

ATIAM - Web Audio

Benjamin Matuszewski et Victor Paredes

Introduction / Context

Web Technologies - *an history*

Internet

Set of protocols dedicated to transfert data (TCP, IP, ...)
and interconnect computers and networks

1972 - First demonstration of *ARPANET*
sent a message between UCLA and Stanford

1983 - *ARPANET* is officially renamed *Internet*

World Wide Web

System dedicated to the sharing of hypertext informations
built on top of Internet

1989-1992 - Development at the CERN by T. Berners Lee
et *Robert Caillau (software and protocoles)*

1993 - the CERN opens the technologies to the public

Web Technologies - *protocols*

HTTP

HyperText Transfert Protocol

URL

Uniform Resource Locator

HTML

HyperText Markup Language



serveur

GET `http://example.com?id=123` HTTP/0.1

HTTP/1.0 200 OK

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>web page</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <div id="container" class="container"></div>
    // ...
    <script src="script.js"></script>
  </body>
</html>
```



client

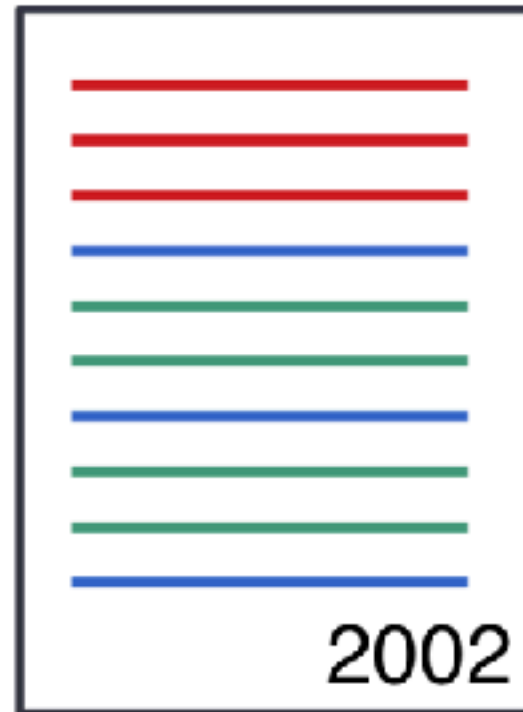
Web Technologies - *languages*

User Interface creation on the web

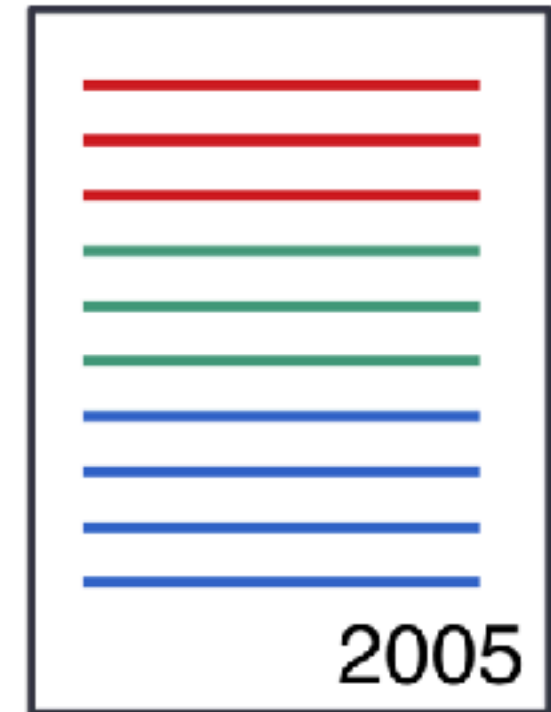
Structure
(HTML)



Presentation
(CSS)



Behavior
(Javascript)



Accessibility, Portability, Maintainability, Reduced Latency, Graceful Degradation

The Web as a Creative Platform

ubiquity

almost every device implements web standards

interactive multimedia

HTML5/CSS, Web GL, Canvas, Web Audio API, DeviceMotion/
Orientation, Geolocation

networking

HTTP, WebSockets, WebRTC

rapid prototyping & interoperability

very rapid development / deployment cycles

The HTML / CSS / javascript Trinity

HTML (Hypertext Markup Language)

Define **content** and **structure**. Not a programming language but a **formatting language**.

Anatomy of an HTML tag

```
<tagName attribute="value">  
    content  
</tagName>
```

```
<p id="my-paragraph">  
    hi there  
</p>
```

Used for CSS and JavaScript reference

HTML tags examples

- `<h1> headline </h1>`
- `<p> paragraph </p>`
- `<!-- unordered list -->`
``
 ` list item `
``
- ` link `

CSS (Cascading Style Sheet)

Define **presentation** and **formatting** rules of an HTML or XML document.

Anatomy of a CSS chunk

```
selector {  
  declaration  
}
```

```
p {  
  color: #000  
}
```

- A selector can be:
 - An element (p, h)
 - A class (.class)
 - An id (#)

Javascript

Programming language. Controls page **behavior**. Makes the web interactive.

- check text value in form
- drag and drop
- dropdown menu
- etc.

not related to Java

javascript

variables

```
const a = true;  
let b = 0;
```

operators

```
let b = 2 * (3 + 1);
```

strings

```
const title = `hello world`;
```

functions

```
function double(x) { return x * 2; }  
const double = (x) => x * 2;
```

arrays

```
const list = [1, 2, 3, 4];
```

objects

```
const dog = {  
  name: `doog`,  
  age: 4,  
  bark: () => console.log(`wouaf`),  
};
```

javascript

conditionals

