Applied Computer Science, vol. 15, no. 2, pp. 5–18
doi:10.23743/acs-2019-09

Submitted: 2019-04-02
Revised: 2019-04-23
Accepted: 2019-04-25

audio effects, audio fade-in, real-time processing, HTML5, web apps

Lucian LUPŞA-TĂTARU [0000-0002-3320-9850]*

IMPLEMENTING THE FADE-IN AUDIO EFFECT FOR REAL-TIME COMPUTING

Abstract

Audio fading is performed in order to smoothly modify over time the level of an audio signal. In particular, the fade-in audio effect designates a gradually increase in the audio volume, starting from silence. In practice, audio fading is mostly carried out within audio editors i.e. in off-line mode by employing various transcendental functions to enforce the fade profile. Taking into account the increasing demand for interactive media services requiring real-time audio processing, the present approach advances an effective method of constructing the audio fade-in shape with a view to real-time computing. The paper encompasses plain and straightforward implementations in pure JavaScript, prepared precisely to validate the method of audio volume processing proposed here.

1. INTRODUCTION

The nature of interactive computing, which is essential for a two-way effective communication between machine and user, implies fast computing and employment of event-driven programming style. On the other hand, with the recent release of the fifth version (HTML5) of HTML standard, which natively allows the control of multimedia content, complex web applications have arisen to complement the traditional native applications on the various low-powered devices. Also, the functionality of the new set of rules characteristic of HTML5 appears to be very convenient to enhance by event-driven programming carried out in JavaScript (Devlin, 2012; Jacobs, Jaffe & Le Hegaret, 2012; Powers, 2011; *HTML 5.2. W3C Recommendation*, 2017).

* Transilvania University of Braşov, Faculty of Electrical Engineering and Computer Science, Department of Electrical Engineering and Applied Physics, Bd. Eroilor No. 29, Braşov, RO-500036, Romania, lupsa@programmer.net, lucian.lupsa@unitbv.ro

In this context, having in view that designing of media development software, simulation software as well as entertainment applications often asks for real-time/fast audio processing, the present investigation puts forward an effective solution for constructing the profile of the fade-in audio effect. Although widely used in order to receive a smooth lead in to an audio content, the fade-in sound effect is usually applied within audio editors that is in off-line mode, by employing linear and transcendental functions to impose the time-related evolution of the audio volume, starting from a level of 0 (silence) (Case, 2007; Jackson, 2015; Langford, 2014; Reiss & McPherson, 2015).

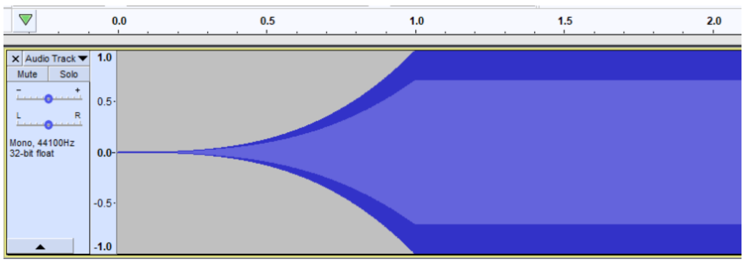

Fig. 1. An exponential fade-in audio effect, received in Audacity editor for the final audio level of 1 and the fade length of 1 s

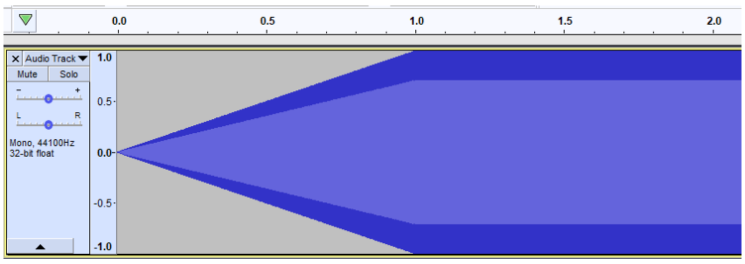

Fig. 2. A linear fade-in audio effect, received in Audacity editor for the final audio level of 1 and the fade length of 1 s

