EECS 1710
Programming for Digital Media

Practical Session P8:: Working with Audio

YORK U
UNIVERSITÉ
UNIVERSITY

Digression – more objects & basic file IO

YORK UNIVERSITÉ UNIVERSITY

Some other useful reference types:

Data

Composite

`Array`	An array is a list of data
`ArrayList`	An `ArrayList` stores a variable number of objects
`FloatDict`	A simple table class to use a `String` as a lookup for a float value
`FloatList`	Helper class for a list of floats
`HashMap`	A `HashMap` stores a collection of objects, each referenced by a key
`IntDict`	A simple class to use a `String` as a lookup for an int value
`IntList`	Helper class for a list of ints
`JSONArray`	A JSONArray is an ordered sequence of values
`JSONObject`	A `JSONObject` is an unordered collection of name/value pairs
`Object`	Objects are instances of classes
`String`	A string is a sequence of characters
`StringDict`	A simple class to use a `String` as a lookup for an `String` value
`StringList`	Helper class for a list of Strings
`Table`	Generic class for handling tabular data, typically from a CSV, TSV, or other sort of spreadsheet file
`TableRow`	Represents a single row of data values, stored in columns, from a `Table`
`XML`	This is the base class used for the Processing XML library, representing a single node of an `XML` tree

ArrayLists

- Like arrays, but a lot more convenient!
  - keeps elements in a sequence (like arrays) but can grow
  - includes methods to add, sort, find max, reverse, shuffle etc…

- IntList → dynamic/resizable array of ints
- FloatList → dynamic/resizable array of floats
- StringList → dynamic/resizable array of Strings

- ArrayList → use if you want a list of any type of object
  (e.g. like an ArrayList of PVector)

StringList

In fact most methods common to all ArrayLists

Methods	
`size()`	Get the length of the list
`clear()`	Remove all entries from the list
`get()`	Get an entry at a particular index
`set()`	Set an entry at a particular index
`remove()`	Remove an element from the specified index
`append()`	Add a new entry to the list
`hasValue()`	Check if a value is a part of the list
`sort()`	Sorts the array in place
`sortReverse()`	A sort in reverse
`reverse()`	Reverse the order of the list
`shuffle()`	Randomize the order of the list elements
`lower()`	Make the entire list lower case
`upper()`	Make the entire list upper case
`array()`	Create a new array with a copy of all the values

IntList

In fact most methods common to all ArrayLists

Methods	
`size()`	Get the length of the list
`clear()`	Remove all entries from the list
`get()`	Get an entry at a particular index
`set()`	Set the entry at a particular index
`remove()`	Remove an element from the specified index
`append()`	Add a new entry to the list
`hasValue()`	Check if a number is a part of the list
`increment()`	Add one to a value
`add()`	Add to a value
`sub()`	Subtract from a value
`mult()`	Multiply a value
`div()`	Divide a value
`min()`	Return the smallest value
`max()`	Return the largest value
`sort()`	Sorts the array, lowest to highest
`sortReverse()`	Reverse sort, orders values from highest to lowest
`reverse()`	Reverse the order of the list elements
`shuffle()`	Randomize the order of the list elements
`array()`	Create a new array with a copy of all the values

```java
final int MAX_ITEMS = 10;
String [] inventory = new String[MAX_ITEMS];
int numItems = 0;


inventory[numItems++] = “banana”;
inventory[numItems++] = “stick”;
inventory[numItems++] = “BFG”;
inventory[numItems++] = “abomb”;
inventory[numItems++] = “magic potion”

// output inventory
println(“You currently have ” + numItems + “ items:”);
for (int i=0; i<numItems; i++) {
        println(inventory[i]);
}
```

```java
StringList inventory = new StringList();

inventory.append(“banana”);
inventory.append(“stick”);
inventory.append(“BFG”);
inventory.append(“abomb”);
inventory.append(“magic potion”);

// output inventory
println(“You currently have ” + inventory.size() + “
items:”);
for (int i=0; i<inventory.size(); i++) {
        println(inventory.get(i));
}
```

No need for fixed size or tracking num elements etc.