```
if (myVar === true) {  
  doThis();  
} else {  
  doThat();  
}
```

loops

```
for (let i = 0; i < 10; i++) {  
  doSomethingWith(i);  
}
```

javascript

`&&` logical AND

`1 && 1 = 1`

`1 && 0 = 0`

`0 && 1 = 0`

`0 && 0 = 0`

`||` logical OR

`1 || 1 = 1`

`1 || 0 = 1`

`0 || 1 = 1`

`0 || 0 = 0`

...some useful resources

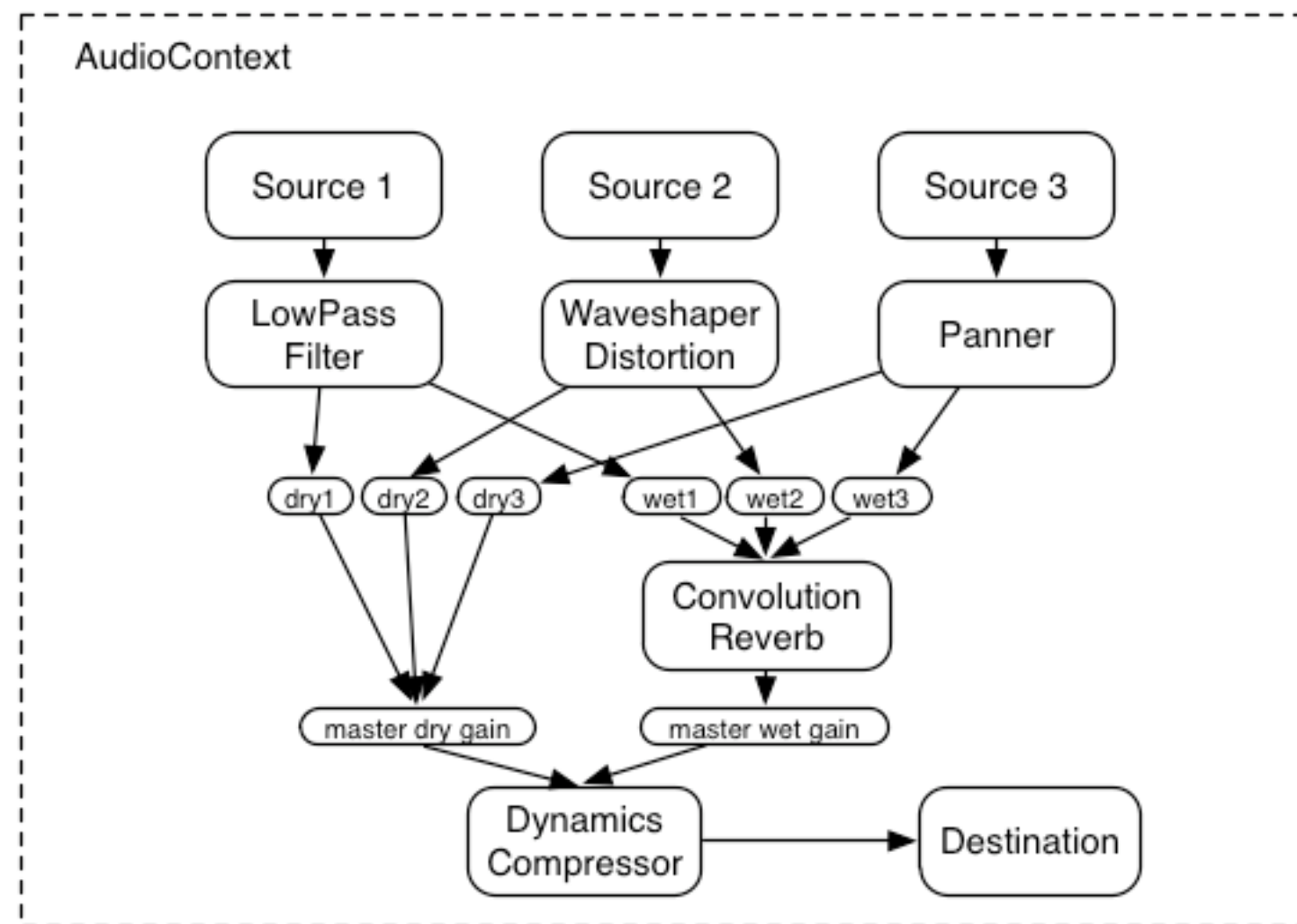
<https://developer.mozilla.org/fr/>

<http://javascript.info/>

<https://babeljs.io/learn-es2015/>

Audio on the Web

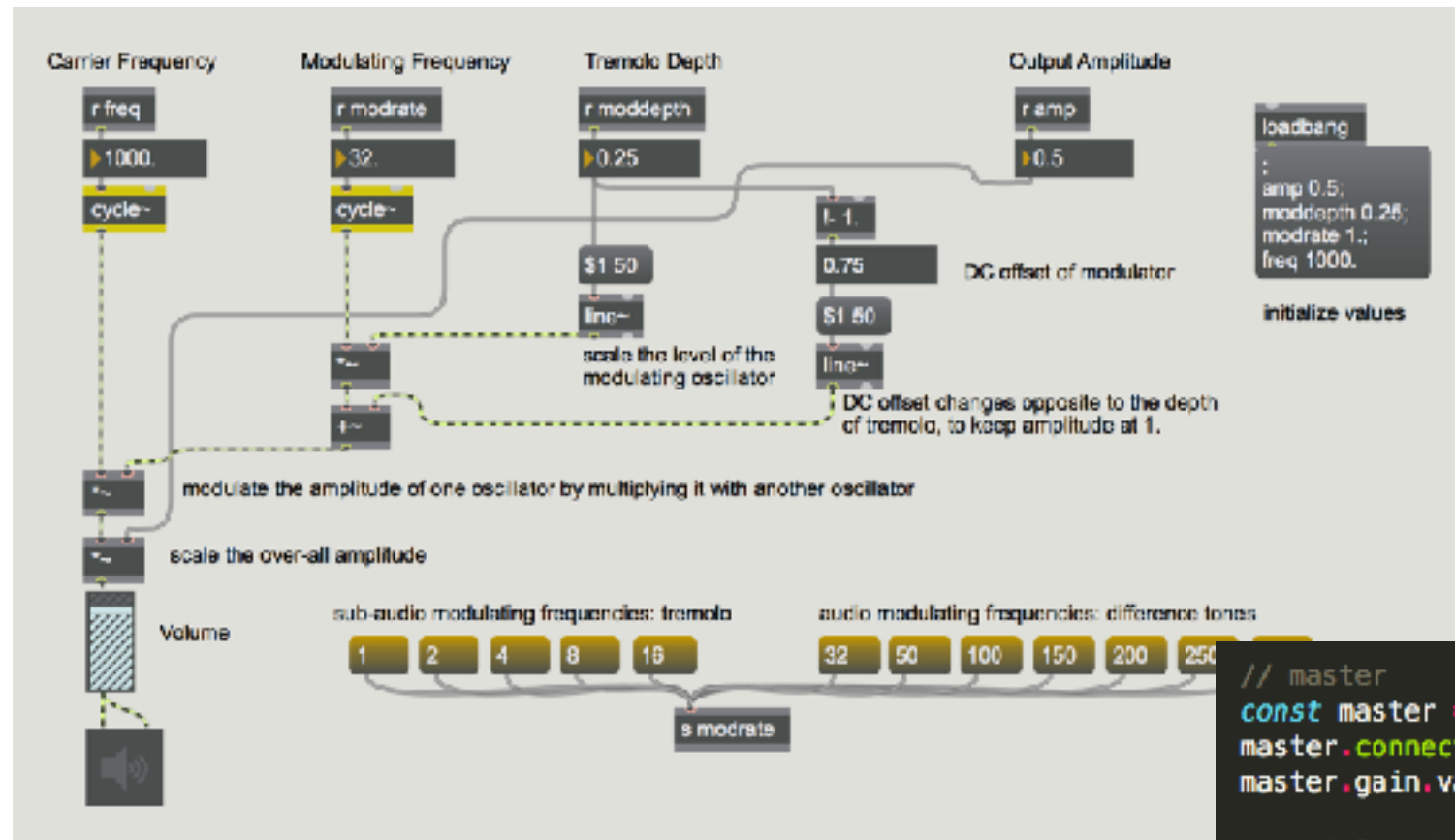
WebAudio API



specification

<https://webaudio.github.io/web-audio-api/>

WebAudio API