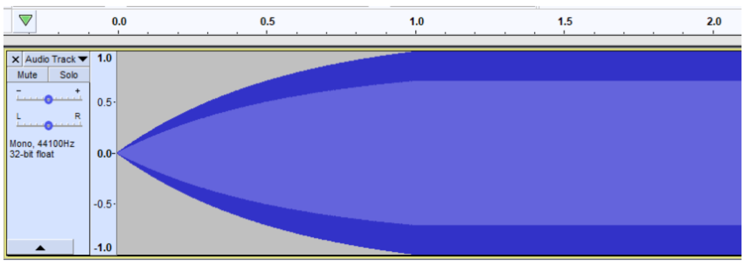

Fig. 3. A logarithmic fade-in audio effect, received in Audacity editor for the final audio level of 1 and the fade length of 1 s

Fig. 1, Fig. 2 and Fig. 3 highlight a set of 1 s length fade-ins of exponential, linear, and logarithmic curve shape, carried out in Audacity editor (Jackson, 2015; Schroder, 2011). It has to be emphasized that both fade-in length and shape should be correlated with the appropriate music genre (Corey, 2017; Panagakis, Kotropoulos & Arce, 2014; Potter, 2002).

By adopting the exponential fade-in, depicted in Fig. 1, one receives an extremely smooth transition from silence, characterized by a low instantaneous rate of change of audio level in the beginning of fading-in. As the opposite of the exponential fade-in shape, the fade-in of logarithmic type, illustrated in Fig. 3, determines a quick increase of the audio volume in the beginning of the fade effect, followed by a decrease in the rate of change of audio volume towards the end of fading-in. Hence, a logarithmic fade-in brings a soft increase in the audio level within the ending region of the effect. On the other hand, as Fig. 2 indicates, the linear fade-in effect is performed at a constant rate of change of audio level and, obviously, in this case, any attempt of smoothing the transition from silence would lead to an increasing of the fade-in length (Case, 2007; Langford, 2014; Reiss & McPherson, 2015).

With a view to real-time computing, one has to consider that the instant of fade-in initiation is associated with an event occurrence, and that the valuation of the outputs of transcendental functions, suitable for shaping the fade-in profile in the off-line mode, comes to be very time consuming.

2. CUSTOMIZING THE FADE-IN PROFILE

To improve the computational capabilities with the purpose of real-time processing, we consider that the evolution of audio volume during fading-in is given here by the output of the following rational function of time variable:

$$\begin{cases} v(t) = \frac{\alpha t^k}{t+\beta}, \quad t \in [0, t_f] \\ k \in \{1,2,3\} \end{cases} \tag{1}$$

where: t_f – designates the fade-in length.

In contrast to valuating the outputs of different transcendental functions, which proved to be suitable for customizing the audio fade-in shape in the off-line mode, e.g. within various audio editors, the implementation of (1) does not require for auxiliary functions or methods to be called, regardless of the adopted programming language. It has to be noticed that the approach corresponding precisely to $k = 2$ in (1) has already been considered in order to construct fade-ins that act, in real-time, similar to the fade-in audio effect of exponential curve shape (Lupsa-Tataru, 2017). Nevertheless, it will be shown that the fade-ins received by computing the more comprehensive expression (1) can act either as the fade-in audio effect of exponential type or as the fade-in audio effect of logarithmic curve shape.

