Journal of Computations & Modelling, vol.7, no.4, 2017, 111-126
ISSN: 1792-7625 (print), 1792-8850 (online)
Scienpress Ltd, 2017

Shaping the Fade-in Audio Effect with a View to JavaScript Implementation

Lucian Lupşa-Tătaru[1]

Abstract

The present paper puts forward a novel technique of customizing the shape of the fade-in sound effect. In audio engineering, the most common fade-in shapes are of linear, logarithmic and exponential type. Since the valuation of the outputs of the transcendental functions (e.g. exponential and logarithm) is very time consuming, their employment with a view to imposing how the audio level has to change over time is mainly suitable for off-line volume processing, e.g. in various digital audio editors. In this paper, to depict the fade-in shape, we consider that the audio volume is the output of a function of time variable that is neither linear nor transcendental. In order to enable a smooth increasing in the audio level i.e. a smooth transition from silence, without notable "glitches", the employed function is defined so that the initial instantaneous rate of change equals zero. More precisely, without taking into account the linear fade-in shape, we advance a method of fade shaping that is suitable for fast processing in real-time.

[1] Department of Electrical Engineering and Applied Physics, Faculty of Electrical Engineering and Computer Science, Transilvania University of Braşov, Romania.
E-mail: lupsa@programmer.net

Article Info: *Received:* June 29, 2017. *Revised:* August 1, 2017.
Published online : October 31, 2017.

Straightforward implementations in pure JavaScript are provided to the reader with the purpose of immediate use. Discretization is achieved by associating the “timeupdate” HTML DOM media event with the audio element.

Mathematics Subject Classification: 94A12, 94A29, 68N15, 68N19, 97P40
Keywords: Audio volume, Real-time processing, Fade-in audio effect, Fade-in shape, Programming techniques, JavaScript

1 Introduction

A fade-in audio effect designates an increasing of the volume of an audio signal, starting from an audio level of zero (silence) [1-3]. The fade-in effect is applied so that one perceives a smooth lead in to an audio recording i.e. a smooth increasing in the audio level, without notable “glitches” (clicks) [1-5].

The shape of an audio fade-in is determined by the function of time variable that has been selected to provide the audio volume as its output. The well-established fade-in shapes are of linear, logarithmic and exponential type [1-3]. It has to be emphasized here that the well-known S-curve (sine curve) shape, determined by the sine function, is particularly adopted for fading-out and cross-fading between two audio tracks [1, 2]. In order to avoid “glitches” and to perceive a smooth transition from silence [1-5], the fade-in profile has to be customized by minimizing the initial rate of change of the audio volume, occurring at the beginning of the fade-in effect. In this context, the employing of the linear fade-in is not appropriate having in view that, in this case, minimizing the rate of change of the audio level is equivalent to maximizing the fade-in length. The fade-in length, i.e. the time interval over which the audio volume is being processed, has to be correlated with the detected music genre [6-8]. Anyhow, short fade-ins, less than 0.5 s, can be useful just to soften the attack of a certain audio region [1, 2].

In the present paper, in order to customize the shape of the fade-in audio effect, we consider that the audio level is represented by the output of a rational function that incorporates two coefficients, which are to be computed by setting the audio level at different instants of time.

Although it implies real-time processing, the connection with JavaScript programming language is entirely justified here having in view that the computing of the output of a rational function is equivalent to valuating algebraic fractions and that the discretization can straightforwardly be achieved by linking the "timeupdate" HTML DOM event to the audio element. Moreover, audio volume processing in HTML5/JavaScript is straightforward: the input of the employed rational function i.e. the time variable is returned by the "currentTime" property of HTML DOM Audio Object while the output of the adopted function i.e. the audio level is used to set the Audio Object "volume" property [9-11].

2 The Fade-in Curve

As aforementioned, the audio fade-ins are usually performed by employing linear or transcendental functions (e.g. exponential and logarithm) to set the manner of time-related increasing of the audio level, starting from zero (silence). However, the use of transcendental functions to customize the fade-in shape is mostly suitable for off-line mode, e.g. in several digital audio editors, and not for real-time volume processing, when the computational capabilities are of utmost significance.