If we have more than 10 elements, array will be an issue

```java
// output inventory
println("You currently have "
         + inventory.size() + " items:");
for (int i=0; i<inventory.size(); i++) {
  println(inventory.get(i));
}

// reverse order
println();
inventory.reverse();
println("Reversed: you currently have "
         + inventory.size() + " items:");
for (int i=0; i<inventory.size(); i++) {
  println(inventory.get(i));
}

// sort (alphabetically)
println();
inventory.sort();
println("sorted: you currently have "
         + inventory.size() + " items:");
for (int i=0; i<inventory.size(); i++) {
  println(inventory.get(i));
}
```

```
You currently have 5 items:
banana
stick
BFG
abomb
magic potion

Reversed: you currently have 5 items:
magic potion
abomb
BFG
stick
banana

sorted: you currently have 5 items:
abomb
banana
BFG
magic potion
stick
```

```java
// output inventory
println("You currently have "
         + inventory.size() + " items:");
for (int i=0; i<inventory.size(); i++) {
  println(inventory.get(i));
}

// reverse order
println();
inventory.reverse();
println("Reversed: you currently have "
         + inventory.size() + " items:");
for (int i=0; i<inventory.size(); i++) {
  println(inventory.get(i));
}

// sort (alphabetically)
println();
inventory.sort();
println("sorted: you currently have "
         + inventory.size() + " items:");
for (int i=0; i<inventory.size(); i++) {
  println(inventory.get(i));
}
```

```
You currently have 5 items:
banana
stick
BFG
abomb
magic potion

Reversed: you currently have 5 items:
magic potion
abomb
BFG
stick
banana

sorted: you currently have 5 items:
abomb
banana
BFG
magic potion
stick
```

Reading in simple text files

colours.txt

```
black 0 0 0
white 255 255 255




red 255 0 0
blue 0 0 255


green 0 255 0


grey 128 128 128


darkgrey 50 50 50
lightgrey 200 200 200
```

Can use a method directly:

```
Strings[] lines = loadStrings(filename);
```

filename (e.g. colours.txt) has to exist within the sketch folder

YORK UNIVERSITÉ UNIVERSITY

Example (read and remove blank/empty lines)

```
String[] readTextFile(String fileName) {

  String[] lines = loadStrings(fileName);
  StringList content;                                  // an arraylist of strings

  println(fileName + " has " + lines.length + " lines");
  if (!(lines.length>0)) return null;

  content = new StringList();  // instantiate empty StringList
  int empty = 0;
  int text = 0;

  for (int i=0; i<lines.length; i++) {

    if (!(lines[i].isEmpty()||lines[i].isBlank()) ) {
       content.append(lines[i]);
       text++;
    }
    else {
      empty++;
    }
  }
  println("-> there were " + empty + " empty lines");
  println("-> there were " + text + " non-empty lines");
  return content.toArray();
}
```

”parsing” the input file…

```
The file: colours.txt has 16 lines
-> there were 8 empty lines
-> there were 8 non-empty lines

colours.txt contains:
          black 0 0 0
          white 255 255 255
          red 255 0 0
          blue 0 0 255
          green 0 255 0
          grey 128 128 128
          darkgrey 50 50 50
          lightgrey 200 200 200
```

Example
(process the lines → using split on each)


```
void setup() {
  size(600, 800);

  String[] colourList = readTextFile("colours2.txt");
  println("\ncolours.txt contains: ");
  float sX = 100;
  float sY = 100;

  for (int i=0; i<colourList.length; i++) {
    println("\t" + colourList[i]);

    // for each colour... set a stroke colour, and draw colour in that colour
    String[] tokens = split(colourList[i], ' ');
    String colName = tokens[0];
    int colrgb = color(int(tokens[1]), int(tokens[2]), int(tokens[3]));

    stroke(colrgb);
    fill(colrgb);
    textSize(128);
    text(colName, sX, sY);
    sY += 100;
  }
}
```