With the time-related audio level provided by (1), the instantaneous rate of change of the audio volume during fading-in comes to be:

$$\begin{aligned} v'(t) &\equiv \frac{dv}{dt} = \frac{d}{dt}\left(\frac{\alpha t^k}{t+\beta}\right) \\ &= \alpha[(k-1)t + k\beta]\frac{t^{k-1}}{(t+\beta)^2} \end{aligned} . \qquad (2)$$

Since the fade-in audio effect is represented by a strict increasing in the signal level, it plainly follows that function (1) has to be strictly increasing (Langford, 2014; Reiss & McPherson, 2015). Having in view (2), this implies:

$$\alpha[(k-1)t + k\beta] > 0. \qquad (3)$$

On the other hand, if we denote the fade-in halfway point i.e. the fade-in midpoint by means of variable

$$t_h = t_f/2, \qquad (4)$$

then, in order to receive a strict increasing in the level of the audio signal, starting from silence, the following conditions come to be essential:

$$v(t_h) = v_h = \rho_h v_f \,, \qquad (5)$$

$$v(t_f) = v_f \,, \qquad (6)$$

wherein, we have

$$v(0) = 0 < v_h < v_f \,, \qquad (7)$$

$$\rho_h = v_h/v_f \,. \qquad (8)$$

Within relations (5)–(8), variable v_h indicates the imposed audio level at the fade-in midpoint (4) whilst variable v_f designates the final volume i.e. the audio level to be reached at the end of fading-in. In this context, taking now into account merely (7) and (8), one obtains

$$0 < \rho_h < 1. \qquad (9)$$

Having in view that the function employed to construct the fade-in profile is given by (1), conditions (5) and (6) lead to the following system of algebraic equations:

$$\begin{cases} \frac{\alpha t_f^k}{t_f+2\beta} = 2^{k-1}\rho_h v_f \\ \frac{\alpha t_f^k}{t_f+\beta} = v_f \end{cases} . \tag{10}$$

One perceives that (10) is linear with respect to the encompassed coefficients α and β. More precisely, one straightforwardly obtains that (10) is equivalent to:

$$\begin{cases} t_f^k \cdot \alpha - 2^k \rho_h v_f \cdot \beta = 2^{k-1}\rho_h t_f v_f \\ t_f^k \cdot \alpha - v_f \cdot \beta = t_f v_f \end{cases} . \tag{11}$$

For a specific value of k in (1), the solution of (11) yields the coefficients of (1) in terms of fade length t_f, the final volume v_f, and quantity (8) that depends on the imposed audio level at the fade-in halfway point (4). One receives:

$$\alpha = \frac{2^{k-1}\rho_h v_f}{2^k\rho_h - 1} \cdot \frac{1}{t_f^{k-1}} , \tag{12}$$

$$\beta = \frac{1-2^{k-1}\rho_h}{2^k\rho_h - 1} t_f . \tag{13}$$

In order to fulfill condition (3) of enforcing a positive rate of change of the audio level, we plainly consider $\alpha > 0$ and $\beta > 0$, respectively. In this context, by inspecting expressions (12) and (13) of the two coefficients, the following conditions arise:

$$\begin{cases} 2^k\rho_h - 1 > 0 \\ 1 - 2^{k-1}\rho_h > 0 \end{cases} \tag{14}$$

or, what is just equivalent

$$\frac{1}{2^k} < \rho_h < \frac{1}{2^{k-1}} . \tag{15}$$

Thus, based on (12), (13) and (15), one finds that the time-related expression of the algebraic fraction (1), which shapes the audio fade-in profile, incorporates the parameters:

$$k = \begin{cases} 3, & 1/8 < \rho_h < 1/4 \\ 2, & 1/4 < \rho_h < 1/2, \\ 1, & 1/2 < \rho_h < 1 \end{cases} \tag{16}$$

$$\alpha = \alpha(\rho_h) = \begin{cases} \dfrac{4\rho_h v_f}{8\rho_h - 1}\dfrac{1}{t_f^2}, & 1/8 < \rho_h < 1/4 \\ \dfrac{2\rho_h v_f}{4\rho_h - 1}\dfrac{1}{t_f}, & 1/4 < \rho_h < 1/2, \\ \dfrac{\rho_h v_f}{2\rho_h - 1}, & 1/2 < \rho_h < 1 \end{cases} \tag{17}$$