```
// master
const master = audioContext.createGain();
master.connect(audioContext.destination);
master.gain.value = 0; // default to muted

// modulated amplitude
const amplitude = audioContext.createGain();
amplitude.connect(master);
amplitude.gain.value = 1 - defaultDepth;
amplitude.gain.setValueAtTime(1 - defaultDepth, audioContext.currentTime);

// carrier
const carrier = audioContext.createOscillator();
carrier.connect(amplitude);
carrier.frequency.value = 1000;

// modulation
const depth = audioContext.createGain();
depth.connect(amplitude.gain);
depth.gain.value = defaultDepth;
depth.gain.setValueAtTime(defaultDepth, audioContext.currentTime);

const mod = audioContext.createOscillator();
mod.frequency.value = 1;
mod.connect(depth);

carrier.start(audioContext.currentTime);
mod.start(audioContext.currentTime);
```

Setting Up a Development Environment

Development Environment

Browsers

prefer Chrome or Firefox

Text Editor

Sublime, VS Studio, ...

Tools

NodeJS (<https://nodejs.org/en/> prefer LTS), npm

Les Classes

Une classe décrit les composantes communes d'un ensemble d'objets, à travers :

- Des propriétés → l'état d'un objet
- Des méthodes → le comportement d'un objet

Exemple de classe qui décrit un point dans un espace en 2D:

```
class Point(object):  
    def __init__(self, x, y):  
        self.x = x  
        self.y = y  
  
    def translate(self, tx, ty):  
        self.x += tx  
        self.y += ty  
  
    def is_origin(self):  
        return (self.x == 0) and (self.y == 0)
```

← Constructeur : méthode appelée à la création de l'objet

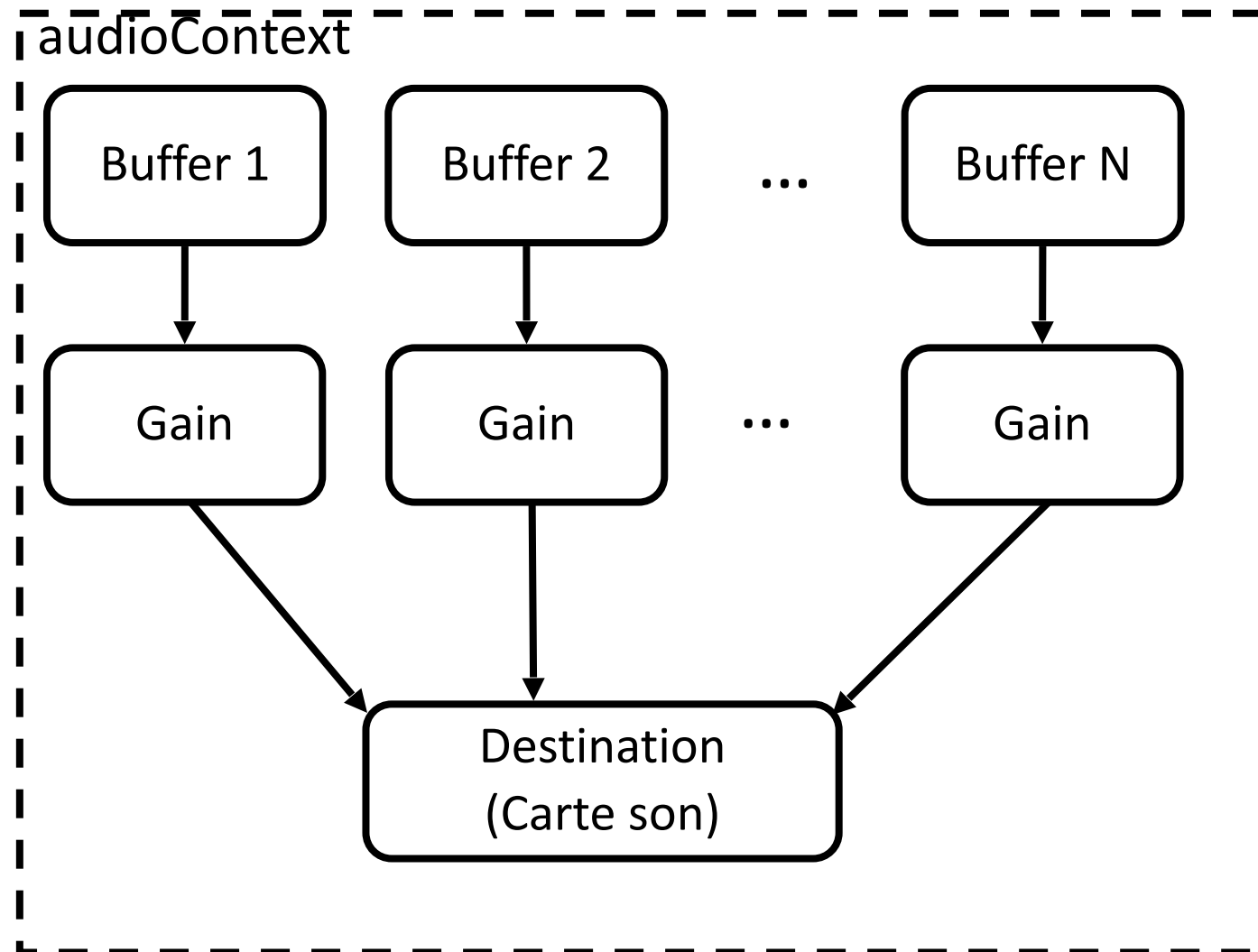
} 2 propriétés : coordonnées en x et y

} Méthode qui translate le point

} Méthode qui vérifie si le point est l'origine du repère

Autres exemples : <https://web.mit.edu/music21/doc/moduleReference/moduleNote.html>
<https://developer.mozilla.org/en-US/docs/Web/API/AudioBuffer>

Playing audio buffers



```
const soundfiles = [
  './assets/kick.wav',
  './assets/snare.wav',
  './assets/clap.wav',
  './assets/hh.wav',
  './assets/rimshot.wav',
];
```

Liste des chemins vers les fichiers sons

