Technium Vol. 4, No. 9 pp.12-22 (2022)
ISSN: 2668-778X
www.techniumscience.com

Piecewise-defined function for effectively implementing the audio volume automation

Lucian Lupşa-Tătaru

Faculty of Electrical Engineering and Computer Science,
"Transilvania" University of Braşov, Bd. Eroilor No. 29, Braşov, Romania,
lupsa@programmer.net, lucian.lupsa@unitbv.ro

Abstract. Typically performed with the aid of media development software like digital audio workstations, audio volume automation requires the implementation of piecewise functions to customize the amplitude envelope of audio contents. The user sets various control points by imposing values for the audio volume at different playback positions and, in return, the application shapes the fades to be applied between each two neighboring control points. The adjustable fades, which act over the subintervals defined by each two back-to-back control points, are traditionally implemented by means of labored transcendental functions. In order to effectively construct the envelope of audio contents as well as to heighten the audio experience with a view to real-time computing, we advance a piecewise mapping with the sub-functions, depicting the adjustable fades, represented by rational functions only. A plain implementation in JavaScript is presented in the paper in order to highlight the real-time audio capabilities of the suggested procedure of performing audio volume automation.

Keywords. Audio volume automation, audio fade, piecewise function, real-time computing, programming techniques, HTML, JavaScript.

1. Introduction

Audio volume automation is mostly carried out in application software such as digital audio workstations (DAWs) by allowing the user to place various control points with the purpose of generating an amplitude envelope [1]-[7]. Since a control point simply indicates the expected audio volume at a certain playback position, adjustable fades (fade-up and/or fade-down audio effects) have to be implemented to automate the volume between each two control points that are placed successively [2]-[8]. In view of this, it has to be pointed out that the audio fades applied between two back-to-back control points are routinely implemented by adopting transcendental functions (especially, the exponential function) to depict the time-related evolution of the audio volume. In this circumstance, volume automation comes to be the result of employing piecewise-defined functions with the constituent sub-functions as elaborate transcendental functions. Nevertheless, with respect to the current practice of incorporating interactive audio not only within applications designed for entertainment purposes but also within simulation and educational software [9]-[15], the employment of transcendental functions merely to build up the amplitude envelope of audio contents seems to be less effective.

In the present investigation, to implement the process of audio volume automation and to enhance the audio experience in the direction of real-time computing, we advance a piecewise-defined function by plainly considering that each individual sub-function is a rational function that encompasses three

coefficients. More precisely, we consider that the adjustable fade to be applied between two successive control points is described by a very specific rational function, previously introduced to persuasively customize the fade-out audio effect [16].

Hence, the starting point in deducing here the piecewise-defined function for real-time volume automation is represented by expressing the coefficients of the adopted rational function, which eventually yields the piecewise mapping sub-functions, so as to receive adjustable fades that can resemble, as the case may be, the exponential fade or the logarithmic fade shape. In this regard, one has to take into consideration that the rate of change of the audio volume in case of a fade-up of exponential shape is smaller at the beginning than at the end of the effect whilst for a fade-up of logarithmic shape, the volume rate of change is smaller at the fade-up ending than at the position of effect initiation. One perceives that the same applies to the case of fade-down effect, during which the audio volume strictly decreases and, thus, one has to deal with a negative rate of change of audio volume.

2. The adjustable fade

An adjustable fade is applied in order to strictly increase or decrease the volume of a certain audio content [2]-[8]. The adjustable fades (fade-up, fade-down) are usually implemented by considering that the audio volume is the output of different transcendental functions of playback position. More specifically, each fade shape is associated with a particular transcendental function and, thus, changing the fade shape requires switching from a predefined transcendental function to another. Such kind of approach to audio fade customization is more suitable for off-line processing within audio editors than for reactive computing, when time constraints are perpetually imposed.