$$\beta = \beta(\rho_h) = \begin{cases} \dfrac{1 - 4\rho_h}{8\rho_h - 1} t_f, & 1/8 < \rho_h < 1/4 \\ \dfrac{1 - 2\rho_h}{4\rho_h - 1} t_f, & 1/4 < \rho_h < 1/2, \\ \dfrac{1 - \rho_h}{2\rho_h - 1} t_f, & 1/2 < \rho_h < 1 \end{cases} \tag{18}$$

wherein one observes that the value of k out of the set $\{1,2,3\}$ as well as the appropriate expressions of the coefficients are decided by the value of quantity (8) that is the ratio between the imposed audio level at the fade-in midpoint and the final audio level.

Since both coefficients (17) and (18), interfering in (1), are positive and, also, $k \in \{1,2,3\}$, it follows that condition (3) of receiving a gradual increasing in the level of the audio content is now fulfilled. Hence, with (16)–(18), one obtains:

$$v'(t) \equiv \tfrac{dv}{dt} > 0, \quad t > 0 \tag{19}$$

and, having in view (2),

$$v'(0) \equiv \left.\tfrac{dv}{dt}\right|_{t=0} = \begin{cases} 0, & 1/8 < \rho_h < 1/4 \\ 0, & 1/4 < \rho_h < 1/2. \\ \alpha/\beta\,, & 1/2 < \rho_h < 1 \end{cases} \tag{20}$$

Considering (1), (2) and (16), then, for $1/8 < \rho_h < 1/4$, one explicitly gets the appropriate expression of the rational function shaping the fade-in profile along with the appropriate expression of the instantaneous rate of change of the audio volume during fading-in i.e.

$$v(t) = \frac{\alpha t^3}{t+\beta}, \quad t \in \left[0, t_f\right], \tag{21}$$

$$v'(t) \equiv \frac{dv}{dt} = \alpha(2t + 3\beta)\frac{t^2}{(t+\beta)^2} > 0, \quad t > 0. \tag{22}$$

Similarly, for $1/4 < \rho_h < 1/2$, one explicitly receives

$$v(t) = \frac{\alpha t^2}{t+\beta}, \quad t \in \left[0, t_f\right], \tag{23}$$

$$v'(t) \equiv \frac{dv}{dt} = \alpha(t + 2\beta)\frac{t}{(t+\beta)^2} > 0, \quad t > 0. \tag{24}$$

while for $1/2 < \rho_h < 1$, the rational function (1), modelling the fade-in profile, and the corresponding rate of change of audio level, yielded by (2), come to be

$$v(t) = \frac{\alpha t}{t+\beta}, \quad t \in \left[0, t_f\right], \tag{25}$$

$$v'(t) \equiv \frac{dv}{dt} = \frac{\alpha\beta}{(t+\beta)^2} > 0, \quad t \in \left[0, t_f\right]. \tag{26}$$

3. THE FADE-IN CURVES

Based on (1), (16)–(18), in Fig. 4, Fig. 5, and Fig. 6, we have illustrated the fade-in profiles for fade length $t_f = 0.5$ s, $t_f = 1$ s and $t_f = 2$ s, respectively, with quantity (8) selected as parameter. Since the validation of the suggested technique of real-time performing the fade-in audio effect is accomplished here by means of plain JavaScript implementations, we have considered that the final volume i.e. the volume to be reached at the end of fading-in has the value of 1 that is the default and, also, the highest volume adopted in HTML5 (Devlin, 2012; Powers, 2011; *HTML 5.2. W3C Recommendation*, 2017). In this context, quantity (8) represents just the audio level imposed at the fades midpoint (4) i.e. $t_h = 0.25$ s for Fig. 4, $t_h = 0.5$ s for Fig. 5 and $t_h = 1$ s for Fig. 6, respectively.