```
const model = {
  buffers: {},
  volume: 1,
};
```

Objet 'model' avec 2 propriétés :

- buffers : objet qui va contenir les buffers
- volume : float du volume général

Ces variables sont déclarées globalement.

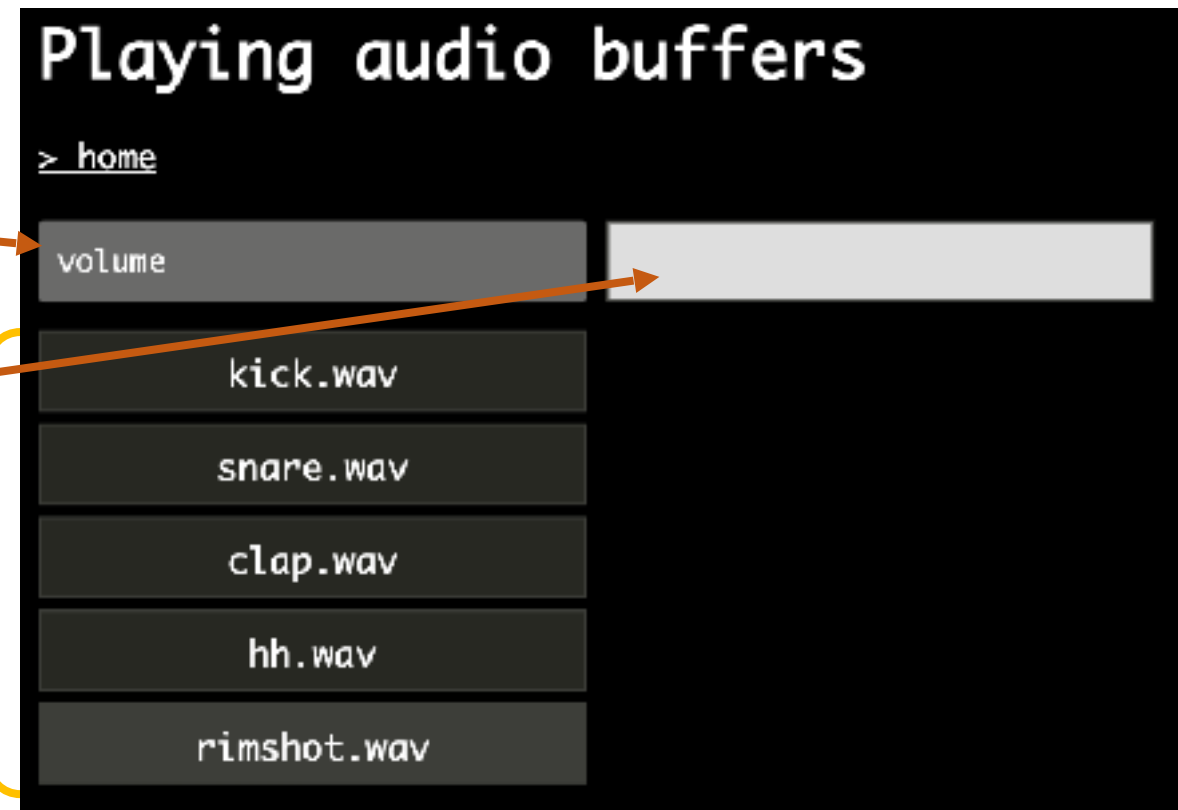
1. Charger les fichiers sons dans des buffers
2. Ajouter ces buffers au modèle

Playing audio buffers

Fonction qui touche directement au html :

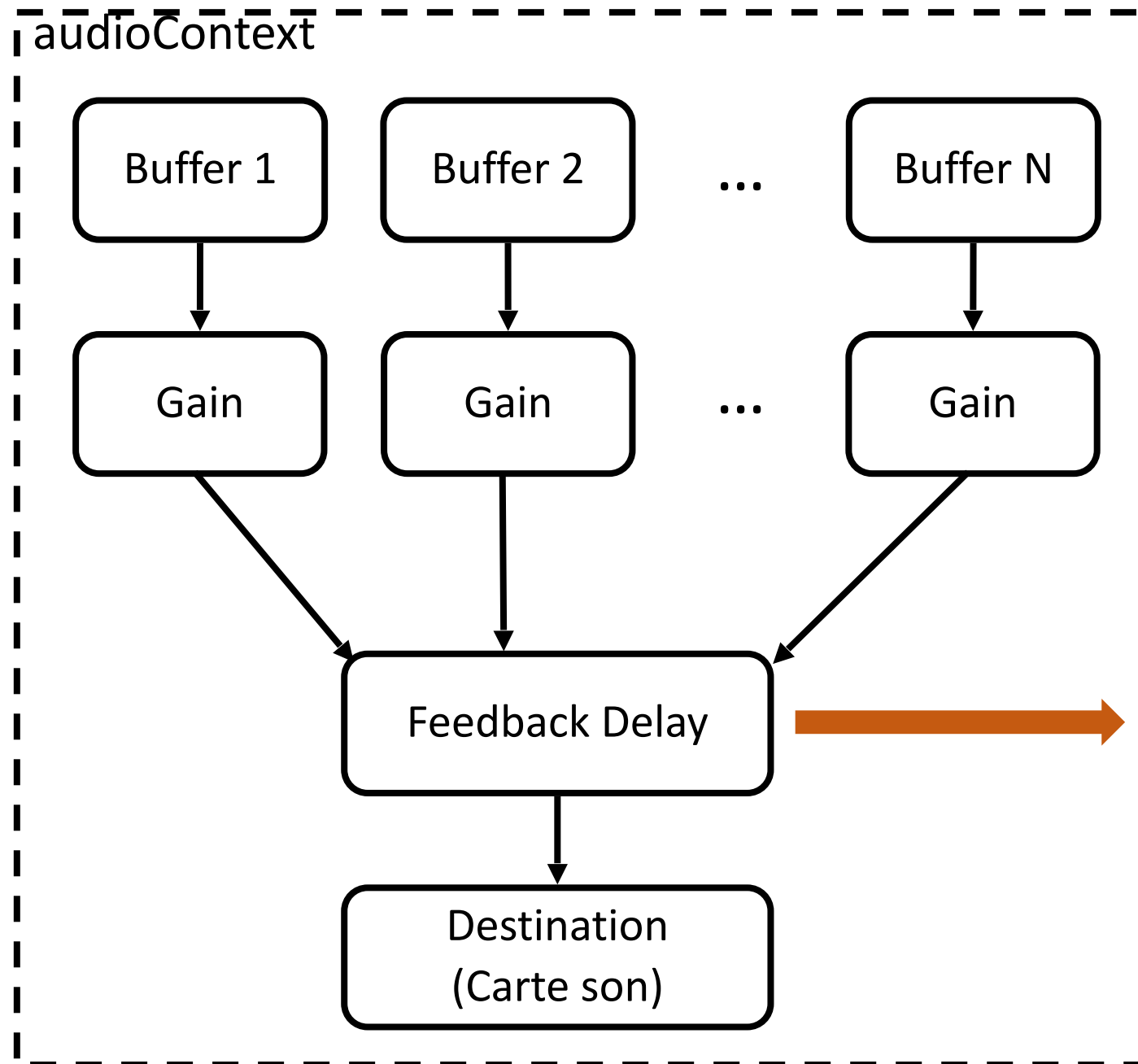
```
// GUI
function renderGUI() {
  const $main = document.querySelector('.main');