The time-related variation of the audio volume is depicted here by means of a function of time, which is neither linear nor transcendental. More precisely, we advance a rational function, i.e. a function defined by an algebraic fraction, to customize the shape of the fade-in effect. Thus, we assume that the audio volume is the output of the following function of time variable:

$$v(t)=\frac{\alpha t^2}{t+\beta}\ ,\quad t\in\left[0,t_f\right] \tag{1}$$

wherein α and β are coefficients that will be computed by setting the audio level at different instants whilst t_f is the length of the fade effect.

The instantaneous rate of change of the audio volume is provided by the time-related derivative of function (1) that is:

$$\frac{dv}{dt}=\alpha\,\frac{t+2\beta}{(t+\beta)^2}\,t\ ,\quad t\in\left[0,t_f\right]. \tag{2}$$

Taking into account that, based on (1) and (2), we have

$$v(0)=0,\quad \left.\frac{dv}{dt}\right|_{t=0}=0\ , \tag{3}$$

it follows that the employing of rational function (1) to shape the fade-in effect ensures a smooth transition from silence, without notable “glitches” (clicks).

The coefficients α and β in (1) allow the fitting of the fade-in profile in accordance with the user requirements. One proceeds to compute these coefficients by setting the volume of the audio signal both at the end of the fade effect and at an intermediate instant of time t_{in} during the fading-in. Thus:

$$\begin{cases} v(t_f)=VOL_f \\ v(t_{in})=VOL_{in} \end{cases} \tag{4}$$

wherein VOL_f and VOL_{in} are the imposed audio levels. Having in view that the audio volume is precisely the output of (1), one obtains:

$$\begin{cases} \dfrac{\alpha t_f^2}{t_f+\beta}=VOL_f \\ \dfrac{\alpha t_{in}^2}{t_{in}+\beta}=VOL_{in} \end{cases} \tag{5}$$

System (5) is linear with respect to coefficients α and β i.e.

$$\begin{cases} t_f^2\cdot\alpha-VOL_f\cdot\beta=VOL_f\cdot t_f \\ t_{in}^2\cdot\alpha-VOL_{in}\cdot\beta=VOL_{in}\cdot t_{in} \end{cases} \tag{6}$$

The solution to the linear system (6) in α and β is given by

$$\alpha = \frac{VOL_{in} \cdot VOL_f}{VOL_{in} \cdot t_f^2 - VOL_f \cdot t_{in}^2} \left(t_f - t_{in} \right), \tag{7}$$

$$\beta = \frac{VOL_f \cdot t_{in} - VOL_{in} \cdot t_f}{VOL_{in} \cdot t_f^2 - VOL_f \cdot t_{in}^2} \, t_{in} t_f \, . \tag{8}$$

Expressions (7) and (8) allow the shaping of the fade-in effect in accordance with the values selected for the fade length t_f, the audio level to be reached at the end of fade-in i.e. VOL_f, the intermediate instant t_{in} and its corresponding audio level VOL_{in}. In practice, the intermediate instant of time is usually represented by the halfway point i.e. the fade-in midpoint [1, 2]. Thus, we consider:

$$t_{in} = t_f / 2 \, . \tag{9}$$

In this case, the coefficients in (1) come to be dependencies just on the fade length t_f, the audio level at the midpoint i.e. VOL_{in}, and the audio level to be reached that is VOL_f. More precisely, based on (7)-(9), we have:

$$\alpha = \frac{VOL_{in} \cdot VOL_f}{4 \cdot VOL_{in} - VOL_f} \frac{2}{t_f} = \frac{v\left(\frac{t_f}{2}\right) \cdot v\left(t_f\right)}{4 \cdot v\left(\frac{t_f}{2}\right) - v\left(t_f\right)} \frac{2}{t_f}, \tag{10}$$

$$\beta = \frac{VOL_f - 2 \cdot VOL_{in}}{4 \cdot VOL_{in} - VOL_f} t_f = \frac{v\left(t_f\right) - 2 \cdot v\left(\frac{t_f}{2}\right)}{4 \cdot v\left(\frac{t_f}{2}\right) - v\left(t_f\right)} \, t_f \, . \tag{11}$$