To customize the fading process with a view to real-time computing, we adopt here the following rational function that depicts the time-related evolution of the audio volume [16]:

$$\begin{cases} v(\tau) = \dfrac{\tau - \alpha}{\beta\tau - \gamma}, \\ \tau \in [0, \tau_f] \end{cases} \tag{1}$$

where τ_f designates the fade length, τ is the playback position, counted from the initiation of the effect, while output v is precisely the audio volume.

With v_0 as the initial audio volume, the boundary condition

$$\begin{cases} v(0) = v_0, \\ v_0 \geq 0 \end{cases} \tag{2}$$

leads to relationship

$$\alpha = \gamma v_0 . \tag{3}$$

Taking into account (1) and (3), the instantaneous volume becomes:

$$v(\tau) = \frac{\tau - \gamma v_0}{\beta\tau - \gamma} . \tag{4}$$

Imposing now v_f as the final volume, we have:

$$\begin{cases} v(\tau_f) = v_f, \\ v_f \geq 0, v_f \neq v_0 . \end{cases} \tag{5}$$

Alongside relation (4), the restrictive condition (5) yields

$$\tau_f - \gamma v_0 = (\beta\tau_f - \gamma)v_f \,. \tag{6}$$

Out of (6) one receives the coefficient γ in terms of coefficient β, fade length τ_f, and the audio volume extrema, i.e.

$$\begin{cases} \gamma = \tau_f \dfrac{\beta v_f - 1}{v_f - v_0} \,, \\ v_f \neq v_0 \,. \end{cases} \tag{7}$$

Having in view relationships (4) and (7), the audio volume can be provided by means of an expression encompassing the coefficient β only:

$$v(\tau) = \frac{\tau - \tau_f \frac{\beta v_f - 1}{v_f - v_0} v_0}{\beta\tau - \tau_f \frac{\beta v_f - 1}{v_f - v_0}} = \frac{(v_f - v_0)\tau - \tau_f v_0(\beta v_f - 1)}{\beta(v_f - v_0)\tau - \tau_f(\beta v_f - 1)} \,. \tag{8}$$

It has to be pointed out that relation (1) has originally been employed to customize the fade-out effect in the regular manner that is by imposing the audio volume at the fade halfway point [16]. For the more general case of constructing adjustable fades of any type (fade-up, fade-down), it appears more convenient for now to impose the playback time, within the fading process, at which the audio volume gets the mean value μ_v. Thus, one may set:

$$\begin{cases} v(\tau_\mu) = \mu_v \equiv (v_0 + v_f)/2 \,, \\ \tau_\mu \in (0, \tau_f) \,. \end{cases} \tag{9}$$

With the audio volume as the output of mapping (8), condition (9) of reaching the volume mean value μ_v at the playback position τ_μ becomes equivalent to

$$\frac{\tau_\mu(v_f - v_0) - \tau_f v_0(\beta v_f - 1)}{\beta\tau_\mu(v_f - v_0) - \tau_f(\beta v_f - 1)} = \frac{v_0 + v_f}{2} \,. \tag{10}$$

One observes that relationship (10) enables the expressing of coefficient β in terms of initial and final audio volume, respectively, fade length and the playback time of achieving the audio volume mean value. More precisely, from (10) one receives:

$$\begin{aligned} &\beta = \frac{2\tau_\mu - \tau_f}{\tau_\mu(v_0 + v_f) - \tau_f v_f} \,, \\ &\tau_\mu \neq \tau_f v_f / (v_0 + v_f) \,. \end{aligned} \tag{11}$$

Furthermore, taking into account relations (11), (7) and (3), one successively identifies

$$\gamma = \tau_f \frac{\frac{2\tau_\mu - \tau_f}{\tau_\mu(v_0 + v_f) - \tau_f v_f} v_f - 1}{v_f - v_0} \,, \tag{12}$$

$$\gamma = \frac{\tau_\mu \tau_f}{\tau_\mu (v_0 + v_f) - \tau_f v_f}\,, \quad \tau_\mu \neq \tau_f v_f / (v_0 + v_f)\,, \tag{13}$$

$$\alpha = \frac{\tau_\mu \tau_f v_0}{\tau_\mu (v_0 + v_f) - \tau_f v_f}\,, \quad \tau_\mu \neq \tau_f v_f / (v_0 + v_f)\,. \tag{14}$$