One observes that the fade-ins of Fig. 4, Fig. 5 and Fig. 6, respectively, received for $\rho_h = 0.15$ and $\rho_h = 0.3$, will act similar to the fade-in audio effect of exponential curve shape i.e. the audio level will increase smoothly till the midpoint, starting from an initial rate of change of zero, as (20) indicates, and, then, it will slope upwards with an increasing rate of change, according to (21) and (23), respectively. At the opposite side, the fade-ins corresponding to $\rho_h = 0.7$ and $\rho_h = 0.85$ will act similar to the fade-in of logarithmic shape i.e. the audio level will go up quickly till the halfway point, and, subsequently, it will increase slowly towards the end of fading-in, in accordance with the time-related evolution enforced by (25).

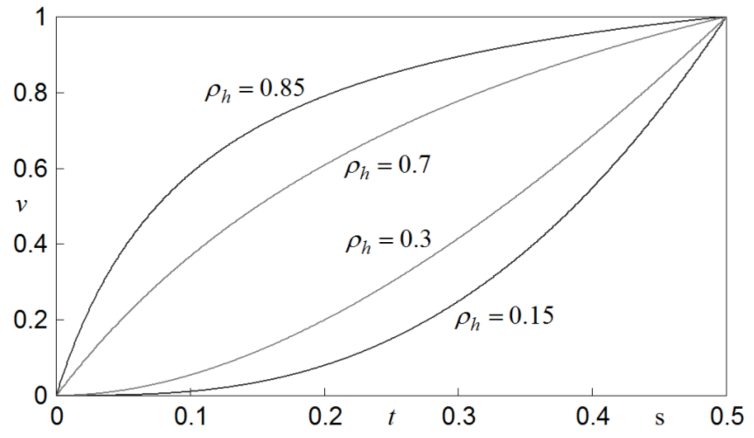

Fig. 4. Fade-in curves for the fade length of 0.5 s and the final volume of 1 (the default value in HTML5)

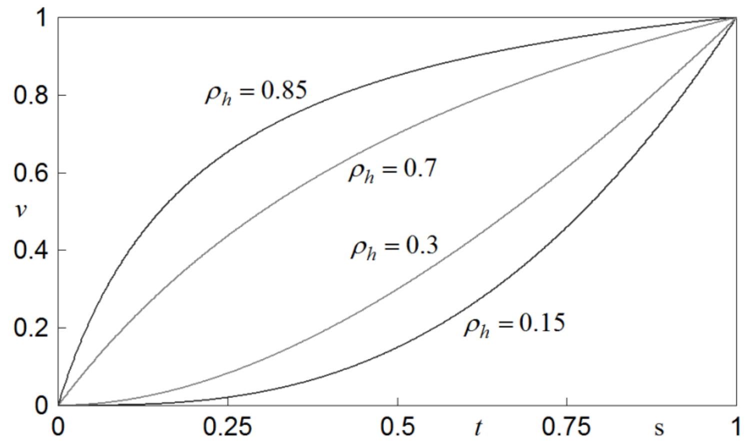

Fig. 5. Fade-in curves for the fade length of 1 s and the final volume of 1 (the default value in HTML5)

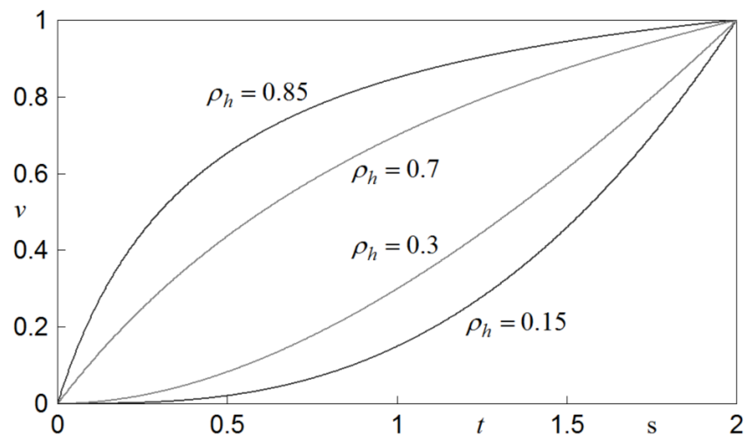

Fig. 6. Fade-in curves for the fade length of 2 s and the final volume of 1 (the default value in HTML5)

