音楽情報処理特論

7回目

今日の予定

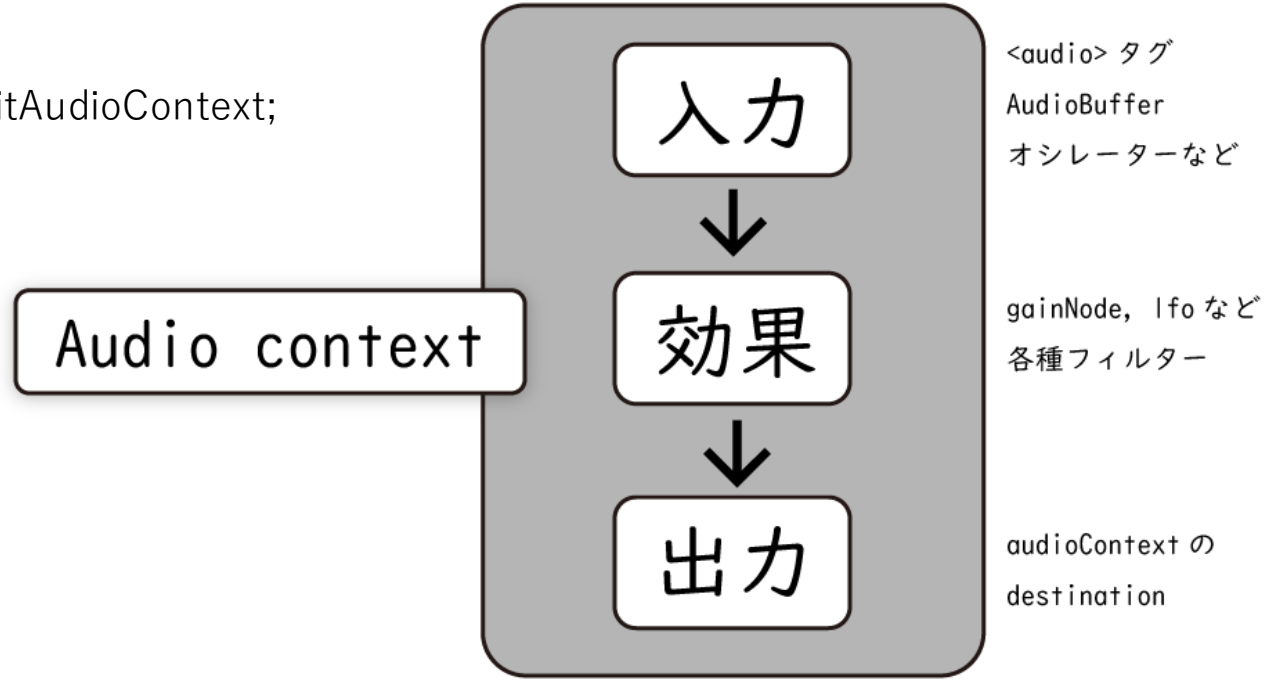
- WebAudioAPIの音出しの基本
- 音をならす
- 音の高さをかえる
- 2つの音でうなりをつくる
- 音色を変える
  - 2つの方法（BufferとOscillator）
- ドレミを作る
- エンベロープ
- ピアノ鍵盤を描く

Web Audio api

- JavaScriptで音をコントロールするapi
- Audio context: 音の入力，効果，出力をコントロール

```
// 最初の書き方
// ブラウザ間の違いをクッション
window.AudioContext = window.AudioContext || window.webkitAudioContext;

// AudioContextのインスタンスを作成
var context = new AudioContext();
```

入力，効果，出力はノード
ノードを定義する
ノードを接続する
入力を再生



オシレーター（振動子）

```
var context = new AudioContext();

// 入力ノード（オシレーター）
var oscillatorNode = context.createOscillator();
// 出力ノード
var destinationNode = context.destination;
//ノードの 接続（入力から出力）
oscillatorNode.connect(destinationNode);

// 再生
oscillatorNode.start(0);
//1000ミリ秒後停止
setTimeout(function(){
        oscillatorNode.stop(0);
}, 1000);
```

効果：音量のコントロール

```
// 音量：gainNode
var gainNode = context.createGain();
var value = 1;     // 音量 0から1
gainNode.gain.value = value;

// 接続：オシレーター => gainNode =>出力
oscillatorNode.connect(gainNode);
gainNode.connect(destinationNode);
```