To make the approach more generic, instead of shaping the fade by specifying the playback position τ_μ at which the audio volume reaches its mean value, we employ the following parameter:

$$\varepsilon = \tau_\mu / \tau_f\,, \quad \varepsilon \in (0,1);\ \varepsilon \neq v_f / (v_0 + v_f)\,. \tag{15}$$

Coupling (15) with expressions (11), (13) and (14) of the three coefficients interfering in (1), one obtains:

$$\alpha = \frac{\tau_f v_0}{v_0 + v_f - \varepsilon^{-1} v_f}\,, \tag{16}$$

$$\beta = \frac{2 - \varepsilon^{-1}}{v_0 + v_f - \varepsilon^{-1} v_f}\,, \tag{17}$$

$$\gamma = \frac{\tau_f}{v_0 + v_f - \varepsilon^{-1} v_f}\,. \tag{18}$$

Expressions (16)-(18), now provided in terms of parameter (15), have to ensure that mapping (1) is strictly monotone. Considering formula (1), one straightforwardly receives the instantaneous rate of change of the audio volume during the fading process [16]:

$$\frac{dv}{d\tau} = \frac{\alpha\beta - \gamma}{(\beta\tau - \gamma)^2} \tag{19}$$

wherein, by means of the newly introduced expressions (16)-(18), one may emphasize

$$\alpha\beta - \gamma = \gamma(\beta v_0 - 1) = \frac{(\varepsilon^{-1} - 1)\tau_f}{\left(v_0 + v_f - \varepsilon^{-1} v_f\right)^2} (v_f - v_0)\,. \tag{20}$$

Relations (19) and (20) plainly lead to

$$\frac{dv}{d\tau} = \frac{(\varepsilon^{-1} - 1)\tau_f}{\left[(v_0 + v_f - \varepsilon^{-1} v_f)(\beta\tau - \gamma)\right]^2} (v_f - v_0)\,. \tag{21}$$

Taking into account (21) and (15), one perceives that when the difference of the final volume and the initial volume is (strictly) positive then the audio volume has a positive rate of change whilst if the difference of the final volume and the initial volume is (strictly) negative then the audio volume has a negative rate of change.

3. Volume automation

By performing the substitutions

$$\begin{aligned} &\tau \leftarrow t - t_i \; ; \; t \geq t_i \, , \\ &\tau_f \leftarrow t_{i+1} - t_i \; ; \; t_{i+1} > t_i \end{aligned} \tag{22}$$

within relation (1), and expressions (16)-(18), respectively, one eventually figures out the procedure needed for fade shaping over a certain interval $[t_i, t_{i+1}]$, $i \geq 0$. More specifically, one derives:

$$\begin{cases} V(t) = \dfrac{t - t_i - \alpha_i}{\beta_i t - \beta_i t_i - \gamma_i} \, , \\ t \in [t_i, t_{i+1}] \, , \end{cases} \tag{23}$$

$$\alpha_i = \frac{(t_{i+1} - t_i)V_i}{V_i + V_{i+1} - \varepsilon_i^{-1} V_{i+1}} , \tag{24}$$

$$\beta_i = \frac{2 - \varepsilon_i^{-1}}{V_i + V_{i+1} - \varepsilon_i^{-1} V_{i+1}} , \tag{25}$$

$$\gamma_i = \frac{t_{i+1} - t_i}{V_i + V_{i+1} - \varepsilon_i^{-1} V_{i+1}} , \tag{26}$$

wherein (t_i, V_i) and (t_{i+1}, V_{i+1}) represent here back-to-back control points.