Thus, all the three figures, plotted for $\rho_h \in \{0.15, 0.3, 0.7, 0.85\}$, do highlight that the received profiles are crucially decided by the value of quantity (8), regardless of the fade-in length, what validates the generality of function (1), employed for fade shaping.

4. JAVASCRIPT IMPLEMENTATIONS

In order to verify the suggested method of fade-in profile customizing for the suitability with real-time computing, we have developed straightforward implementations in plain (“vanilla”) JavaScript, with the discretization being achieved by employing the “setInterval()” method of the “window” object. For the sake of simplicity, the two applications advanced in the paper have been optimized so that the only intervention of the user consists in providing the appropriate audio content (sample) upon which the fade-in effect has to be applied.

The first application has been implemented by taking into account the more complex relations (16)–(18). Therefore, although we have set the value of 0.15 for quantity (8) in order to receive a 2 s audio fade-in that acts similar to the fade-in effect of exponential curve shape (see Fig. 6), a function is incorporated just to compute the parameters of (1) for any value of ratio (8) that is located within the open intervals of representations (16)–(18). One perceives that as a result of linking the “play” media event to the audio element, the fade-in effect is applied, from the beginning of the audio sample, whenever the audio has been started. As aforementioned, the discrete-time processing is accomplished here by means of the “setInterval()” method of the “window” object. More precisely, as the code plainly indicates, once every 50 ms, the current position within the audio content (sample) i.e. the audio object “currentTime” property is passed to a specialized function that returns the output of rational function (1), which, in turn, is used to update the audio object “volume” property. The code of the described application is put forward in what follows right away.

Listing 1. The code of the described application

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

var ae;                 // the audio element (object)
var k, alpha, beta;     // parameters of the rational function, shaping
                        // the fade-in
var timerId;            // id value returned by setInterval() method

function compPrm( tF, vF, rhoH ) {
/* computes the parameters of the algebraic fraction (1);
function arguments: fade length, final volume i.e. audio volume to be
reached, ratio (8) between the audio volume at fade-in midpoint and the
final volume */

   var rhoH2 = rhoH + rhoH;
   var rhoH4, rhoH8, auxVar;
           // auxiliary variables
   if ( 0.5 < rhoH && rhoH < 1.0 ) {
      k = 1;
      auxVar = rhoH2 - 1.0;
      alpha = rhoH * vF / auxVar;
            // based on representation (17), for 1/2 < rhoH < 1
      beta = ( 1.0 - rhoH ) * tF / auxVar;
            // based on representation (18), for 1/2 < rhoH < 1
   }
   else if ( 0.25 < rhoH && rhoH < 0.5 ) {
      k = 2;
      rhoH4 = rhoH2 + rhoH2;
      auxVar = rhoH4 - 1.0;
      alpha = rhoH2 * vF / auxVar / tF;
            // based on representation (17), for 1/4 < rhoH < 1/2
      beta = ( 1.0 - rhoH2 ) * tF / auxVar;
            // based on representation (18), for 1/4 < rhoH < 1/2
   }
   else {      // 0.125 < rhoH && rhoH < 0.25 (1/8 < rhoH < 1/4)
      k = 3;
      rhoH4 = rhoH2 + rhoH2; rhoH8 = rhoH4 + rhoH4;
      auxVar = rhoH8 - 1.0;
      alpha = rhoH4 * vF / auxVar / tF / tF;
            // based on representation (17), for 1/8 < rhoH < 1/4
      beta = ( 1.0 - rhoH4 ) * tF / auxVar;
            // based on representation (18), for 1/8 < rhoH < 1/4
   }
}
```

