Une (très courte) introduction à l'API WebAudio

Option audionumérique - ensea
S. Reynal

25 janvier 2017

1 Introduction

Ce TP vous propose de vous initier à l'API WebAudio à travers la réalisation d'un embryon de page web incluant du traitement du signal audio. En terme de programmation de site web, on distingue :

- les langages client-side, c'est-à-dire que le code est exécuté par le "client" (le navigateur : chrome, firefox, IE, ...) : HTML, CSS et Javascript pour l'essentiel
- les langages server-side, pour lesquels le code est exécuté par le serveur : PHP, Java (servlet), Ruby, Python, C♯ et .NET, ...

1.1 HTML, CSS, JavaScript, PHP, ...

HTML (HyperText Markup Language) est un langage de spécification de structure de document (titre, sous-titre, paragraphes, sections, notes de bas de page, image, etc.). CSS est un langage de description de style (gras, italique, taille de police de caractère, ...). Ces deux langages sont statiques, au sens où ils décrivent un document sans réelle possibilité d'interaction avec l'utilisateur (à l'exception des hyperliens). Javascript, au contraire, permet d'ajouter de l'interactivité aux pages web. Il s'agit d'un langage hérité du langage Java, avec quelques modifications de syntaxe, et une grammaire plus tolérante. Enfin, PHP permet, côté serveur, de générer des pages web dynamiquement, c'est-à-dire que le code HTML (notamment) est généré à la volée par le serveur, par exemple en fonction du contenu d'une base de donnée. Programmer un site web suppose de maîtriser un peu de tous ces langages...

1.2 l'API WebAudio

L'API WebAudio est une API (Application Programming Interface = ici, ensemble de fonctions ou de classes regroupés dans une bibliothèque) en langage JavaScript, permettant d'étendre les fonctionnalités d'une page

web en y incluant une chaîne de traitement audio. Auparavant (i.e., avant HTML 5), ces fonctionnalités (en fait, seulement une partie d'entre elle) étaient souvent réalisées via un plug-in de type flashplayer. L'introduction d'une telle API dans HTML 5 est donc une réelle avancée offrant davantage de souplesse.

Les applications sont variées. On peut songer aux moteurs sonores pour les jeux en ligne programmés en Javascript, ainsi qu'aux applications musicales en ligne, y compris sur smartphone puisque Safari (iPhone) et Chrome (Android) supportent l'API Webaudio. Mentionnons quelques applications :

- ToneCraft, un système de composition de musique orienté "electro", utilisant une interface 3D : http://labs.dinahmoe.com/ToneCraft/
- Plink, un jeu de composition musicale collective : http://labs.dinahmoe.com/plink/
- Un visualizer audio en ligne qui accepte des URL Soundcloud : http://www.michaelbromley.co.uk/experiments/soundcloud-vis/#muse/undisclosed-desires

La programmation via l'API WebAudio repose sur la construction d'un graphe modulaire, où chaque module réalise un traitement audio particulier : source sonore, filtrage, distortion, panoramique, mélange (mixer), compression dynamique, restitution sur la sortie audio (casque, HP, ...), cf. figure 1. Cette construction est réalisée dans la partie "script" de la page web.

2 Mise en oeuvre

Pour développer du code côté client (javascript, html, css, ...) vous n'avez normalement pas besoin de faire tourner un serveur web sur votre machine, et il vous suffit simplement de charger votre page localement dans votre navigateur (avec le protocole file :// au lieu du traditionnel http ://). Cependant, quelques fonctionnalités (notamment le chargement de fichiers MP3 distants) nécessitent d'avoir un serveur web. Dans ce cas, vous pouvez installer un petit serveur web de test sur votre machine, par exemple MAMP : http://www.mamp.info/en/mamp_windows.html (ne fonctionne que sur Windows 7 et 8 / attention vous aurez besoin du mot de passe admin).

2.1 Structure d'une page web

Un document web est avant tout structurée par un ensemble de balises HTML (ou plus généralement, XML, où ML=markup language). Un "bloc" est délimité par :

- une balise ouvrante, par exemple <body>
- la balise fermante correspondante, </body>

Deux blocs sont nécessairement, soit inclus l'un dans l'autre, soit complètement disjoints (i.e., vous ne pouvez pas avoir de blocs qui se chevauchent).

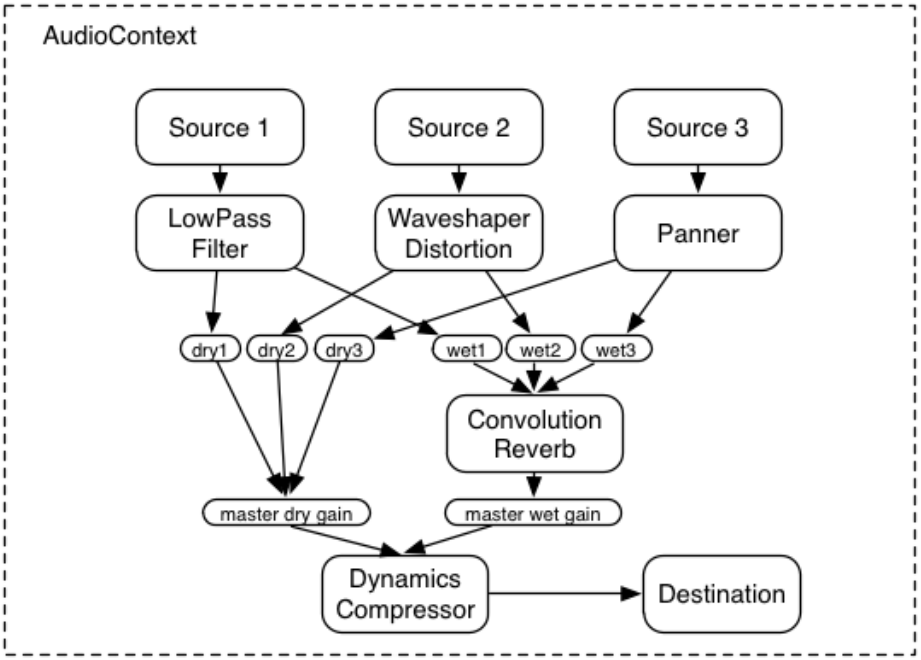

FIGURE 1 – La programmation via l'API WebAudio repose sur la construction d'un graphe modulaire, où chaque module réalise un traitement audio particulier : source sonore, filtrage, distortion, panoramique, mélange (mixer), compression dynamique, restitution sur la sortie audio (casque, HP, ...).

