# Section B
# Unit #5
# FFT

# Content

## Unit #5 FFT

# Introduction

This section explores sound and music. For this you will need a microphone, most machines have them built in. You will be creating and graphing sound using the microphone and downloading free music, and from this you can create visualisation of the sounds and/or music.

This brings in two very useful topics that have many applications. If nothing else it is fun and interesting.

For where we just using sounds you will need a built in microphone (most computers have a built in one, or if you are attaching a separate USB webcam they often have the microphone built in. If you can I would recommend getting a good quality microphone that you can add yourself (usually though a USB port). It just gives a better sound input but is not essential.

Here also you can create your own sounds and I guess you can create your own music if that is your thing. I have seen some examples where you can create a simple keyboard once you know the frequency of the different notes.

# FFT

## Unit #5 Fast Fourier Transform

### Introduction

This unit looks at using the Fast Fourier Transform (FFT) to analyse the sound waveform information. This may seem like a mouthful and it is.

# Sketch B5.1 FFT (part 1)

We will be using the same sound file

```
let song
let fft
let spectrum

function preload()
{
  song = loadSound('sounds/music.mp3')
}

function setup()
{
  createCanvas(400, 400)
  song.play()
  fft = new p5.FFT()
}

function draw()
{
  background(220)
  spectrum = fft.analyze()
  console.log(spectrum.length)
}
```

## Notes

This gives an array of length 1024, that means 1024 frequency bands

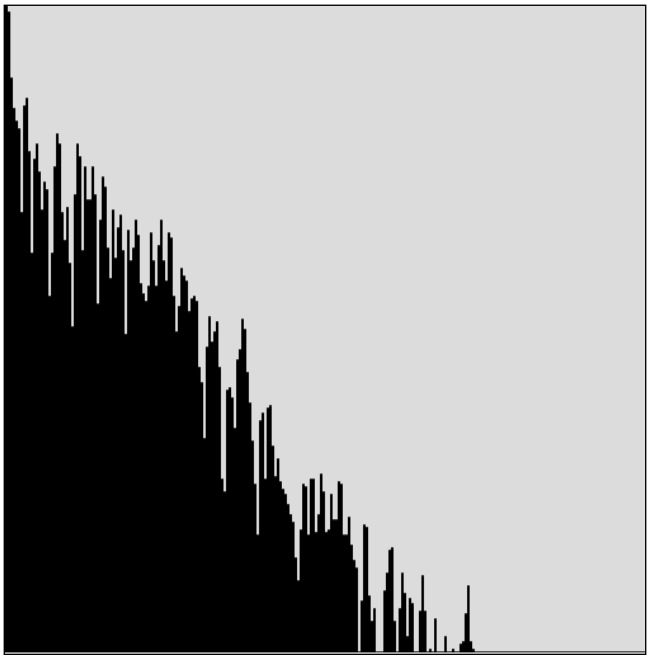
# FFT (part 2)

The **p5.FFT()** function takes two arguments, the first is smoothing between 0-1 and the second is the number of bands (a function of 1024). We are going to loop through the array and draw each frequency band of one pixel width (hence changing the canvas size)

```
let song
let fft
let spectrum
let amp
let y

function preload()
{
  song = loadSound('sounds/music.mp3')
}

function setup()
{
  createCanvas(256, 256)
  song.play()
  fft = new p5.FFT(0, 256)
}

function draw()
{
  background(220)
  spectrum = fft.analyze()
  for (let i = 0; i < spectrum.length; i++)
  {
    amp = spectrum[i]
    y = map(amp, 0, 256, height, 0)
```

```
    line(i, height, i, y)
  }
}
```

This is the result



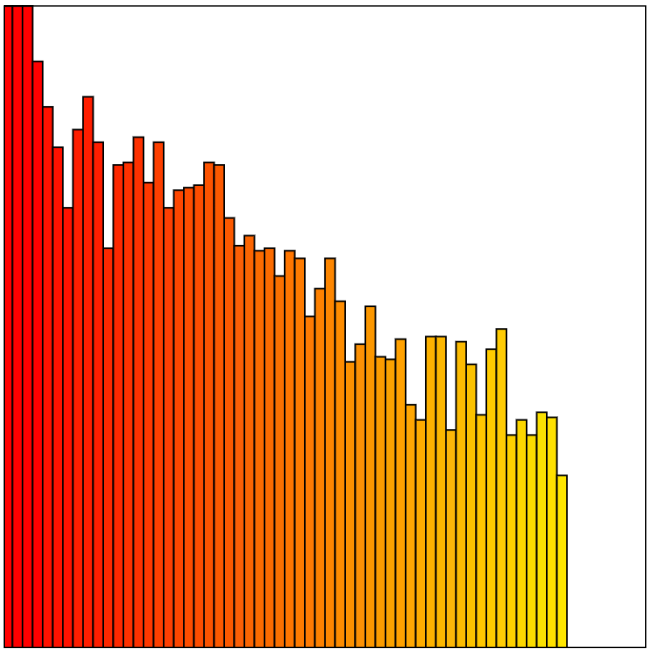
# FFT (part 3)

Adding colour, bars and generally improving appearance

```
let song
let fft
let spectrum
let amp
let y
let w

function preload()
{
  song = loadSound('sounds/music.mp3')
}

function setup()
{
  createCanvas(400, 400)
  song.play()
  fft = new p5.FFT(0, 64)
  w = width/64
  colorMode(HSB)
}

function draw()
{
  background(220)
  spectrum = fft.analyze()
  for (let i = 0; i < spectrum.length; i++)
  {
    let amp = spectrum[i]
```

```
    let y = map(amp, 0, 256, height, 0)
    fill(i, 255, 255)
    rect(i * w, y, w, height - y)
  }
}
```