オシレーター（入力）の設定

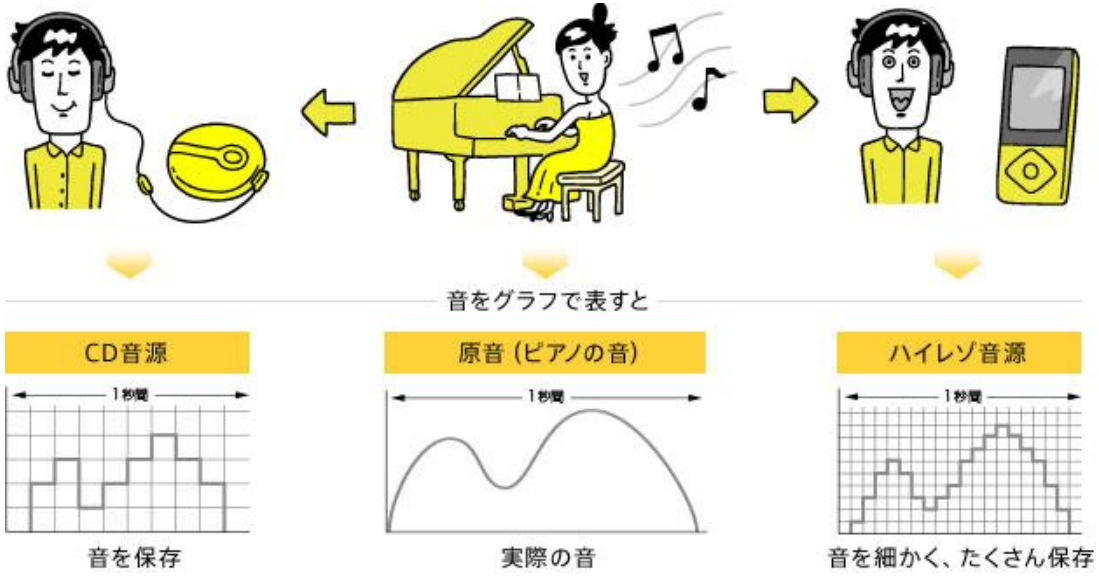
```
// オシレーター：波形生成
var oscillatorNode = context.createOscillator();

// 波形を設定
var type = "sine";              // square, sawtooth, triangleでもOK
oscillatorNode.type = type;

// 周波数を設定
var frequency = 440;        //音高（周波数 Hz）
oscillatorNode.frequency.value = frequency;

// 接続：oscillatorNode => gainNode
oscillatorNode.connect(gainNode);
```

サンプルプログラム

- http://www.is.noda.tus.ac.jp/isws/3IS-javascript/sample_webaudio/play/001_soundPlay_oscillator.html

オーディオデータ

- 波のデータ（アナログ値）をデジタル化
- 標本化
  - 測定周期（サンプリングレート）で値（横軸）をデジタル化
- 量子化
  - 強度（縦軸）を近似してデジタル化



Sony ハイレゾのページより

バッファ（入力）

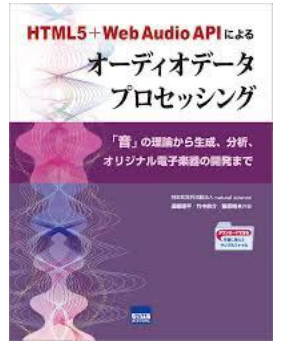
```
// 音源ノード（AudioBufferSourceNode）を生成
var sourceBufferNode = context.createBufferSource();

var sampleRate = context.sampleRate;     // デフォルト値4800
var dt = 1 / sampleRate;                 // オーディオデータの時間刻み
var f = 440;                             // 音源の周波数 [Hz]
var duration = 1;                        // 音源の長さ [s]

// オーディオバッファの作成
var buf = context.createBuffer(
            1,                           // チャンネル数
            sampleRate * duration,       // バッファサイズ（サンプル数）
            sampleRate                   // サンプリングレート
);

// チャンネル番号 0 のオーディオバッファデータ配列
var data = buf.getChannelData(0);
```

```
//データの生成（波形を1秒間描く）
for (var i = 0; i < sampleRate * duration; i++) {
            //時刻の取得
            var t = i * dt;
            // バッファーへデータを格納
            data[i] = Math.sin( 2 * Math.PI * f * t );
}
// 音源ノードのオーディオバッファ設定
sourceBufferNode.buffer = buf;
// 接続
sourceBufferNode.connect( destinationNode );
// 再生
sourceBufferNode.start(0);
//1000ミリ秒後停止
setTimeout(function(){
            sourceBufferNode.stop(0);
}, 1000);
```