For more than two control points, one has to deal with volume automation over an interval $[t_0, t_n]$, $n \geq 2$. In this case, constructing the amplitude envelope implies the adoption of mapping (23) as the sub-function of a piecewise-defined function, which has to describe the time-related evolution of audio volume over the considered interval $[t_0, t_n]$. Hence, real-time volume automation with control points (t_0, V_0), (t_1, V_1), …(t_n, V_n) is to be accomplished by means of the following piecewise function:

$$V(t) = \begin{cases} \dfrac{t - t_0 - \alpha_0}{\beta_0 t - \beta_0 t_0 - \gamma_0} \; ; \; t \in [t_0, t_1) \, , \\ \quad \vdots \\ \dfrac{t - t_{n-1} - \alpha_{n-1}}{\beta_{n-1} t - \beta_{n-1} t_{n-1} - \gamma_{n-1}} \; ; \; t \in [t_{n-1}, t_n] \, , \end{cases} \tag{27}$$

wherein the encompassed coefficients get the expressions

$$\alpha_0 = \frac{(t_1 - t_0)V_0}{V_0 + V_1 - \varepsilon_0^{-1} V_1} , \quad \ldots \quad \alpha_{n-1} = \frac{(t_n - t_{n-1})V_{n-1}}{V_{n-1} + V_n - \varepsilon_{n-1}^{-1} V_n} , \tag{28}$$

$$\beta_0 = \frac{2 - \varepsilon_0^{-1}}{V_0 + V_1 - \varepsilon_0^{-1} V_1} , \quad \ldots \quad \beta_{n-1} = \frac{2 - \varepsilon_{n-1}^{-1}}{V_{n-1} + V_n - \varepsilon_{n-1}^{-1} V_n} , \tag{29}$$

$$\gamma_0 = \frac{t_1 - t_0}{V_0 + V_1 - \varepsilon_0^{-1} V_1} , \quad \ldots \quad \gamma_{n-1} = \frac{t_n - t_{n-1}}{V_{n-1} + V_n - \varepsilon_{n-1}^{-1} V_n} . \tag{30}$$

Over each subinterval of the body of piecewise mapping (27), the shape of the appropriate fade is customized by imposing the value of the associate parameter ε_i, $i \in [0, n-1]$.

Technium Vol. 4, No. 9 pp.12-22 (2022)
ISSN: 2668-778X
www.techniumscience.com

Considering, for instance, $n = 3$ in (27), volume automation over the interval $[t_0, t_3]$ results in customizing three audio fades, depicted by the three sub-functions that serve to make up the following piecewise representation:

$$V(t) = \begin{cases} \dfrac{t - t_0 - \alpha_0}{\beta_0 t - \beta_0 t_0 - \gamma_0} \; ; \; t \in [t_0, t_1) \, , \\ \dfrac{t - t_1 - \alpha_1}{\beta_1 t - \beta_1 t_1 - \gamma_1} \; ; \; t \in [t_1, t_2) \, , \\ \dfrac{t - t_2 - \alpha_2}{\beta_2 t - \beta_2 t_2 - \gamma_2} \; ; \; t \in [t_2, t_3] \, , \end{cases} \tag{31}$$

wherein, according to (24)-(26), and for $i \in \{0, 1, 2\}$,

$$\alpha_0 = \frac{(t_1 - t_0)V_0}{V_0 + V_1 - \varepsilon_0^{-1} V_1} \, , \quad \alpha_1 = \frac{(t_2 - t_1)V_1}{V_1 + V_2 - \varepsilon_1^{-1} V_2} \, , \quad \alpha_2 = \frac{(t_3 - t_2)V_2}{V_2 + V_3 - \varepsilon_2^{-1} V_3} \, , \tag{32}$$

$$\beta_0 = \frac{2 - \varepsilon_0^{-1}}{V_0 + V_1 - \varepsilon_0^{-1} V_1} \, , \quad \beta_1 = \frac{2 - \varepsilon_1^{-1}}{V_1 + V_2 - \varepsilon_1^{-1} V_2} \, , \quad \beta_2 = \frac{2 - \varepsilon_2^{-1}}{V_2 + V_3 - \varepsilon_2^{-1} V_3} \, , \tag{33}$$

$$\gamma_0 = \frac{t_1 - t_0}{V_0 + V_1 - \varepsilon_0^{-1} V_1} \, , \quad \gamma_1 = \frac{t_2 - t_1}{V_1 + V_2 - \varepsilon_1^{-1} V_2} \, , \quad \gamma_2 = \frac{t_3 - t_2}{V_2 + V_3 - \varepsilon_2^{-1} V_3} \, . \tag{34}$$

