DT2300 Sound in Interaction Lab 1, 2, & 3

This document contains all instructions needed for the three laboratory sessions in the course Sound In Interaction. The three sessions are divided into two types of tasks, a less formal introduction to pd in session 1, followed by sets of more structured tasks for sessions 2 and 3. In all three sessions, the main software used is Pure Data (pd). Pure Data is an open source, cross platform, data flow programming environment that is geared towards real time audio. In sessions 2 and 3, you will also use the Motion Capture system in the lab as a source for control data, creating musical instruments and musical games. To get a passing grade on the laboratory module in the course, you must complete the tasks detailed in this document.

- The lab is equipped with computers, but if you prefer it you can use your own laptop. The software, Pure Data (pd), runs on MacOS, Windows, and Linux.
- To log in to the lab computers, use the user name ”op_csc-studio” and the password ”stud101Lab”.
- Create a folder and save all your files to one place. When the lab session is done, back up your files and then delete them from the lab computer.
- Save often and save to different files, maintaining a history of your work.
- If you open files in pd by double clicking them in Explorer a new instance of pd will be created which might lead to confusing results. Use the file-menu in pd to avoid this.
- The integrated help in pd contains enough information for you to be able to solve all the tasks below. Right click on an object and select ”Help” to get more info on that object

Online Material

Some excellent resources that can be used both as references as well as sources of inspiration are:

- Pure Data Main Portal
  `http://puredata.info/`
- Pure Data FAQ
  `http://puredata.info/docs/faq/`
- Pure Data FLOSS Manual
  `http://en.flossmanuals.net/PureData/`
- Programming Electronic Music in Pd
  `http://www.pd-tutorial.com/english/index.html`
- Mobile Music Platform (creating mobile applications with pd)
  `http://danieliglesia.com/mobmuplat/`
- The Pure Data Vimeo Group
  `https://vimeo.com/groups/puredata`
- The Pure Data YouTube Channel
  `https://www.youtube.com/user/PureDataVideopedia`
- The Pure Data Patches of Martin Brinkmann
  `http://www.martin-brinkmann.de/pd-patches.html`
- The Generative Music and Procedural Sound Design of Sim Cel
  (An example of using pd to create music for a game)
  `https://www.youtube.com/watch?v=0xr4aL1C24E`
- Critter & Guitari - Organelle
  (Hardware synth that runs pd)
  `https://www.youtube.com/watch?v=TW6FAxLFzLk`

- MONOLOG X
  (A great and free pd synthesizer patch)
  `http://www.monologx.com/ecosystem/`

Lab session 1

In the first laboratory session, you will familiarize yourselves with the software *pure data*, often abbreviated as *pd*. If you have not used pd before, you will have plenty of time to get familiar with the very basic operations in session 1. If you have used pd or similar environments before you will progress through the material quicker and can either cut the first session short, or start to work on the material for session 2. However, please note that the purpose of this lab session is for you to calmly and thoroughly get to know pd, so take your time, take notes of key points, save all your work for future reference, read the help file for any object you don't understand, and ask your lab session supervisor if anything is unclear. The instructions for the lab session are as follows:

Set up your workstation

1. Log in to the computer, using the user name `op_csc-studio` and the password `stud101Lab`.
2. Create a local temporary folder for all your files. This folder is to be deleted at the end of the session, after you have copied all your files to your kth account or some other personal storage.

Setup your audio output and test it in pd

1. Check that your computer is equipped with an external USB sound card
2. Check that your sound card has two pairs of headphones connected, either to one output using a splitter, or to two separate outputs.
3. Verify that you can listen to the computer's audio output through the headphones, by playing a YouTube video.
4. Take note of the listening volume and make sure that you know how to adjust it, this might become crucial later in the lab sessions.

5. Open pd and try the connection to the sound card by first selecting the sound card as an output device in `Audio Settings...` and then testing it using `Test Audio and MIDI...`, both found in the `Media` menu in pd. Mind your ears!

Get started with pd

1. Start by going through the first 3 of Dr. Rafael Hernandez's excellent video tutorials on pure data: `https://www.youtube.com/playlist?list=PL12DC9A161D8DC5DC`, and pause between each video to look at the help files for the elements used in the video. Recreate the patches that are used in the videos yourselves. Make sure you understand the concepts presented in the videos. Try to modify each patch to do something slightly different, like printing another message or performing another calculation. Save your progress in separate files as you go.