Reading in simple text files

colours.txt

```
black 0 0 0 110.2
white 255 255 255 202.123




red 255 0 0 289.412
blue 0 0 255 334.98


green 0 255 0 431.5


grey 128 128 128 550.756


darkgrey 50 50 50 600
lightgrey 200 200 200 150.21
```

Format of the file has to be known

e.g. could use 4th number for positioning Text labels in y direction

Can have first line read and processed to figure out how to read the rest of the file (more on this next lecture)

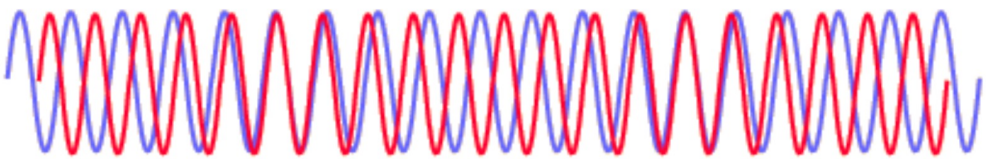
YORK UNIVERSITÉ UNIVERSITY

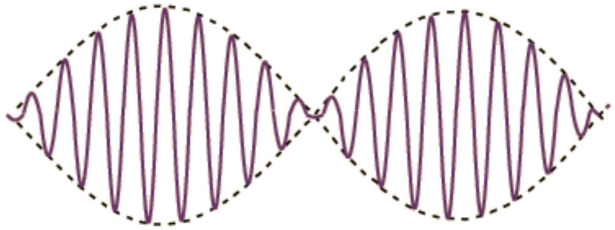
Example
(let file determine y positions of text labels)


```
void setup() {
  size(600, 800);

  String[] colourList = readTextFile("colours2.txt");
  println("\ncolours.txt contains: ");
  float sX = 100;
  float sY = 100;

  for (int i=0; i<colourList.length; i++) {
    println("\t" + colourList[i]);

    // for each colour... set a stroke colour, and draw colour in that colour
    String[] tokens = split(colourList[i], ' ');
    String colName = tokens[0];
    int colrgb = color(int(tokens[1]), int(tokens[2]), int(tokens[3]));

    stroke(colrgb);
    fill(colrgb);
    textSize(128);
    text(colName, sX, sY);
    sY = float(tokens[4]);
  }
}
```

Chord Example (revisited)

- Dyad (adding two tones: f1, f2)
  - E.g. harmonics (guitar tuning)





- Triad (adding three tones: f1,f2,f3)
  - E.g. “chords” :
    - A-Major
      - A4 (440.0 Hz) +
      - C5# (554.36526 Hz) +
      - E5 (659.25511 Hz)

Sound Resources?

- Some sound resources online
  - https://opengameart.org
  - https://www.audiomicro.com

- ** search for royalty free *.wav files
    - OR record your own!! (try it)

- More info on what we can do in processing with sound?
  - https://processing.org/tutorials/sound

YORK UNIVERSITÉ UNIVERSITY

https://opengameart.org/art-search-advanced?keys=&field_art_type_tid[]=13&sort_by=count&sort_order=DESC

Getting Started | https://login.teksav... | Getting Started | Cyberworlds 2022 | [Guide] Remote Ga... | Best Power Tool Bra... | Lennox 13ACX Air C... | Departmental Servi... | Labtest Mode [Tech... | Firs

OpenGameArt.Org

OpenID | Username or | Password | Log In | Register

Home | Browse | Submit Art | Collect | Forums | FAQ | Leaderboards | ❤ Donate

Advanced Search

Search

Title

Tags

Is one of

Enter a COMMA SEPARATED list of tags. (example: "sword, weapon, item")