  render(html`
    <div style="padding-bottom: 10px">
      <sc-text
        value="volume"
        readonly
      ></sc-text>
      <sc-slider
        min="0"
        max="1"
        value="${model.volume}"
        @input=${e => model.volume = e.detail.value}
      ></sc-slider>
    </div>
    ${Object.keys(model.buffers).map(filename => {
      return html`
        <sc-button
          style="display: block; padding-bottom: 4px"
          value="${filename}"
          @input=${e => playSound(filename)}
        ></sc-button>
      `
    })}
  `
    , $main);
}
```



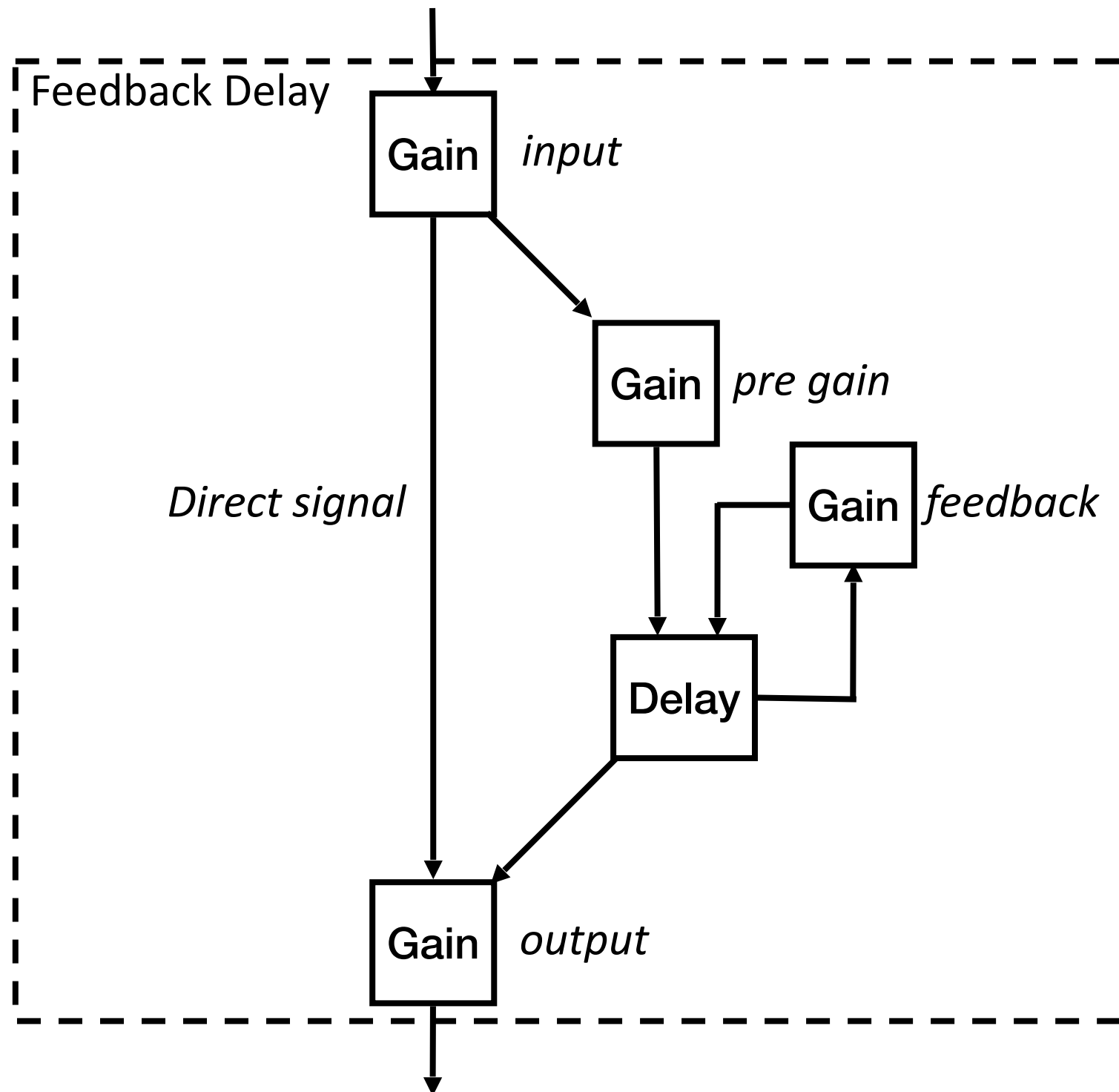
Reste à écrire la fonction 'playSound' dans laquelle on va définir le graph precedent.

Feedback Delay

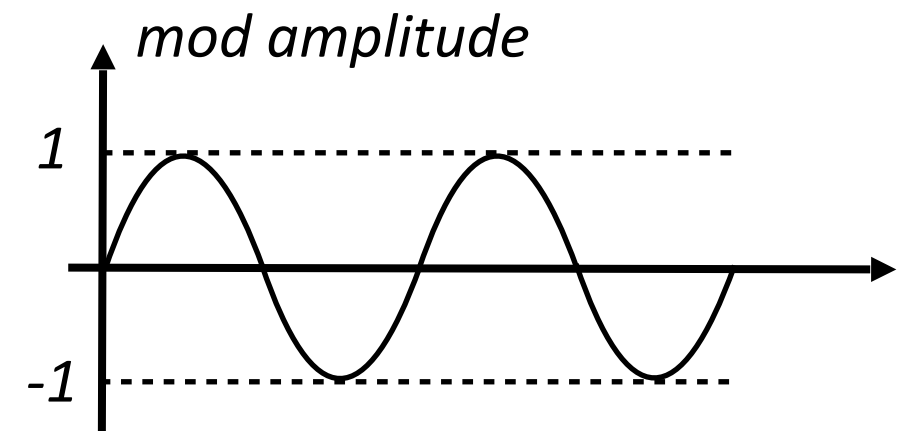