We consider $\alpha > 0$ and $\beta > 0$, respectively. In this context, having in view expressions (10) and (11), one receives:

$$\begin{cases} 4 \cdot v\left(\frac{t_f}{2}\right) - v\left(t_f\right) > 0 \\ v\left(t_f\right) - 2 \cdot v\left(\frac{t_f}{2}\right) > 0 \end{cases}$$

i.e.

$$\frac{1}{4} v\left(t_f\right) < v\left(\frac{t_f}{2}\right) < \frac{1}{2} v\left(t_f\right). \tag{12}$$

Taking into account expression (2) of the time-related derivative of function (1), condition (12) leads to

$$\frac{dv}{dt} > 0, \quad t \in \left(0, t_f\right]$$

since both $\alpha > 0$ and $\beta > 0$. Hence, with (12), the rate of change of audio volume is of positive sign, thus acting in the direction of increasing the audio level.

To avoid “glitches” and for the sake of simplicity, we plainly take

$$v\left(\frac{t_f}{2}\right) = \frac{1}{3} VOL_f = \frac{1}{3} v\left(t_f\right). \tag{13}$$

Having in view (13), one perceives that the resulted fade-in effect will act similar to a fade-in of exponential shape i.e. the audio volume will increase slowly till the halfway point and, then, it will go up quickly with an increasing rate of change [1-3]. Moreover, with (13), the coefficients α and β in (1), shaping the fade-in, become expressions in terms of only fade length t_f and the audio level to be reached i.e. VOL_f. Based on (10), (11) and (13), we have:

$$\alpha = 2 \frac{VOL_f}{t_f} = 2 \frac{v\left(t_f\right)}{t_f}, \quad \beta = t_f \,. \tag{14}$$

With a view to JavaScript implementation, the output of function (1) has to be located within interval [0, 1], with the mention that the output value of 1 designates the highest volume [9, 10]. We consider here that the volume to be

reached at the end of fade-in is:

$$VOL_f = v\left(t_f\right) = 0.9 . \tag{15}$$

With (15), both coefficients (14), interfering in (1), come to be dependencies on fade-in length only:

$$\alpha = 1.8 / t_f , \quad \beta = t_f . \tag{16}$$

In this context, Figure 1 highlights the fade-in curves i.e. the outputs of (1) for fade lengths $t_f = 0.5\,\mathrm{s}$, $t_f = 1\,\mathrm{s}$ and $t_f = 2\,\mathrm{s}$, respectively. For the same set of fade lengths, Figure 2 illustrates the rate of change of audio volume i.e. the values of (2).

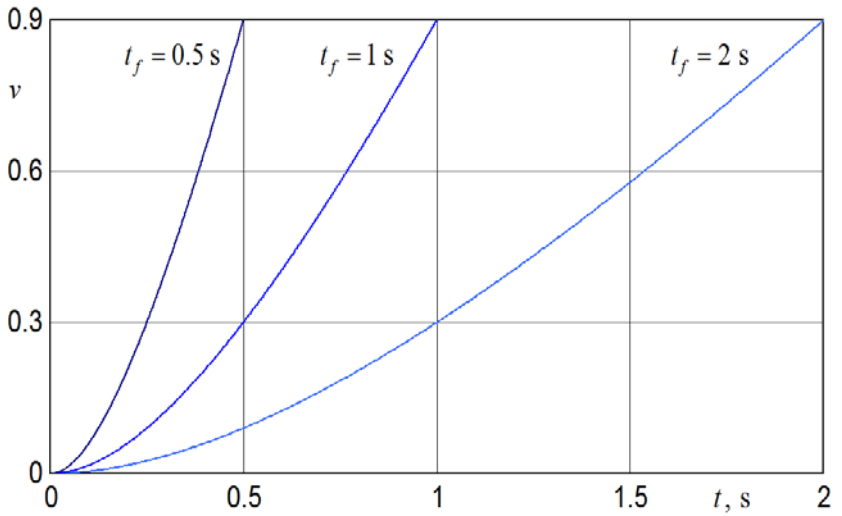

Figure 1: Fade-in curves for fade lengths $t_f = 0.5\,\mathrm{s}$, $t_f = 1\,\mathrm{s}$ and $t_f = 2\,\mathrm{s}$