Submitter

Art Type

- [ ] 2D Art
- [ ] 3D Art
- [ ] Concept Art
- [ ] Texture
- [ ] Music
- [x] Sound Effect
- [ ] Document

License(s)

- [ ] CC-BY 4.0
- [ ] CC-BY 3.0
- [ ] CC-BY-SA 4.0
- [ ] CC-BY-SA 3.0
- [ ] GPL 3.0
- [ ] GPL 2.0
- [ ] OGA-BY 3.0
- [ ] CC0
- [ ] LGPL 3.0
- [ ] LGPL 2.1

Sort By: Favorites

Order: Desc

Items per page: 24

Collect into...: Select a collection

Search

Chat with us!

Search Art

Legal notice regarding NFTs:

WARNING: Taking art from OpenGameArt.org to be sold as NFTs? You may be committing FRAUD. Visit this link for legal details: https://opengameart.org/content/warning-taking-art-from-opengameartorg-t...

Note of caution to NFT purchasers or those interested in trading NFTs: You could be getting scammed! Please visit this link for more information: https://opengameart.org/content/note-of-caution-to-nft-purchasers-or-tho...

Displaying 1 - 24 of 1427

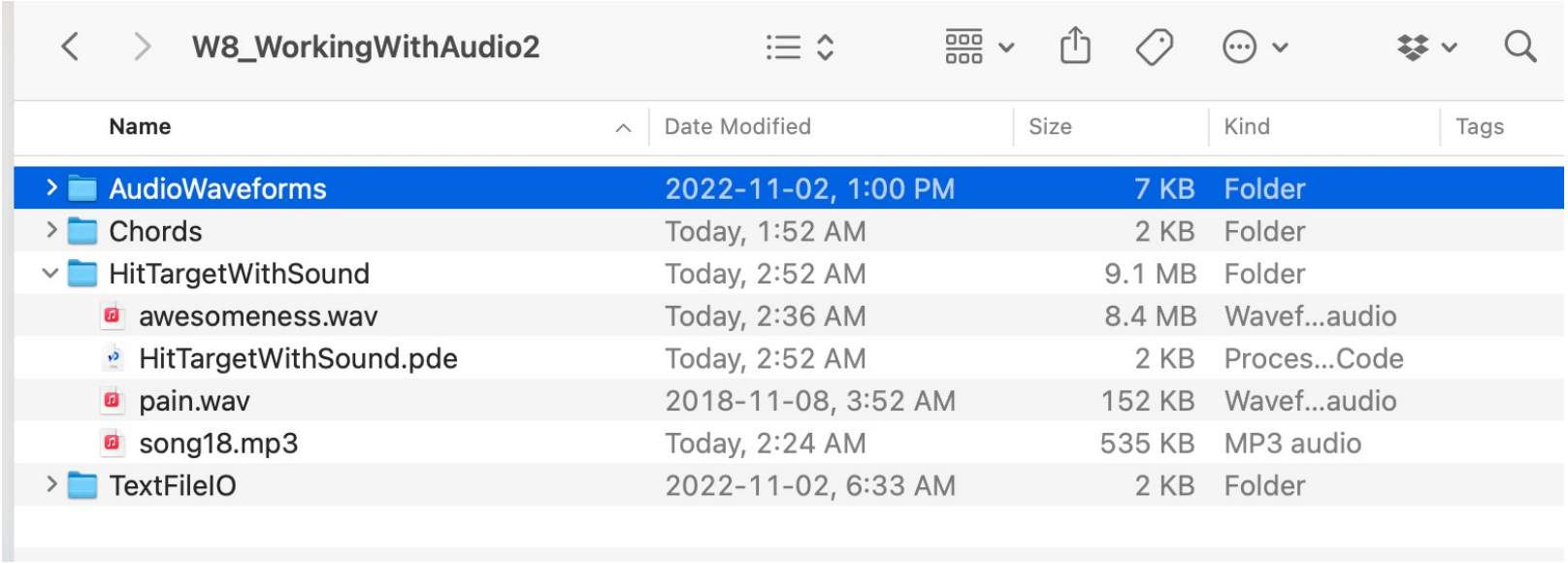
RPG Sound P... | 512 Sound Ef... | 50 RPG sound... | GUI Sound Ef... | 37 hits/punches | 51 UI sound ef...

