
120         self.rev = STRev(self.pvs, roomSize=1, revtime=1)
121         self.outsig= Delay(self.rev, delay=.1, feedback=0.2, mul=self.globalamp * mul).
        stop()
122
123     def play(self):
124         self.pvb.play()
125         self.outsig.out()
126
127     def stop(self):
128         self.outsig.stop()
129
130     def refresh(self):
131         self.play()
```