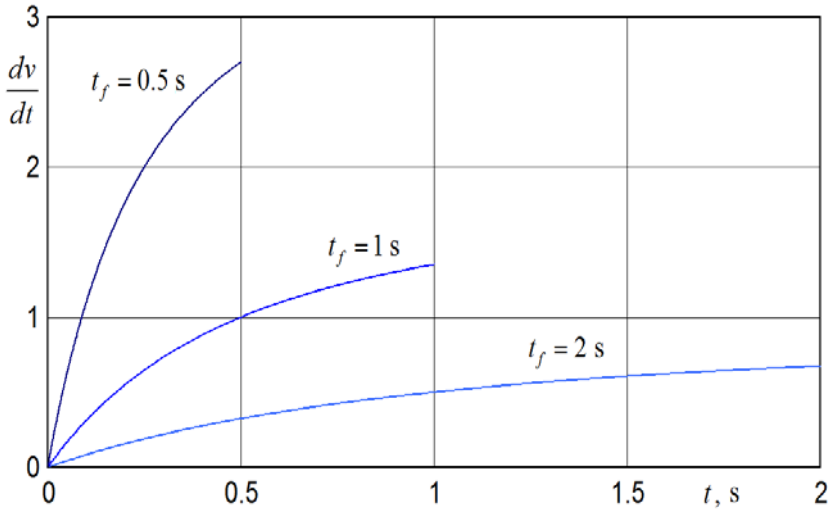

Figure 2: Instantaneous rate of change of audio volume for fade lengths $t_f = 0.5\,\text{s}$, $t_f = 1\,\text{s}$ and $t_f = 2\,\text{s}$

3 JavaScript Implementations

The applications put forward in this paper have been developed by considering the audio volume to be reached of value $VOL_f = 0.9$ and the enhanced fade-in effect of length $t_f = 8\,\text{s}$. It has to be pointed out that the applications encompassed by the paper have been optimized so that the user can easily change the fade-in length and, then, the coefficients of (1) will be computed straightforwardly, according to (16).

In this context, Figure 3 highlights both the fade-in curve i.e. the output of (1) and the instantaneous rate of change of audio volume i.e. the time-related derivative (2).

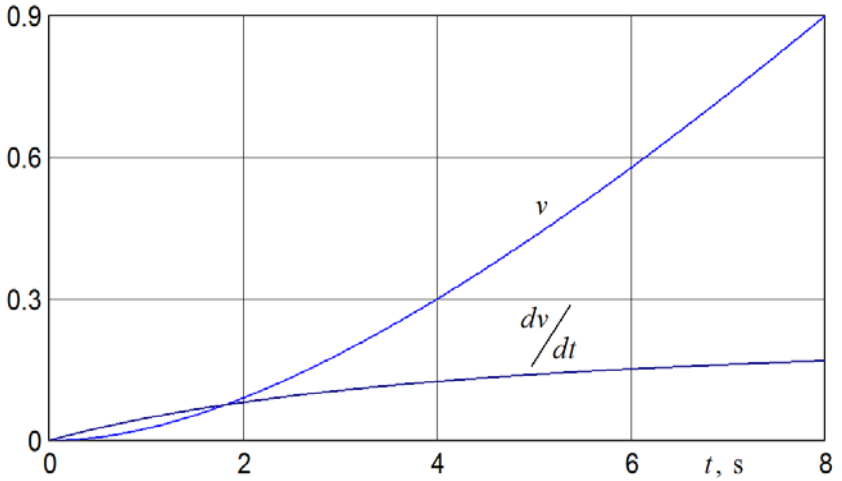

Figure 3: Fade-in curve and the rate of change of audio volume for the enhanced fade-in effect of length $t_f = 8\,\mathrm{s}$

We also emphasize that the audio volume, provided here by (1), wherein the coefficients are given by (16), is to be set by using the "volume" property of HTML DOM Audio Object. With the purpose of discretization, one has to associate a "timeupdate" media event with the audio element. Thus, the discrete-time processing will require the use of the "currentTime" property of the Audio Object, which returns the position of the audio playback [9, 10].