63 Digital sou... | Inventory Sou... | Level up, pow... | Voiceover Pa... | Fantasy Soun... | Swish - bambo...

Battle Sound ... | UI Sound effe... | wind1 | 4 projectile l... | Chaingun, pist... | Spell Sounds ...

YORK UNIVERSITÉ UNIVERSITY

Save / move audio files into sketch folder



YORK UNIVERSITÉ UNIVERSITY

Multiple Sound Effects (in one file)

- No problem
- Can play background music, and while it is playing can play new SoundFiles (e.g. for special effects)

E.g. Adding sound to our target/hit-test demo

```
import processing.sound.*;
//…
SoundFile musicGame;
SoundFile effect;

void setup() {
  size(640, 480);
  generateTarget();
  musicGame = new SoundFile(this,"awesomeness.wav");
  effect = new SoundFile(this,"pain.wav");
  musicGame.loop();
}

void moveProjectile(float x0, float y0, float v0, float theta, float t) {
  background(255, 255, 255);
  float x = x0 + v0*t*cos(theta);
  float y = y0 + v0*t*sin(theta) + 0.5*GRAVITY*pow(t, 2) ;
  noFill();
  stroke(0,0,255);
  ellipseMode(RADIUS);
  circle(x, y, 10);
  line(x0,y0,x1,y1);
  println("(" + x + ", " + y + ") hit target = " + hitTarget(x,y));
  if (hitTarget(x,y)) { hit = true; effect.play(); }
}
void draw() {
  t+=0.05;
  if (!hit) moveProjectile(x0,y0,v0,theta,t);
  drawTarget();
}
```

```
if (hitTarget(x,y)) {
   hit = true;
   effect.play();
   musicGame.stop();

}
```

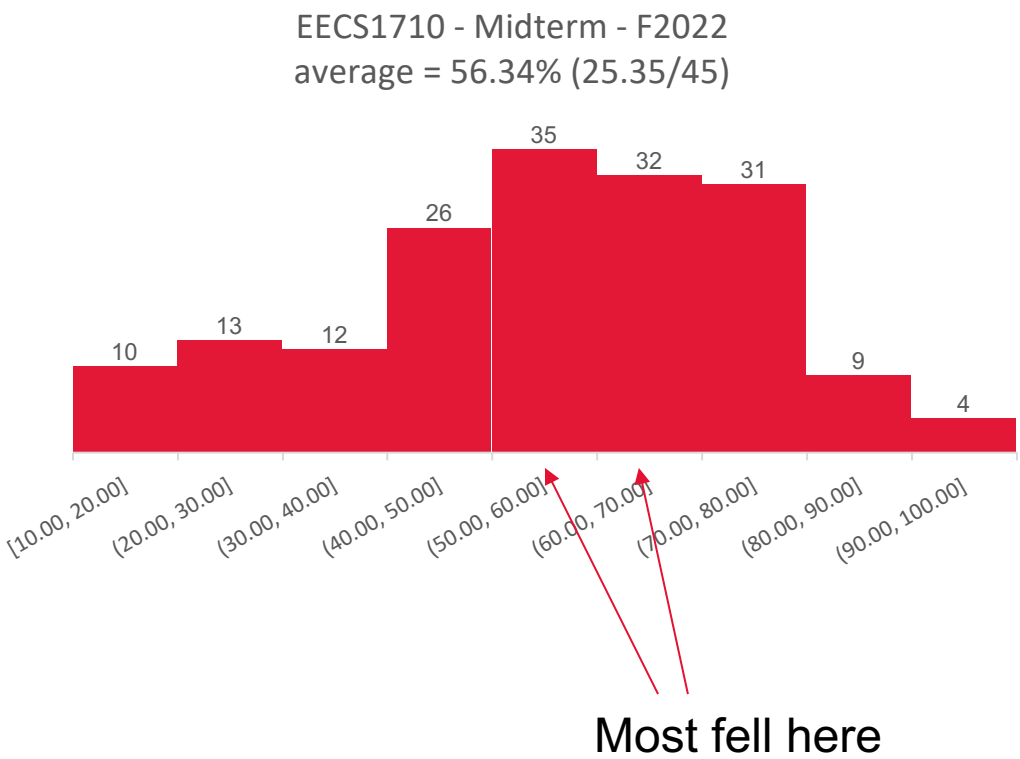
What else can we do?
Visualizing Frequencies