2. Make sure that you can answer the following questions correctly:

   - What does "bang" mean in the pd context?
   - What is "the canvas"?
   - What is the difference between an object and a message?
   - How are "hot" and "cold" objects different?
   - How do you work with the trigger object? What potential issues does it solve and when would you use it?
   - In what type of situations might you be at risk of creating a stack overflow?

3. Consider the following questions and come up with a brief answer to each of them.

   - Data Flow is different in some ways to programming with code written with text, can you give some examples of that?

- What would you consider the visual metaphor of boxes and wires to be particularly suited for?
-

4. Check the MIDI configuration by first selecting an output device in `MIDI Settings...` and then testing it using `Test Audio and MIDI...`, both found in the `Media` menu in pd. Mind your ears!

5. Watch Dr. Rafael Hernandez's 6th video, at `https://www.youtube.com/watch?v=nTTZZyD4xlE` and his 8th video, found at `https://www.youtube.com/watch?v=ojO6woTngG8`. Like above, try and replicate one of the patches while watching, pause when you need to. Look at the help files for any objects you haven't encountered before. Make a version of the patch that is different in some way but still functions. Save your new version.

6. Alert the lab session supervisor to your progress and take a few minutes to show your patches and discuss any questions you might have on pd so far.

Pure Data in a larger context

1. Watch the following videos that show real world use cases for pd:

   - What is MobMuPlat?
     `https://vimeo.com/85295522`
   - The Generative Music and Procedural Sound Design of Sim Cel
     `https://www.youtube.com/watch?v=0xr4aL1C24E`
   - Critter & Guitari - Organelle
     `https://www.youtube.com/watch?v=TW6FAxLFzLk`

2. Try and identify some of of Pure Data's strengths and weaknesses. How could pd be of use in your course projects? When would something else be a better tool for the job? What makes pd unique and why

do you think so many people use it for so many things? Take notes of your conclusions.

3. Sum up your discussions on the use of pd for your lab supervisor.

Your first patch from scratch

1. Design a small, well defined, task for yourselves, like performing a particular calculation, counting to a certain number, creating a button that makes pd print 5 words with a 2 second delay between each word, or something else that you would like to explore.

2. Describe the task to the lab supervisor to make sure that it is appropriate.

3. Construct the patch and make sure that it actually does what you set out to do.

4. Show your work and sum up your discussions on pd for your lab supervisor. Backup all your files and notes and remove them from the lab computer.

Lab session 2 & 3

In the second and third laboratory session, you will tackle three levels of increasing complexity. In the second laboratory session you should aim to finish level 1 and level 2. In the third session, you have to finish level 3. Each level contains several tasks for you to choose from. You finish a task by presenting your finished results to the lab session supervisor. If you haven't finished level 1 and 2 when the second laboratory session has ended, you have to finish them by yourself, at home, before the next session so that they can be presented at the start of the third and last lab session. To get a passing grade, level 3 must be finished by the end of the third lab session.

Read the instructions for each task carefully and ask questions if something is unclear. Each task has a list of suggested objects that will help you solve the task. Look at the help files for the suggested objects and make sure you understand how they work. Make sure that your solution fulfills all the requirements in the instructions before you present your work. Begin by setting up your workstation in the same way as in session 1.

Level 1: Getting Started

Some simple exercises that cover the basic pd objects that you need to be familiar with to solve the tasks in level 1 & 2. You must finish all tasks before advancing to the next level.

Bang by comparison

Create a variable number ”A” (e.g. a slider or a number-box). Compare that number to another number ”B”. The comparison should output 1 when ”A” is bigger then ”B” and 0 otherwise. Bang once when the comparison changes from 0 to 1. Important objects to explore are VSlider, Number, Bang, [<], [select], [change].

Combine comparisons

Create a variable number ”A” (e.g. a slider or a number-box). Compare that number to two other numbers ”B” and ”C”. The comparison should output 1 when ”B” $<$ ”A” $<$ ”C” and 0 otherwise. Bang once when the comparison changes from 0 to 1. Important new objects to explore are [$*$].