In the following application, the audio is loaded with the page, and the playback starts as soon as the audio is loaded. The fade-in effect i.e. the audio volume processing is automatically applied from the beginning of the audio until the output of function (1) is greater than the audio level to be reached. One perceives that the "fade_in" global variable is used here just to skip volume processing if the value returned by (1) has been found greater than the value $VOL_f = 0.9$, selected for the volume to be reached at the end of fade-in. The input of rational function (1) i.e. the current playback time is provided by the Audio Object "currentTime" property. The code that implements the described application is given next.

```
<!DOCTYPE html>
<html>
<head>
    <title>Fade-in</title>
</head>
<body>
<script>

    // create the audio element
    var ae = document.createElement( “AUDIO” );
    ae.preload = “auto”;       // the audio is loaded with the page
    ae.controls = true;

    // specify the audio file
    ae.src = “sample.mp3”;  // audio/mpeg

    // associate a timeupdate event with the audio element
    ae.addEventListener( “timeupdate”, setVol );

    document.body.appendChild( ae );

    var tf = 8.0;                  // fade-in length, in second
    // compute coefficients for the fade length of 8 s
    var alpha = 1.8 / tf;
    var beta = tf;

    var fade_in = true;
    ae.currentTime = 0; ae.volume = 0;       // sets properties
    ae.play();
```

```
    function v( t ) {
        var retVol = alpha * t * t / ( t + beta );
        return retVol;
    }

    function setVol() {
        if ( fade_in ) {
            var finalVol = 0.9;                 // audio volume to be reached
            var currentVol = v( ae.currentTime );
            if ( currentVol <= finalVol ) {
                ae.volume = currentVol;
            }
            else {
                ae.volume = finalVol;     // volume suppressing
                fade_in = false;
            }
        }
    }

</script>
</body>
</html>
```

More complex applications could be developed by linking multiple HTML DOM media events to the audio element, as performed within the implementation that follows.

```
<!DOCTYPE html>
```

```
<html>
<head>
    <title>Fade-in</title>
</head>
<body>
<script>

    // create the audio element
    var ae = document.createElement( “AUDIO” );
    ae.preload = “auto”;       // the audio is loaded with the page
    ae.controls = true;

    // specify the audio file
    ae.src = “sample.mp3”;  // audio/mpeg

    // associate a play event with the audio element
    ae.addEventListener( “play”, init );

    // associate a timeupdate event with the audio element
    ae.addEventListener( “timeupdate”, setVol );

    document.body.appendChild( ae );

    var tf = 8.0;                  // fade-in length, in second
    // compute coefficients for the fade length of 8 s
    var alpha = 1.8 / tf;
    var beta = tf;

    var fade_in, referenceTime;
```

```
    function v( t ) {
        var retVol = alpha * t * t / ( t + beta );
        return retVol;
    }

    function init() {
        fade_in = true;
        referenceTime = ae.currentTime;   // starting time
        ae.volume = 0;
    }

    function setVol() {
        if ( fade_in ) {
            var finalVol = 0.9;              // audio volume to be reached

            var deltaT = ae.currentTime - referenceTime;
            currentVol = v( deltaT );
            if ( currentVol <= finalVol ) {
                ae.volume = currentVol;
            }
            else {
                ae.volume = finalVol;     // volume suppressing
                fade_in = false;
            }
        }
    }

</script>
```

```
</body>
</html>
```