- Instead of amplitude vs. time (normal waveform view)
- Can look at amplitude vs. frequency (fft view)

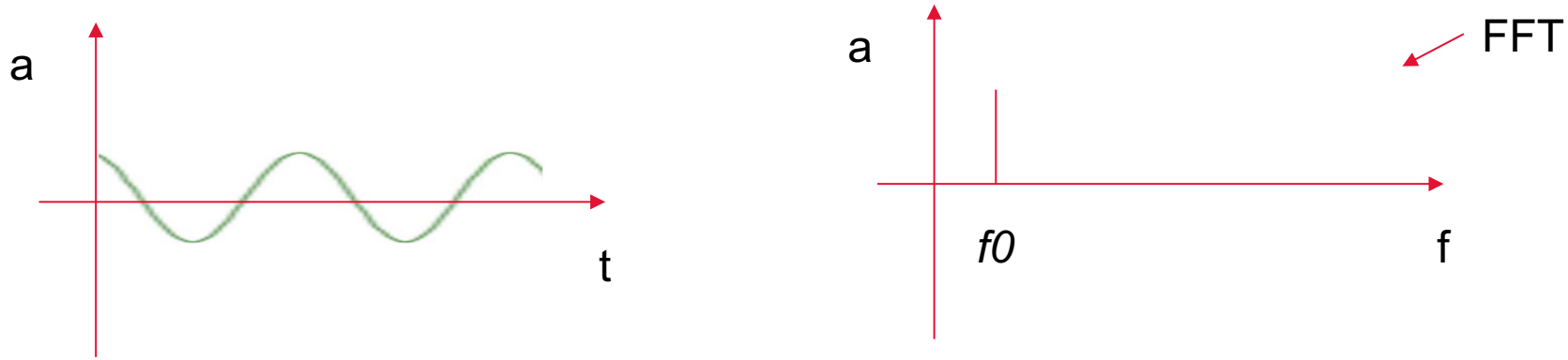
- FFT object → allows us to connect a frequency analyser to a SoundFile
  - Creates a kind of bar graph (histogram) that accumulates and displays how much of each frequency occurs in the sound
    - Can think of this as how much amplitude of any given tone is in the sound signal
  - Low frequency tones create stronger responses near zero freq
  - High frequency tones create stronger responses at higher freq