Beep

Create a toggle button that controls the amplitude of a sinus oscillator. When the toggle is active, the sound from the oscillator should be heard. When the toggle is inactive, there should be no sound. The oscillator should have a frequency of 900 Hz. Important new objects to explore are Toggle, [$*$~], [cycle~].

Beep by comparisons

Combine a comparison like the one in task 2 (”B” $<$ ”A” $<$ ”C”) with the beep from task 3 so that the sound is heard only when the comparison is true.

Map to a scale

Create a sawtooth oscillator and two variable numbers. Use the variable numbers to control the amplitude and pitch of the oscillator. Limit the variable numbers so that the amplitude control can vary between 0 and 1, while the pitch control varies between the C note at 130.813 Hz (MIDI note nr 48) and the C note at 523.251 Hz (MIDI note nr 72). In addition, the pitch control should change in steps of 2 semitones, creating a whole note scale. Important new objects to explore are [int], [mtof], [phasor~].

Level 2: Musical Instruments

Create real time musical interaction by mapping the data from the MoCap system to synthesised sounds in pure data. Use the provided patch for a reference on how to receive data from the MoCap system in pd. Select two tasks and solve them to move on to the next level.

Theremin

Create a Theremin-like instrument by controlling a sound using the position of your trackable object. Map the x, y, and z position of the object to the amplitude, pitch and timbre of the sound, respectively. Make sure that the possible values for the amplitude and pitch are reasonable and create an interaction where you can control the sound reliably enough to play a simple melody. Smooth the incoming MoCap data to reduce uncontrolled variations in the sound. Create the sound by generating a sawtooth wave and modify the timbre with a low pass filter. Important objects to explore are [mavg], [lop~].

Drum set

Create a virtual drum set by defining a set of 3 squares in the MoCap space. Detect when your object hits a square and play a sound. A square is considered hit when the object passes through it in one direction (but not the other). The sound should be played only once when the object hits the square. Connect the different squares to different sounds. One sound should consist of band pass filtered noise, emulating a snare drum. Another sound, emulating a hi-hat sound, should consist of three sawtooth waves that are multiplied with each other and sent through a high pass filter. Choose whatever method you like to produce the third sound. The sounds should be perceived as equally loud. Important objects to explore are [noise~], [bp~], [hip~], [line~].

Velocity sensitive piano

Create a virtual piano by defining a keyboard rectangle in the MoCap space. Detect when your object hits the rectangle and play a piano sound. The rectangle is considered hit when the object passes through it in one direction (but not the other). The sound should be played only once when the object hits the rectangle. Depending on where the object passes through the rectangle, the resulting sound should have a different pitch, just like how each note on the piano has a different pitch. Make sure that the pitches correspond to the notes available on a piano. The sound should be generated by playing back a sample of a piano note. A sound file containing a piano note will be provided in the lab files. To achieve the difference in pitch, the playback-speed should be varied. The amplitude of the sound should correspond to the speed of the trackable object when it hits the keyboard rectangle. Important objects to explore are [tabread4], [array], [line].

Your own idea

If you have an idea of your own for a musical instrument, talk with the lab session supervisor and make sure that it is practically possible and of suitable complexity.

Level 3: Games

Create playful interaction using interactive real time audio production. Solve at least one task to complete the lab session.

Head, Shoulders, Knees and Toes

Create a periodic rhythm and play a sound on each beat. Measure the height of a player's head, shoulder, knees and toes. Create a game where the player is to move the trackable object so that it is equal in height to the measured heights, on time with the beat, in a predefined order. Play a sound every time a correct height is measured and another sound if an incorrect height

is measured. Reset the game if the player gets it wrong, i.e., an incorrect measurement is detected at the time of the beat.

Seek and Smash game

Choose a random point in the capture space. Guide the player to the point using interactive sound. You can use whatever sounds you like. You should at least map the distance to the point to the sound, but if you also use the direction to the point it will be more effective. When the player moves the trackable object close enough to the point you should play a sound and move the point to a new random location in the capture space.

Your own idea

If you have an idea of your own for a game, talk with the lab session supervisor and make sure that it is practically possible and of suitable complexity.