One observes that, in this case, alongside the “timeupdate” media event, a “play” media event has been associated with the audio element in order to allow the applying of the fade-in effect whenever the user has started playing the audio i.e. when the audio is no longer paused. More precisely, in contrast to the previous application, the playback does not start immediately after the audio is loaded and the fade-in effect is not initiated only at the beginning of the audio but also whenever the audio is no more paused. We have introduced the “referenceTime” global variable to hold the position of the audio playback each time the audio is started. Thus, the volume processing is performed according to formula (1) but with respect to the current value t_{ref} of “referenceTime” variable i.e. according to the relation:

$$v(\Delta t)=\frac{\alpha(\Delta t)^2}{\Delta t+\beta}\ ,\quad \Delta t\in[0,t_f] \tag{17}$$

wherein

$$\Delta t=t-t_{ref}\ .$$

4 Conclusion

The shape of the audio fade-ins [1, 2] is usually customized in off-line mode, e.g. in various digital audio editors, by making use of different transcendental functions in order to set the time-related evolution of the audio level starting from zero (silence). In the present paper, to construct the fade-in shape, we consider that the audio volume is the output of a specific rational function. Having in view that the valuation of the outputs of rational functions is equivalent to computing values of algebraic fractions, such kind of approach comes to be suitable for real-time

volume processing.

To make feasible a smooth increasing in the audio level that is a smooth transition from silence, the employed rational function has been defined so that both its initial output i.e. the initial audio volume and the corresponding initial instantaneous rate of change, occurring at the beginning of fade-in, have a value of zero. To allow the fitting of the fade-in profile, two coefficients have been introduced. The values of these coefficients have been computed by setting the audio level at the fade-in midpoint and at the end of the fade effect, taking into account that, with a view to JavaScript implementation [9-11], the audio volume, which becomes a property of HTML DOM Audio Object, has to be located within interval [0, 1]. The process of discretization has been accomplished by linking a "timeupdate" HTML DOM media event to the audio element.

The optimized implementations, encompassed by the paper, and prepared for immediate utilization, emphasize the efficiency of the suggested technique of customizing the shape of the fade-in audio effect. By simply pointing to different audio files and by varying the fade-in length, one could perceive that the increasing in the audio level is smooth enough, without notable "glitches" (clicks).

References

[1] S. Langford, *Digital Audio Editing: Correcting and Enhancing Audio in Pro Tools, Logic Pro, Cubase, and Studio One*, Focal Press, Burlington, MA, USA, 2014.

[2] A.U. Case, *Sound FX: Unlocking the Creative Potential of Recording Studio Effects*, Focal Press, Burlington, MA, USA, 2007.

[3] J.D. Reiss and A. McPherson, *Audio Effects: Theory, Implementation and Application*, CRC Press, Boca Raton, FL, USA, 2015.

[4] D. Creasey, *Audio Processes: Musical Analysis, Modification, Synthesis, and Control*, Routledge, New York, NY, USA, 2017.

[5] J. Corey, *Audio Production and Critical Listening: Technical Ear Training*, 2nd edition, Routledge, New York, NY, USA, 2017.

[6] Y. Panagakis, C.L. Kotropoulos and G.R. Arce, Music genre classification via joint sparse low-rank representation of audio features, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, **22**(12), (December 2014), 1905-1917.

[7] E. Benetos and C. Kotropoulos, Non-negative tensor factorization applied to music genre classification, *IEEE Transactions on Audio, Speech, and Language Processing*, **18**(8), (November 2010), 1955-1967.

[8] Y. Panagakis, C. Kotropoulos and G.R. Arce, Non-negative multilinear principal component analysis of auditory temporal modulations for music genre classification, *IEEE Transactions on Audio, Speech, and Language Processing*, **18**(3), (March 2010), 576-588.

[9] I. Devlin, *HTML5 Multimedia: Develop and Design*, Peachpit Press, Berkeley, CA, USA, 2012.

[10] S. Powers, *HTML5 Media*, O'Reilly Media, Sebastopol, CA, USA, 2011.

[11] I. Jacobs, J. Jaffe and P. Le Hegaret, How the open web platform is transforming industry, *IEEE Internet Computing*, **16**(6), (November-December 2012), 82-86.