```
function v( t ) {
/* returns the output of the rational function shaping the fade-in,
for a given instant of time */
   var newVol = alpha * t / ( t + beta );
            // algebraic fraction (25), corresponding to 1/2 < rhoH < 1
   if ( k == 2 ) { newVol = newVol * t; }
            // algebraic fraction (23), corresponding to 1/4 < rhoH < 1/2
   else if ( k == 3 ) { newVol = newVol * t * t; }
            // algebraic fraction (21), corresponding to 1/8 < rhoH < 1/4
   return newVol;
}

function setVol( vF ) {
/* sets the audio volume during fading-in;
argument: final volume i.e. the audio volume to be reached */
   var timeVar = ae.currentTime;
            // the current position within the audio content, in second
   var currentVol = v( timeVar );
            // calls v( t ), with parameter t receiving the current
            // playback time
   if ( currentVol < vF ) { ae.volume = currentVol; }
   else {
      ae.volume = vF;                     // volume supressing
      window.clearInterval( timerId );    // clears the timer
   }
}

function fadeIn() {
/* performs the audio fading-in by means of
setInterval() method of the window object */
   ae.currentTime = 0.0; ae.volume = 0.0;
      // audio object properties initialization;
      // the fade-in effect is initiated just at the beginning of audio
      // content
   timerId = window.setInterval( setVol, 50, 1.0 );
      // the setInterval() method calls setVol() function once every 50 ms;
      // the setInterval() method passes the value of 1 to parameter vF
}

compPrm( 2.0, 1.0, 0.15 );
            // arguments: fade length of 2 s, final volume of 1, ratio (8)
            // of 0.15
ae = document.createElement( “AUDIO” );
            // creates the audio element (object)
ae.controls = true;        // displays audio controls
ae.src = “sample.mp3”;     // indicates an audio/mpeg file; provided by user
ae.addEventListener( “play”, fadeIn );
            // associates the play event swith the audio element
document.body.appendChild( ae );
            // appends the created audio element to the document

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

The implementation given next has been reduced to a straightforward and short structure in order to highlight a significant benefit that comes from associating the “play” media event with the audio element (Lupsa-Tataru, 2017).

Listing 2. Reduced structure of developed code

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

var ae;
var alpha, beta;

var refTime;
   // holds the playback position when the audio has started to play

var timerId;

function compPrm( tF, vF, rhoH ) {
   // for the case of 1/8 < rhoH < 1/4 only
   var rhoH4 = rhoH + rhoH; rhoH4 = rhoH4 + rhoH4;
   var rhoH8 = rhoH4 + rhoH4;
   var auxVar = rhoH8 - 1.0;
   alpha = rhoH4 * vF / auxVar / tF / tF;
   beta = ( 1.0 - rhoH4 ) * tF / auxVar;
}
function v( t ) {
   var newVol = alpha * t * t * t / ( t + beta );
   return newVol;
}
function setVol( vF ) {
   var tau = ae.currentTime - refTime;
   var currentVol = v( tau );
   if ( currentVol < vF ) { ae.volume = currentVol; }
   else {
      ae.volume = vF;
      window.clearInterval( timerId );
   }
}
function fadeIn() {
   refTime = ae.currentTime; ae.volume = 0.0;
   timerId = window.setInterval( setVol, 50, 1.0 );
}

compPrm( 2.0, 1.0, 0.15 );
ae = document.createElement( “AUDIO” );
ae.controls = true;
ae.src = “sample.mp3”;
ae.addEventListener( “play”, fadeIn );
document.body.appendChild( ae );

