Empathy Swarm

Reward and Punish | Programming workshop
by Katrin Hochschuh and Adam Donovan

Requirements:

- Processing:
  - Version 3.5.4: Download here: https://github.com/processing/processing/releases?after=processing-0006
  - Libraries needed:
    - oscP5 0.9.9 by Andreas Schlegel: An Open Sound Control (OSC) implementation
    - To add libraries in Processing: click on Sketch>Import Library...>Add Library... then the Contribution Manager will open. Type the library name in the search field (oscP5), click on the result and click on the Install button.
  - To validate that you are ready, check if the following examples work for you:
    - Processing > File > Examples ... > Topics > Simulate > Flocking
      - Press play and a window with a moving swarm opens.
    - Processing > File > Examples ... > Contributed Libraries > oscP5 > oscP5sendReceive
      - Press play, a window opens, then click into that window. In the console at the bottom of processing should be written:

        ### received an osc message. addrpattern: /test typetag: i

- Python:

  - Download and install Anaconda from here:
    https://www.anaconda.com/products/distribution

  - Open the Anaconda Prompt (it's a command-line interpreter and looks like a black window with text)

  - Afterwards create a virtual environment with the needed Python version by typing:
    conda create -n "ESrl" python=3.8

  - Activate it:
    conda activate "Esrl"

  - Install the required dependencies:

    - python-osc (tested with version: 1.7.7)
    - numpy (tested with version 1.20.3)
    - pickle (tested with version 4.0)
    - matplotlib (tested with version 3.4.2)

  - For that type in the shell:
    pip install python-osc==1.7.7
    pip install numpy==1.20.3
    pip install pickle-mixin
    pip install matplotlib==3.4.2

    (These are the commands under Windows 10. If this does not work with your operating system, type the commands and your os in a search engine and follow the instructions.)

  - Download PyCharm from here:
    https://www.jetbrains.com/pycharm/

  - In order to make the newly configures Python interpreter visible for all future projects:
    - Click on File > Settings > Project > Python Interpreter
    - In the dropdown menu of Python Interpreter, click on > All
    - In the new window of Python Interpreters, press +, the Add Python Interpreter window opens.
    - Click on Virtualenv Environment, select Existing environment, press on the three dots … and navigate to your anaconda3\envs\ESrl folder and select python.exe (e.g. in Windows: C:\Users\Username\anaconda3\envs\ESrl\python.exe)
    - Select Make available to all projects, then press OK

- To validate that you are ready, check if the following works for you:
  - Click on File > Create Project
    - Choose the Python Interpreter
      - Select: Previously configured interpreter
      - in the dropdown menu of the Interpreter, you should see something like this (in Windows 10):

        Python 3.8 C:\Users\Username\anaconda3\envs\ESrl\python.exe

        It's just important that anaconda and the name of the environment we created before (ESrl) are in folder structure of the python interpreter as we installed the necessary dependencies in that environment.
  - When you click on open, the Python sample script opens. When you press play (top right), you can read „Hi, PyCharm" in the console at the bottom.
  - Delete all the code and type instead:
    import pythonosc
    print(help(pythonosc))
    - When you press play and „Help on package pythonosc:" etc. appears in the console without any red text! then you are ready to go.
  - Do the same for pickle, numpy, matplotlib
    import pickle
    print(help(pickle))
    etc.
- You can also try if you manage to get OSC communication between Processing and Python up and running.
  For that you could use the Processing example oscP5sendReceive again and copy the Simple client code from https://pypi.org/project/python-osc/ into your Python program, you just need to change the default port to the one Processing is listening to.
- Alternatively you can also use the Python Simple server code, and e.g. adapt the Processing example to the matching port for the remote location and the matching filter e.g. „/filter" for the OSCMessage.