Using HSB colour mode



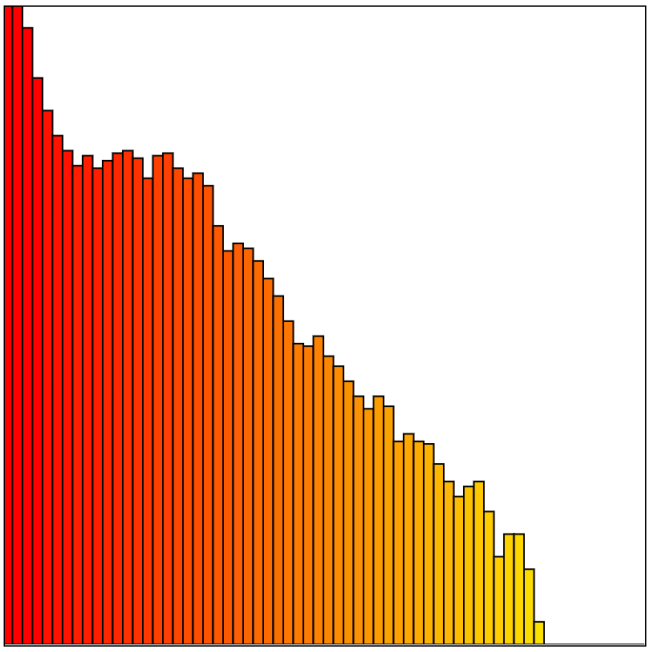
# FFT (part 4)

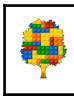
Adding **smoothing** has a very pleasing effect

```
let song
let fft
let spectrum
let amp
let y
let w

function preload()
{
  song = loadSound('sounds/music.mp3')
}

function setup()
{
  createCanvas(400, 400)
  song.play()
  fft = new p5.FFT(0.9, 64)
  w = width/64
  colorMode(HSB)
}

function draw()
{
  background(0)
  spectrum = fft.analyze()
  for (let i = 0; i < spectrum.length; i++)
  {
    amp = spectrum[i]
```

```
    y = map(amp, 0, 256, height, 0)
    fill(i, 255, 255)
    rect(i*w, y, w, height - y)
  }
}
```

With smoothing





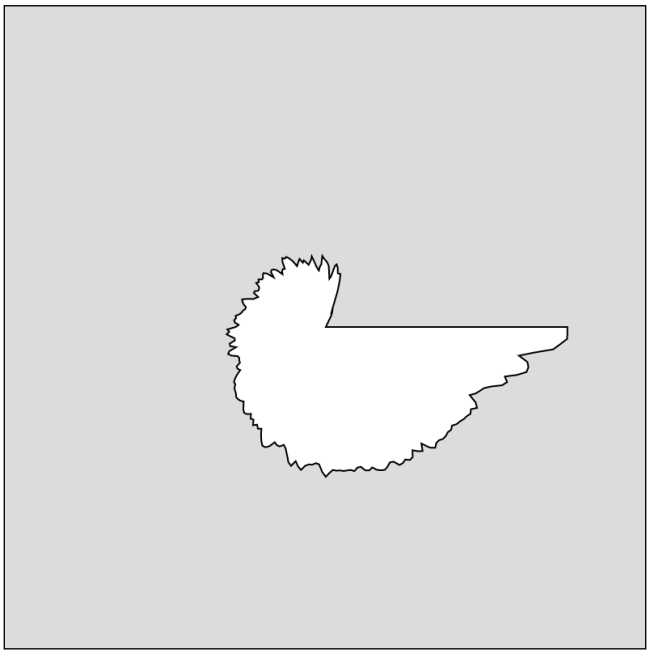
# FFT (part 5)

Creating a circular graph

```
let song
let fft
let spectrum
let amp
let y
let angle
let r
let x

function preload()
{
  song = loadSound('sounds/music.mp3')
}

function setup()
{
  createCanvas(400, 400)
  song.play()
  fft = new p5.FFT(0.9, 256)
  angleMode(DEGREES)
}

function draw()
{
  background(220)
  fill(255)
  stroke(0)
  spectrum = fft.analyze()
```

```
  translate(width/2, height/2)
  beginShape()
  for (let i = 0; i < spectrum.length; i++)
  {
    angle = map(i, 0, spectrum.length, 0, 360)
    amp = spectrum[i]
    r = map(amp, 0, 256, 0, 150)
    x = r * cos(angle)
    y = r * sin(angle)
    vertex(x, y)
  }
  endShape(CLOSE)
}
```

This is the effect



# FFT (part 6)

Adding a line and some colour

```
let song
let fft
let spectrum
let amp
let y
let angle
let r
let x

function preload()
{
  song = loadSound('sounds/music.mp3')
}

function setup()
{
  createCanvas(400, 400)
  song.play()
  fft = new p5.FFT(0.9, 256)
  angleMode(DEGREES)
  colorMode(HSB)
}

function draw()
{
  background(0)
  strokeWeight(2)
  spectrum = fft.analyze()
```

```
  translate(width/2, height/2)
  for (let i = 0; i < spectrum.length; i++)
  {
    angle = map(i, 0, spectrum.length, 0, 360)
    amp = spectrum[i]
    r = map(amp, 0, 256, 0, 200)
    x = r * cos(angle)
    y = r * sin(angle)
    stroke(i, 255, 255)
    line(0, 0, x, y)
  }
}
```

Something like this