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

One observes that, although this second implementation has been designed to work merely for the case of $1/8 < \rho_h < 1/4$, it comes with the plus of allowing the initiation of the fade-in effect not only at the beginning of the audio content but also whenever the audio is no more paused. This obviously will smooth the listening experience for long audio contents on audio/video sharing platforms, when the user alternatively accesses the “play” and “pause” standard media controls. To gain benefit, in addition to the first application, we have introduced the “refTime” variable of global scope in order to hold the playback position within the audio content whenever the user accesses the “play” audio control. Hence, the fade-in audio effect is applied here with respect to the current value of “refTime” global variable i.e. according to (21), wherein the following replacement is to be performed

$$t \leftarrow \tau;$$
$$\tau = t - t_{ref}$$

with t_{ref} standing for the value of “refTime” variable of the implementation, which comes now to be the instant of fade-in initiation.

5. CONCLUSIONS

Audio fade-ins are widely used to smooth the transition from silence of audio signals (Case, 2007; Jackson, 2015; Langford, 2014; Reiss & McPherson, 2015; Schroder, 2011). They are usually applied within audio editors, i.e. in off-line mode, by employing different transcendental functions of time variable to impose the evolution of audio level. However, the design of various interactive products, like simulation software and entertainment applications, frequently calls for fast audio processing techniques. By adopting a rational function to enforce the time-related evolution of the audio volume, the present investigation advances an efficient method of constructing the audio fade-in profile, suitable for real-time computing. It is shown that the resulted fade-ins, straightforwardly obtained by valuating the output of the selected rational function for numerous values of the encompassed parameters, can act either as the audio fade-in of exponential type or as the audio fade-in of logarithmic shape.

With the audio content/sample as the preference of the reader, the optimized implementations in pure JavaScript, put forward in the present paper, plainly emphasize the effectiveness of the proposed solution to constructing the audio fade-in profile.

Further developments could be geared towards real-time audio cross-fading techniques, by employing the method of audio fading-in suggested here along with a technique of audio fading-out suitable for fast computing (Lupsa-Tataru, 2018).

REFERENCES

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

Corey, J. (2017). *Audio Production and Critical Listening: Technical Ear Training.* New York, NY, USA: Routledge.

Devlin, I. (2012). *HTML5 Multimedia: Develop and Design.* Berkeley, CA, USA: Peachpit Press.

Jackson, W. (2015). *Digital Audio Editing Fundamentals: Get started with digital audio development and distribution*. Berkeley, CA, USA: Apress Media. doi:10.1007/978-1-4842-1648-4

Jacobs, I., Jaffe, J., & Le Hegaret, P. (2012). How the open web platform is transforming industry. *IEEE Internet Computing, 16*(6), 82-86. doi:10.1109/MIC.2012.134

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

Lupsa-Tataru, L. (2017). Shaping the fade-in audio effect with a view to JavaScript implementation. *Journal of Computations & Modelling, 7*(4), 111–126.

Lupsa-Tataru, L. (2018). Novel technique of customizing the audio fade-out shape. *Applied Computer Science, 14*(3), 5–14. doi:10.23743/acs-2018-17

Panagakis, Y., Kotropoulos, C. L., & Arce, G. R. (2014). Music genre classification via joint sparse low-rank representation of audio features. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 22*(12), 1905–1917. doi:10.1109/TASLP.2014.2355774

Potter, K. (2002). *Four Musical Minimalists: La Monte Young, Terry Riley, Steve Reich, Philip Glass (Series: Music in the Twentieth Century).* Cambridge, UK: Cambridge University Press.

Powers, S. (2011). *HTML5 Media.* Sebastopol, CA, USA: O'Reilly Media.

Reiss, J. D., & McPherson, A. (2015). *Audio Effects: Theory, Implementation and Application.* Boca Raton, FL, USA: CRC Press.

Schroder, C. (2011). *The Book of Audacity: Record, Edit, Mix, and Master with the Free Audio Editor.* San Francisco, CA, USA: No Starch Press.

WebPlat WG (Web Platform Working Group). (2017). *HTML 5.2. W3C Recommendation*. W3C technical reports index: https://www.w3.org/TR/2017/REC-html52-20171214