Chaque balise représente une partie de la structure du document : entête (<head>), corps du texte (<body>), titres (<h1>...<h4>), pied de page (<footer>), barre de navigation (<nav>), ... Deux des balises les plus utiles sont , qui permet de délimiter un bloc de texte (paragraphe, tableau, etc.), et , qui permet d'isoler une partie de texte dans un bloc (par exemple, pour le mettre en italique).

Détaillons un premier exemple, figure 2.1. On trouve successivement les blocs

- <html> : bloc principal
- <head> : leader (titre, script, style)
- <style> : spécifications de style au format CSS, cf. `http://www.w3schools.com/css/default.asp` pour la syntaxe détaillée
- <script> : code javascript (vide pour l'instant)
- <body> : corps du texte
- : paragraphe
- <h1> : titre
- <audio> : balise permettant d'inclure un lecteur audio

On trouvera de plus amples détails sur les balises HTML ici : http://www.w3schools.com/html/default.asp.

2.2 AudioContext et code javascript WebAudio

Nous allons enrichir notre page d'un fragment de code javascript permettant de charger un fichier audio et de lui appliquer quelques traitements. Pour cela, nous allons inclure dans le bloc <script> le code de la figure 2.2. Le graphe audio est composé à présent d'une source (créée par l'objet générique "AudioContext"), d'un filtre passe-bas et d'un destination (haut-parleur ou casque). Le fichier MP3 est à présent joué via un filtre passe-bas coupant à 440Hz.

Nous allons enfin ajouter des boutons permettant de modifier les paramètres du filtre, en l'occurence la fréquence de coupure. Pour cela, incluons dans le bloc `<body>` le code de la figure 2.2, et à la fin de la fonction `$(document).ready(function(){...}` le code de la figure 2.2. La syntaxe `$("#plusdaigu").click(function(){filter.frequency.value*=2;})` permet d'affecter une fonction à l'évènement "click sur le bouton possédant l'ID "plusdaigu"). En l'espèce cette fonction multiplie la fréquence du filtre par 2. D'autres paramètres sont modifiables, on en trouvera la liste par exemple ici : http://www.html5rocks.com/en/tutorials/webaudio/intro/.

Enjoy !

```
<!DOCTYPE html> <!-- ceci est un commentaire -->
<html>
<head>
<style> /* specifications de style au format CSS */
body {
        background-color: #DDDDFF;
        font: 15px "Helvetica Neue", Helvetica, Arial, sans-serif;
        text-align: center;
}
div {
        width: 50\%;
        margin: 0px auto;
        border-style: solid;
        border-width: 5px;
        border-radius: 25px;
        border-top-width: 50px;
        min-height: 500px;
        padding: 20px;
}
</style>
<title>Test Web Audio API</title>
<script>
// ici se trouve le code javascript...
</script>
</head>

<body>
<div>
<h1>Chargement d'un son via la balise "audio"</h1>
<audio src="http://goo.gl/9BWH1l" controls></audio>
<br>
Cette page realise le chargement d'un fichier MP3 via la balise
<code> < audio > </code>.
</div>
</body>
</html>
```

FIGURE 2 – Premier exemple de code pour une page web incluant de l'audio (attention, un simple copier-coller de ce code vers votre éditeur donne parfois des résultats étranges...).

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/
3.1.0/jquery.min.js"></script>
<script>
var context, audio, source;
// la fonction suivante est appelee
// lorsque la page est chargee (=ready)
$(document).ready(function(){
        // obtention du contexte audio de la page courante
        window.AudioContext = window.AudioContext
                        || window.webkitAudioContext;
        context = new AudioContext();
        audio = new Audio();
        // creation d'une source a partir du contexte
        source = context.createMediaElementSource(audio);
        // creation d'un filtre passe-bas
        filter = context.createBiquadFilter();
        filter.type = filter.LOWPASS;
        filter.frequency.value = 440;
        // creation du graphe audio source -> filtre -> casque
        source.connect(filter);
        filter.connect(context.destination);
        // choix du fichier MP3 pour la source
        audio.src = 'http://goo.gl/9BWH1l';
        audio.play();
});
</script>
```

FIGURE 3 – Code javascript permettant de créer un embryon de graphe audio. Noter la première ligne, qui permet d'importer la bibliothèque jQuery de gestion d'évènements (http://www.w3schools.com/jquery/default.asp).

```
<button id="plusdaigu">plus d'aigu</button>
<button id="moinsdaigu">moins d'aigu</button>
```

FIGURE 4 – Code HTML pour l'ajout de boutons. Le paramètre "id" permet de référencer le bouton depuis le code javascript.

```
$("#plusdaigu").click(function(){filter.frequency.value*=2;});
$("#moinsdaigu").click(function(){filter.frequency.value*=0.5;});
```

FIGURE 5 – Code Javascript pour gérer les évènements liés aux boutons.