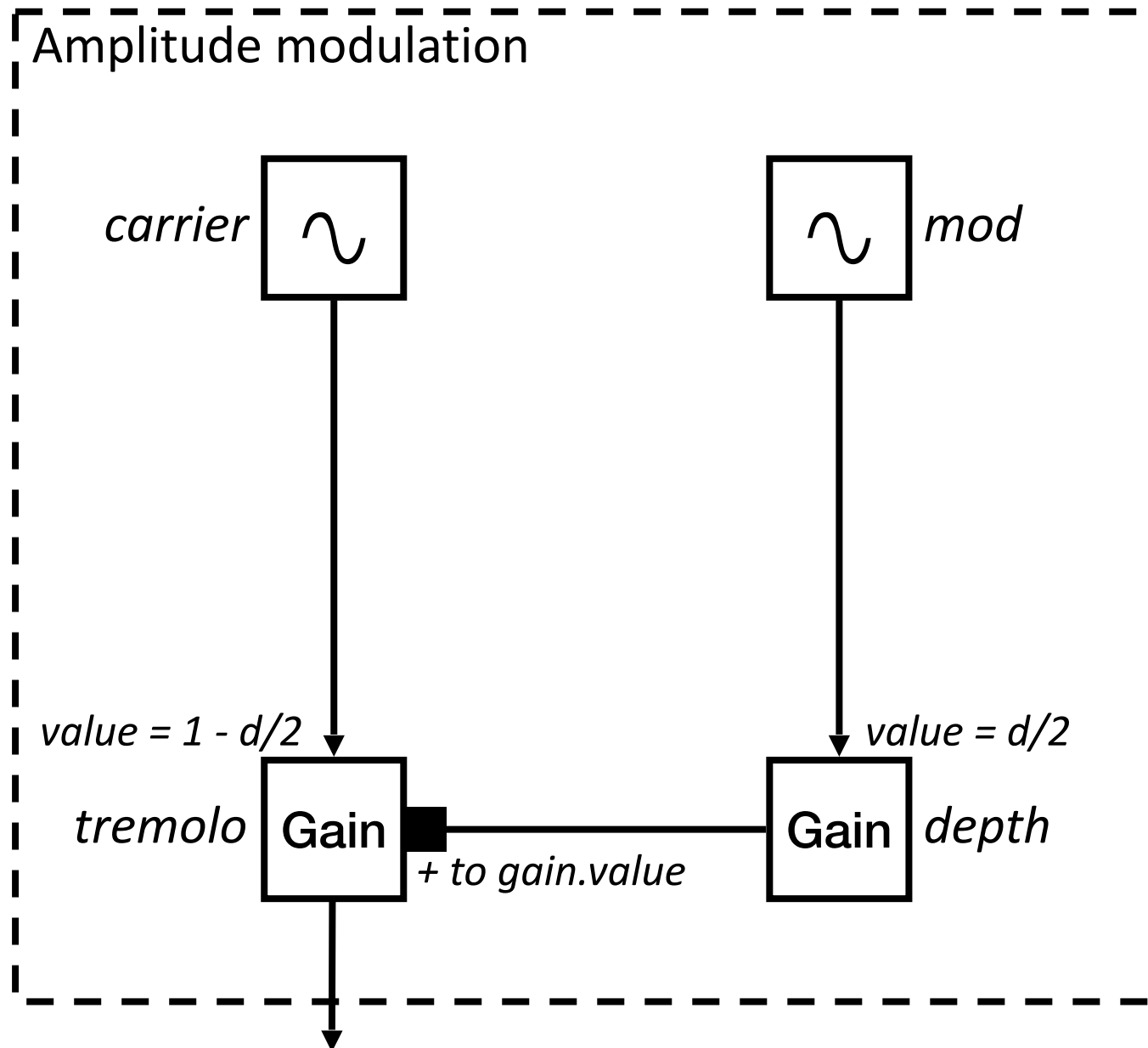


On doit créer une nouvelle classe qui s'intègre correctement dans l'API WebAudio

Feedback Delay

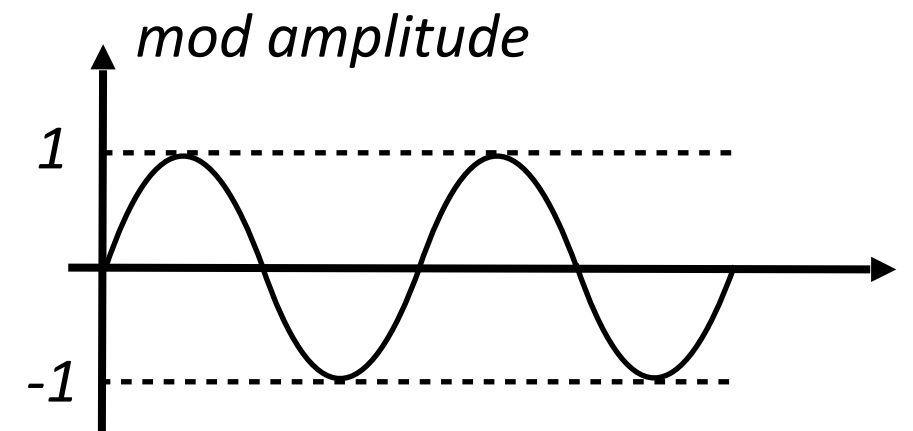
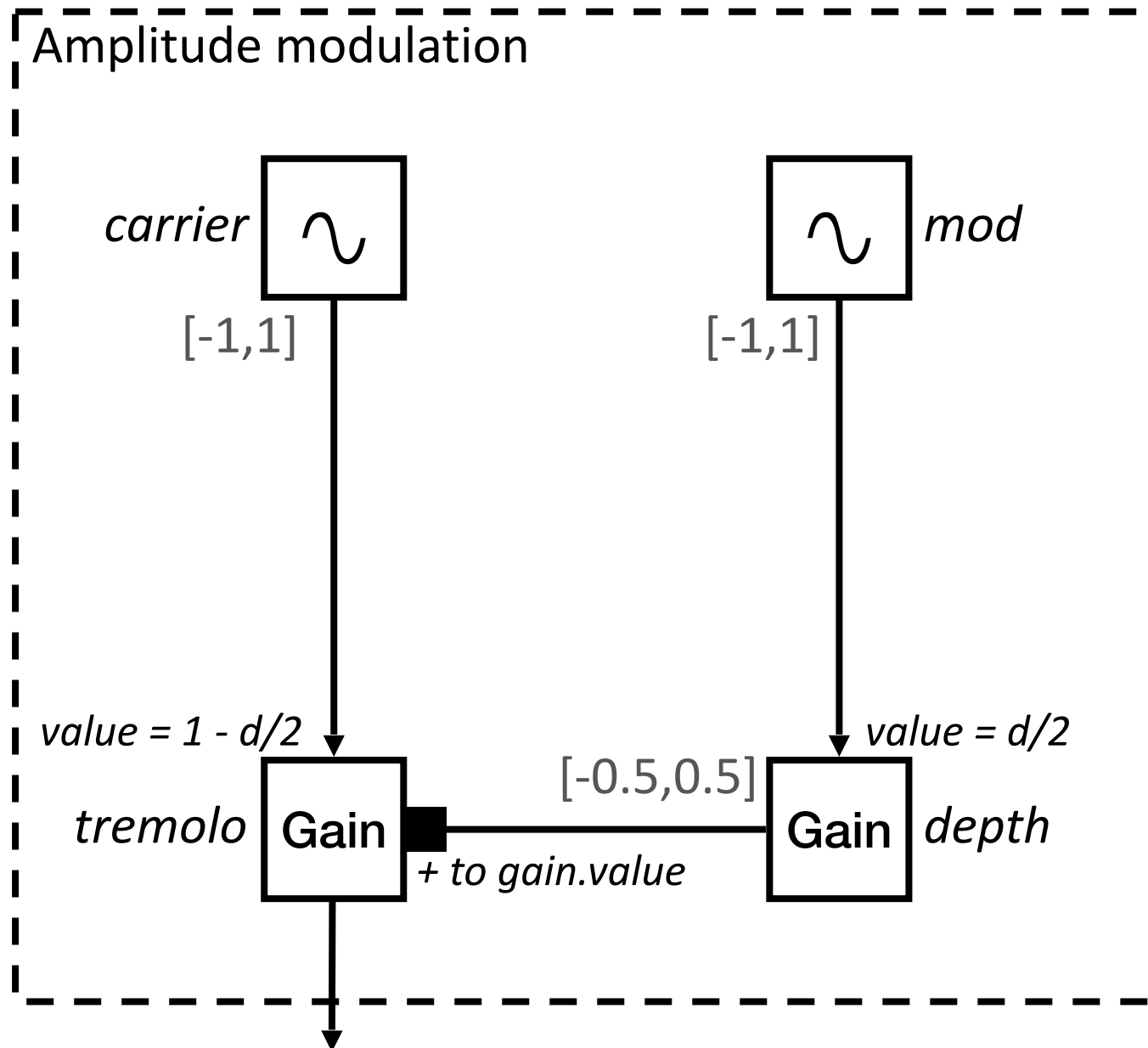


Amplitude Modulation

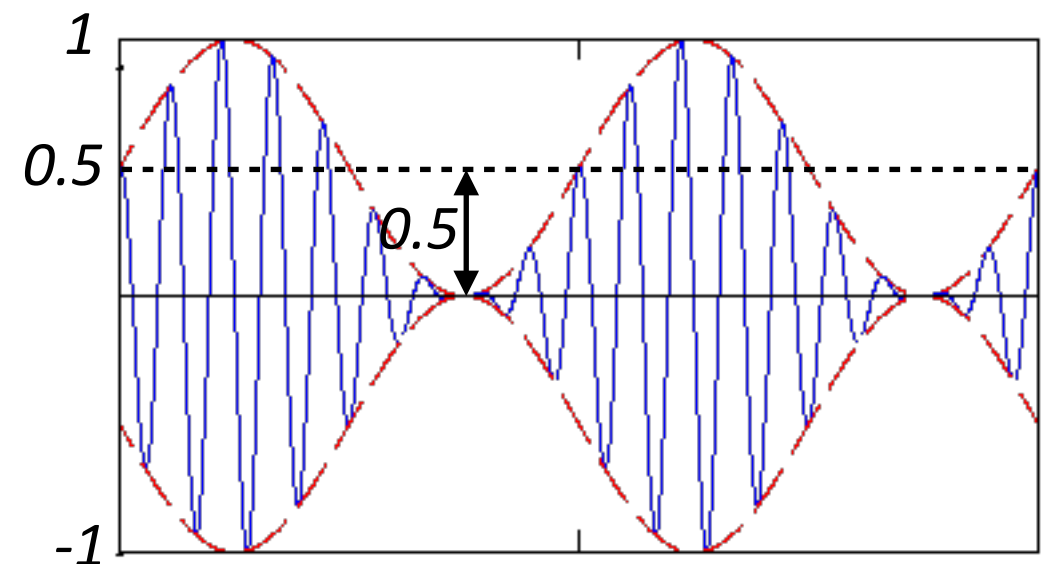


Paramètre de quantité de modulation :
 $d = 0 \rightarrow$ aucune modulation
 $d = 1 \rightarrow$ modulation max

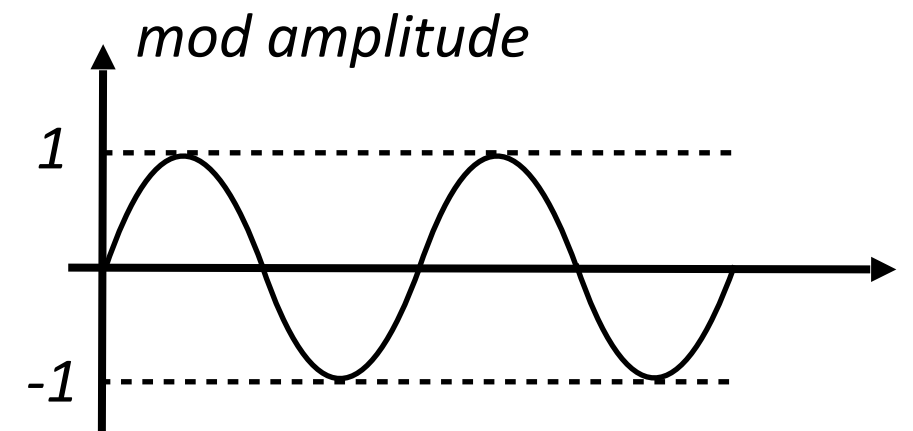
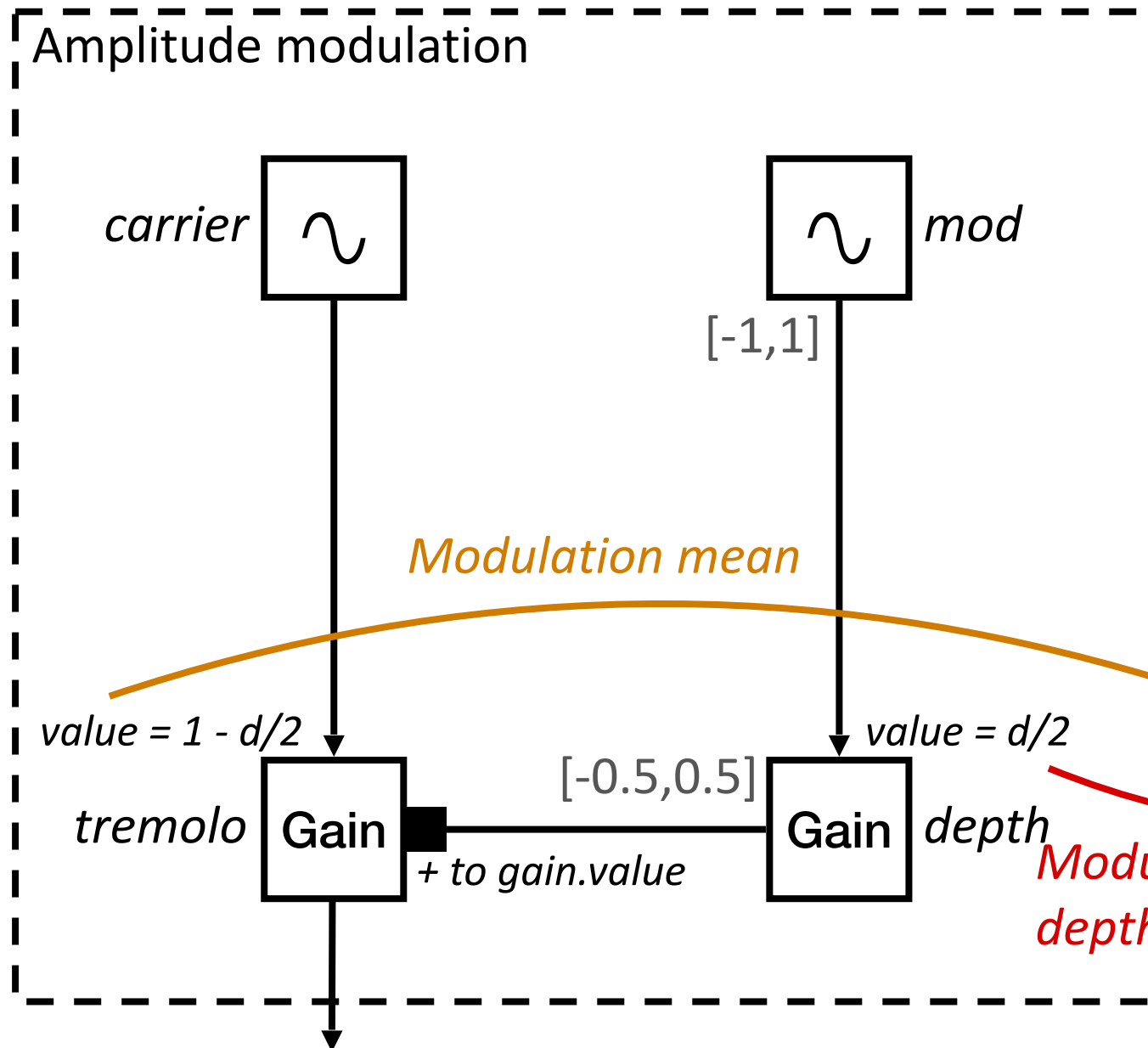
Amplitude Modulation



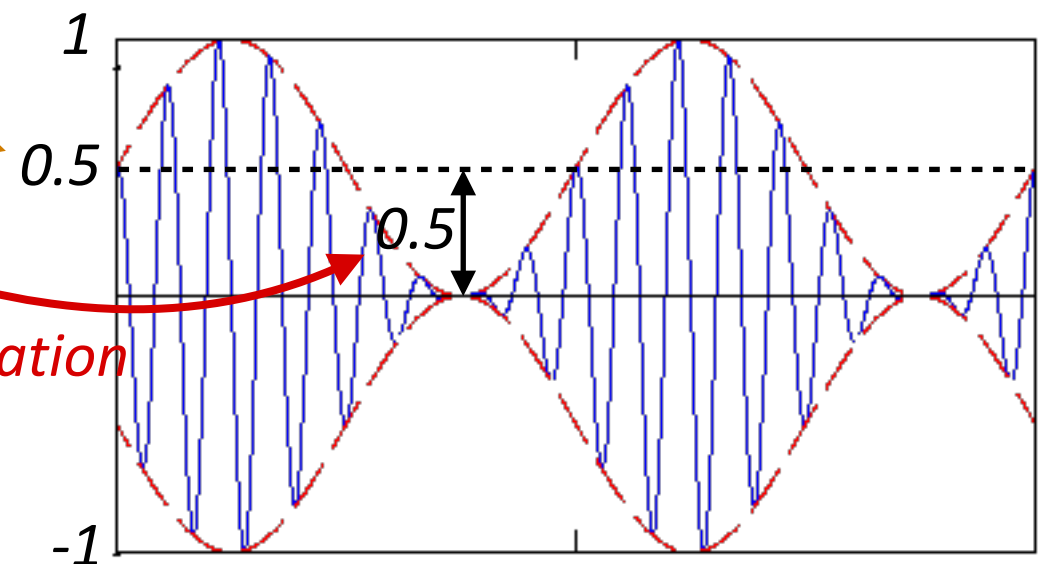
Paramètre de quantité de modulation :
 $d = 0 \rightarrow$ aucune modulation
 $d = 1 \rightarrow$ modulation max
Ex : $d = 1$



Amplitude Modulation



Paramètre de quantité de modulation :
 $d = 0 \rightarrow$ aucune modulation