Analogy

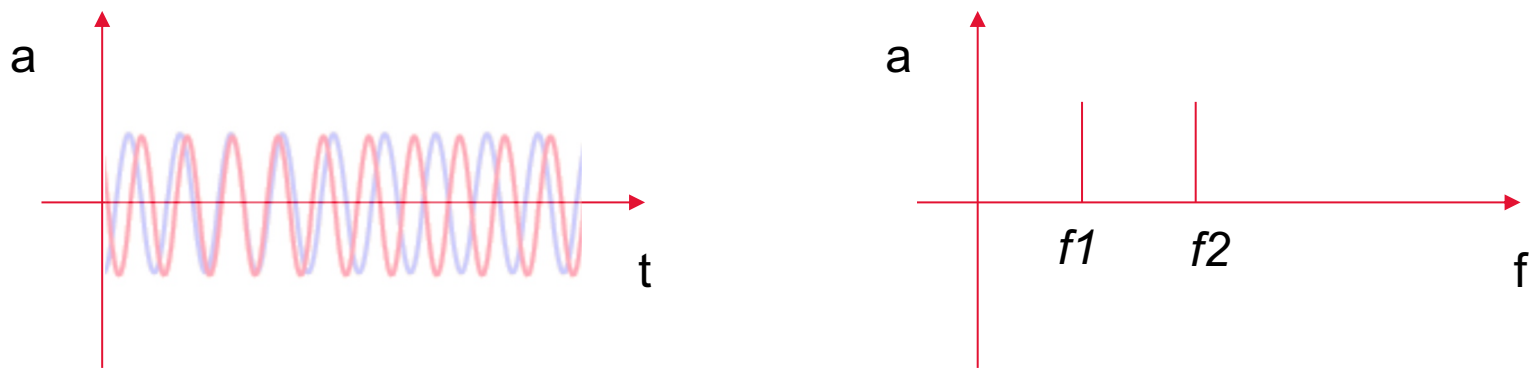
- Midterm marks (bars represent how many fall into specific mark ranges)



FFT Intuition (indicates presence of tones):



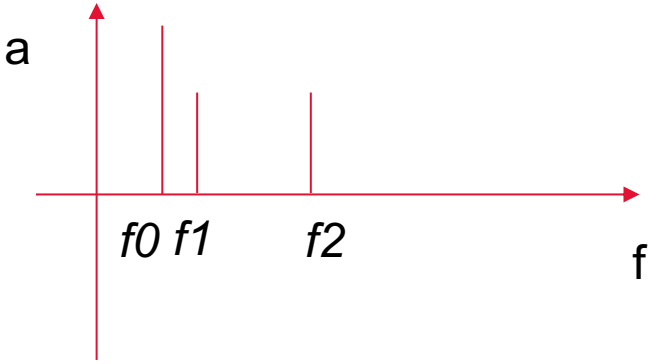
pure (single) tone



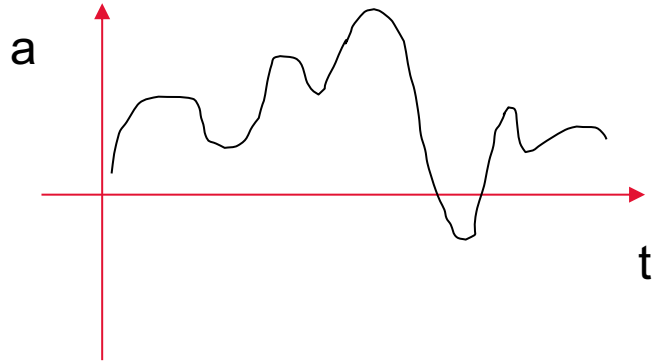
Dyad (2 tones)

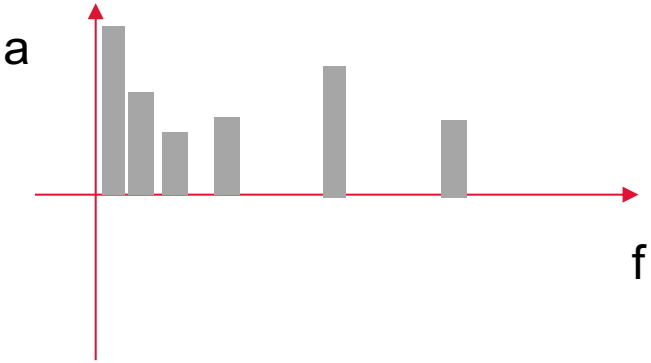
YORK UNIVERSITÉ UNIVERSITY

Intuition:



more freqs





arbitrary sounds/music

Using an array to store FFT

```
SoundFile file;
FFT fft;

// Define how many FFT bands to use (this needs to be a power of two)
int bands = 32;

// Define a smoothing factor (between consecutive snapshots of the waveform)
// normally cutting a waveform off and analysing creates high freq artifacts
float smoothingFactor = 0.2;

// Create a vector to store the smoothed spectrum data in
float[] spectrum = new float[bands];

// scaling factor for adjusting the height of the rectangles
int scale = 5;
// Declare a drawing variable for calculating the width of the
float barWidth;
```

```
void setup() {
  size(640, 360);
  background(255);

  // Load a soundfile from the /data folder of the sketch and play it back
  file = new SoundFile(this, "awesomeness.wav");

  barWidth = width/float(bands);
  // Create the FFT analyzer and connect the playing soundfile to it.
  fft = new FFT(this, bands);
  fft.input(file);

}

void draw() {
  background(125, 255, 125);
  fill(255, 0, 150);
  noStroke();

  // Perform the analysis
  fft.analyze();

  for (int i = 0; i < bands; i++) {
    // Smooth the FFT spectrum data by smoothing factor
    sum[i] += (fft.spectrum[i] - sum[i]) * smoothingFactor;

    // Draw the rectangles, adjust their height using the scale factor
    rect(i*barWidth, height, barWidth, -sum[i]*height*scale);
  }
}
```

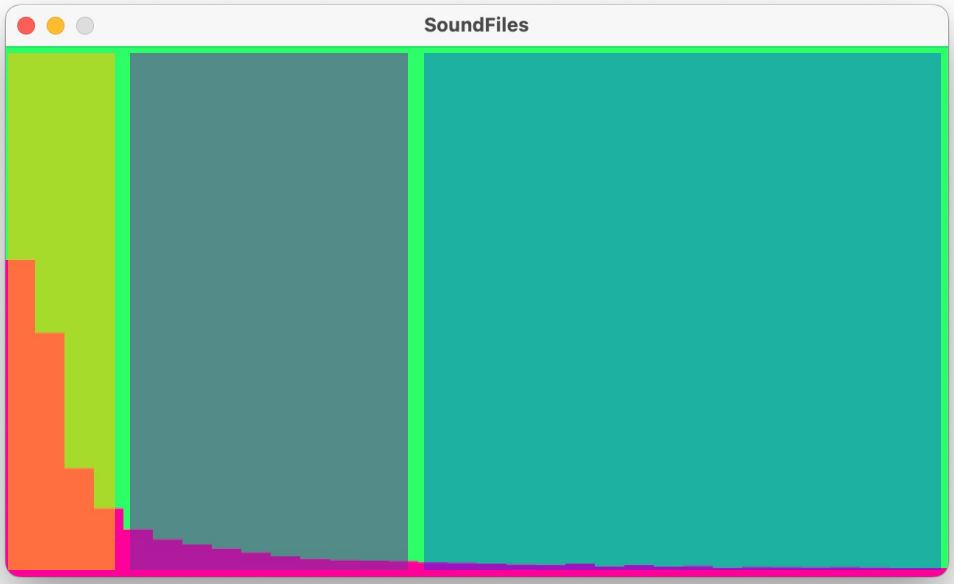
Snapshot of frequencies → acts as a signature of the sound

- Can use this for many things
- E.g. comparing sounds (are they similar?)
- Usually if we know signature (can regenerate signal) by doing inverse FFT → we can do filtering on fft easily

What else can we do?
Effects Filtering

- Filtering (low/mid/high frequencies)
  - Low makes sound “softer”, and more ”muddy” or bassy
  - High makes it sharper (more treble)

- Reverb and other effects (see Effects section of Sound library reference documentation)



LOW PASS FILTER (keep low frequencies)

BAND PASS FILTER (keep midrange freqs)

HIGH PASS FILTER (keep only high freqs)

YORK UNIVERSITÉ UNIVERSITY