サンプルプログラム

- http://www.is.noda.tus.ac.jp/isws/3IS-javascript/sample_webaudio/play/002_soundPlay_buffer.html

参考図書・プログラム



- https://www.cutt.co.jp/book/978-4-87783-375-6.html
- 2015年で少々古い
- 当時のスタイルで書いてある
  - JavaScriptが古い: ES5
  - jQuerryを使っている: javascriptを簡単に書けるようにするライブラリ

  - この本じゃなくてもネット上にたくさん情報がある

- サンプルプログラムを流用します

サンプルプログラム

- 波と波形
  - http://www.is.noda.tus.ac.jp/isws/3IS-javascript/sample_webaudio/play/SoundGenerator.html

- 周波数を変えてみる
  - Math.sin(2 * Math.PI * 660 * t)
- うなり　＝＞差大きくすると差音
  - Math.sin(2 * Math.PI * 440 * t) + Math.sin(2 * Math.PI * 442 * t)
  - Math.sin(2 * Math.PI * 440 * t) + Math.sin(2 * Math.PI * 540 * t)
- 波形
  - (t % (1 / 440) < 1 / 440 / 2) ? 1 : 0　　//矩形波
  - (t % (1 / 440) < 1 / 440 / 2) ? 2 * (t % (1 / 440) / (1 / 440)) : 2 - 2 * (t % (1 / 440) / (1 / 440))　　//三角波
  - t % (1 / 440) / (1 / 440)　// のこぎり波

サンプルプログラム

- 音律
  - http://www.is.noda.tus.ac.jp/isws/3IS-javascript/sample_webaudio/play/DoReMi.html
  - ダウンロードしてドレミを定義し直してみよう

【復習】音をつくる

- 2つの方法
- Bufferに音波の波形を保存する
  - 自由に波形を描ける
    - ピッチ，音色も関数でコントロール

- Oscillatorをつかう
  - 正弦波，矩形波，のこぎり波，三角波があらかじめ用意してある
    - Oscillator nodeの変数で設定

- http://www.is.noda.tus.ac.jp/isws/3IS-javascript/sample_webaudio/play/SoundPlay_buffer_controllable.html
- http://www.is.noda.tus.ac.jp/isws/3IS-javascript/sample_webaudio/play/SoundPlay_osc_controllable.html

うなり

- スライダでピッチをコントロール
- 二つの音を再生
- スライダをうごかして，ハモるおとってわかる？
- http://www.is.noda.tus.ac.jp/isws/3IS-javascript/sample_webaudio/play/SoundPlay_wolf.html

エンベロープ: ADSR

- t = context.currentTime;
  - 現在時間
- gainNode.gain.setValueAtTime(v, t);
  - 時間tに値vにする
- gainNode.gain.linearRampToValueAtTime(v, t);
  - 時間tに値vになるように線形に変化
- gainNode.gain.setTargetAtTime(s, t, d);
  - 時間tになったら値sにむけてdの期間を変化
- gainNode.gain.cancelAndHoldAtTime(t);
  - 時間t以降の変化をキャンセル

http://www.is.noda.tus.ac.jp/isws/3IS-javascript/sample_webaudio/play/SoundPlay_adsr.html

ピアノ鍵盤を描く

- Canvas
- 四角の描き方
- 四角の塗りつぶし方

- http://www.is.noda.tus.ac.jp/isws/3IS-javascript/sample_webaudio/play/keyBord.html

PCのキーボードで演奏する

- KeyのListener
- KeyPress/KeyDown: キーが押されたとき（pressは文字のみ）
- Keyup: キーが離れたとき

来週は自習です

- 課題
  - ステップシーケンサーをつくる
  - ピアノ鍵盤クリック演奏をつくる
- できる範囲でOK

- 提出先：LETUS
- 提出物：htmlやjavascript（動作するモノ一式）
- 提出期限：11/13 13:00まで