 $d = 1 \rightarrow$ modulation max
Ex : $d = 1$



Libraries used during the class

Resume audio context helper (one-liner to automatically create a button to resume the AudioContext) :

<https://github.com/ircam-ismm/resume-audio-context>

Waves-masters (library that contains the scheduler) :

<https://github.com/wavesjs/waves-masters>

Waves-loaders (to load and decode audio files) :

<https://github.com/wavesjs/waves-loaders>

Simple-components (set of components to create GUIs) :

<https://github.com/ircam-ismm/simple-components>

<https://ircam-ismm.github.io/simple-components/> (examples)

Some papers

Steven Yi Victor Lazzarini et Joseph Timoney. _Web Audio: Some Critical Considerations_. In : VI Ubimus. 2015.

Benjamin Tayler. _A History of the Audience as a Speaker Array_. In Proceedings of the NIME'17 Conference, 2017.

Chris Wilson. _A Tale of Two Clocks - Scheduling Web Audio with Precision_. 2013, <http://www.html5rocks.com/en/tutorials/audio/scheduling/>.

Lonce Wyse and Srikumar Subramanian. 2013. _The Viability of the Web Browser as a Computer Music Platform_. Computer Music Journal 37, 4, 2013, pp. 10–23.

Lonce Wyse. _Spatially Distributed Sound Computing and Rendering Using the Web Audio Platform_. In 1st Web Audio Conference, 2015, Paris.

Norbert Schnell, Victor Saiz, Karim Barkati, Samuel Goldszmidt. Of Time Engines and Masters An API for Scheduling and Synchronizing the Generation and Playback of Event Sequences and Media Streams for the Web Audio API. WAC, Jan 2015, Paris, France. <hal-01256952>