5. Audio

In this lab, we need some special library for audio processing. In processing, we will use the Minim and Sound libraries which are software library written in Java for realtime audio programming. To install Minim: open the contribution manager in Processing (Sketch -> Import Library -> Add Library); Search 'Minim'; Click 'Install'. To install Sound: open the contribution manager in Processing (Sketch -> Import Library -> Add Library); Search 'Sound'; Click 'Install'. In python, we will use pygame and pydub. To install these two libraries, please refer to the official guide. Pydub: https://github.com/jiaaro/pydub, pygame: https://www.pygame.org/download.shtml and https://www.pygame.org/wiki/Compilation

5.1 Play Sound File

5.1.1 Processing

You can use the two packages we introduced previously to play with audio clips. Try to add some basic operation with mouse button function. If you want to know more about the mouse function in processing, please refer to : https://processing.org/reference/mouseButton.html.

Task1: Mouse clicking to play the sound

There are three buttons in your mouse (left, middle and right). In this task, you need to add play, pause and restart functions on the mouse buttons. Here we give you a basic structure for playing an audio, please add your own code in the mouseClicked function.

```
import ddf.minim.*;  //import the library

Minim minim;
AudioPlayer song;

void setup()
{
```

```
  size(500, 500);

  minim = new Minim(this);  //initialization

  // this loads .wav file from the data folder
  song = minim.loadFile("drum.wav");
  song.play();
  //song.loop();  //if you want to make it loop
}

void draw()
{


  background(0);

}

void mouseClicked() {

  // Enter your code here
}
```

Task2: Control the volume with the mouse position

In this section, we need to change the volume of the audio by clicking positions in the window. Here we use another package for audio processing.

```
import processing.sound.*; //another package for audio processing
SoundFile song;

void setup() {
  size(500, 500);


  // Load a soundfile from the /data folder of the sketch and play it back
  song = new SoundFile(this, "sample-section1-2.wav");
  song.play();
  song.loop();
}

void draw() {

   background(0);

}
```

```
void mouseClicked() {

  // Enter your code here
  // Hint: The volume of an audio is just its amplitude

}
```

5.1.2 Python

In python, both pygame and pydub are able to play audio file. In pygame, an audio can be displayed with the following code:

```
import pygame

pygame.init()  # Before using pygame, you must initialize it
pygame.mixer.init()
screen = pygame.display.set_mode((400, 300))
# Set a screen for pygame. Only by clicking your mouse in this window
# can make the GUI functions work.


pygame.mixer.music.load("drum.wav")

# load a sound into a mixer variable

pygame.mixer.music.play(0) # play the sound
time.sleep(1)
# Why we need this line? Just remove it and see what will happen.

# Also try to change the number in sleep().
```

Pygame also contains many functions connected with mouse, which means you can manipulate the audio file by clicking or moving the mouse. However, pydub does not have such GUI functions, you have to install other GUI libraries. In pydub, you can play an audio in this way:

```
from pydub import AudioSegment
from pydub.playback import play

song = AudioSegment.from_wav("sample-section1-2.wav")

play(song)
# No need to initialize anything compared with pygame
```

Task1: Mouse clicking to play the sound

In this task, you need to play, pause and continue a sound file by clicking the mouse buttons. Read the code and hints, enter your own code to finish the task.

```
import pygame

pygame.init()  # Before using pygame, you must initialize it
pygame.mixer.init()
screen = pygame.display.set_mode((400, 300))
# Set a screen for pygame

pygame.mixer.music.load("drum.wav")

#Always use this kind of while loop and for loop
# when using some GUI functions in pygame

done = False

while not done:
    for event in pygame.event.get():
    # get all the event from pygame
        if event.type == pygame.QUIT:
            done = True

        # Three buttons, middle to play music, left to pause,
        #right to unpause. Enter your code here

     pygame.display.flip()


# Hints:
# 1 'pygame.MOUSEBUTTONDOWN' is a type of event for clicking the button
# 2 There are three buttons on your mouse, when you click each button,
# a variable named 'event.button' will return different values (0,1,2)
# 3 To pause and unpause an audio, use functions pygame.mixer.music.pause()
# and pygame.mixer.music.unpause() respectively.
```

Task2: Control the volume with the mouse position

In this task, we need to change the volume of an audio while playing by clicking different positions in the pygame window. You can change the volume when the mouse click a different x position or y position.

```
import pygame

pygame.init()  # Before using pygame, you must initialize it
```

```
pygame.mixer.init()
screen = pygame.display.set_mode((400, 300))
# Set a screen for pygame

pygame.mixer.music.load("drum.wav")

#Always use this kind of while loop and for loop
# when using some GUI functions in pygame

done = False


while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
        # Still use three buttons on your mouse. Middle button to
        # play the sound, left button to change the position and the
        # volume and right button to stop.
        # Enter your code here


    pygame.display.flip()

# Hints:
# 1 'pygame.mouse.get_pos()' will get the current position of your mouse (x,y).
# 2 'pygame.mixer.music.set_volume(rate)' is used to change the volume of an
# audio. Rate is a float in range (0,1). After using this function, the
# current volume is previous_volume * rate. In this way, you cannot enlarge
# the volume with this function.
# 3 You may need to use the pause and unpause functions introduced in the
# previous task.
```

5.2 Sound Filtering

5.2.1 Processing

To apply a low-pass filter on the sound clip, we can declare a sound filter as below.

```
OnePoleFilter filter1; // this is our filter unit generator
// set up our new filter with a cutoff frequency of 200Hz
// We only keep the sound with frequency lower than 200Hz
filter1 = new OnePoleFilter(ac, 200.0);
// connect the SamplePlayer to the filter
filter1.addInput(sp); // The same sound sampler we defined in last section
```

Try to apply this low-pass filter everytime before the sound sample is played. Try to compare the difference between different frequencies.

5.2.2 Python

In python, pydub contains both lowpass and high pass filters. Just run the code, and compare different effects.

```
from pydub import AudioSegment
from pydub.playback import play
from pydub.scipy_effects import low_pass_filter, high_pass_filter

song = AudioSegment.from_wav("sample-section1-2.wav")
play(song)

low_pass = low_pass_filter(song, 300)

play(low_pass)

high_pass = high_pass_filter(song,300)

play(high_pass)
```