Examining relations (31)-(34), one identifies the four control points (t_0, V_0), (t_1, V_1), (t_2, V_2), (t_3, V_3) and the three parameters ε_0, ε_1, ε_2, required for fade shaping over the subintervals $[t_0, t_1)$, $[t_1, t_2)$ and $[t_2, t_3]$, respectively.

The envelopes depicted in Fig. 1 till Fig. 4 have been obtained by adopting the piecewise mapping (31) and correspond to the following four control points:

$$(t_0 \equiv 0 \text{ s}, \, V_0 \equiv 0), \qquad (t_1 \equiv 5 \text{ s}, \, V_1 \equiv 1),$$
$$(t_2 \equiv 25 \text{ s}, \, V_2 \equiv 0.6), \qquad (t_3 \equiv 30 \text{ s}, \, V_3 \equiv 0).$$

One observes that achieving each of the envelopes illustrated in Fig. 1 till Fig. 4 asks for successively performing a 5 s fade-in, a 20 s fade-down and a 5 s fade-out audio effect. On the other hand, as they correspond to distinct values of the parameters ε_0, ε_1 and ε_2, which interfere in expressions (32)-(34), Fig. 1 up to Fig. 4 clearly reveal the high grade of versatility in performing volume automation by employing mapping (31).

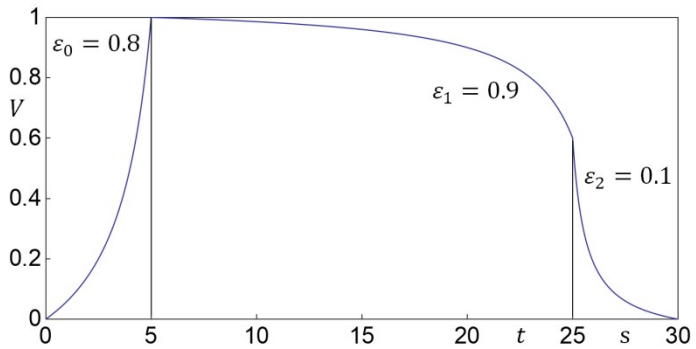

Fig. 1. Volume automation by adopting the piecewise function (31), the control points (0, 0), (5, 1), (25, 0.6), (30, 0) and the parameters ε_0, ε_1, ε_2 in (32)-(34) of value 0.8, 0.9 and 0.1, respectively.

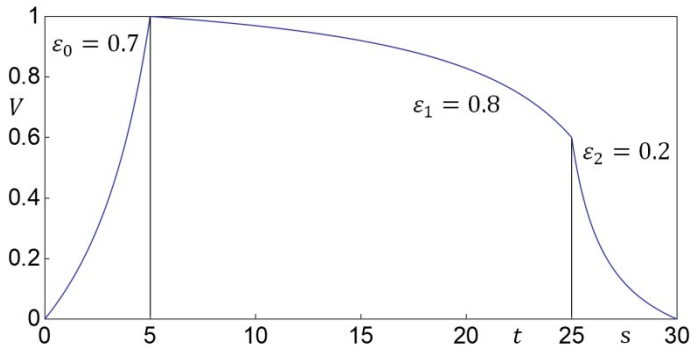

Fig. 2. Volume automation by adopting the piecewise function (31), the control points (0, 0), (5, 1), (25, 0.6), (30, 0) and the parameters ε_0, ε_1, ε_2 in (32)-(34) of value 0.7, 0.8 and 0.2, respectively.

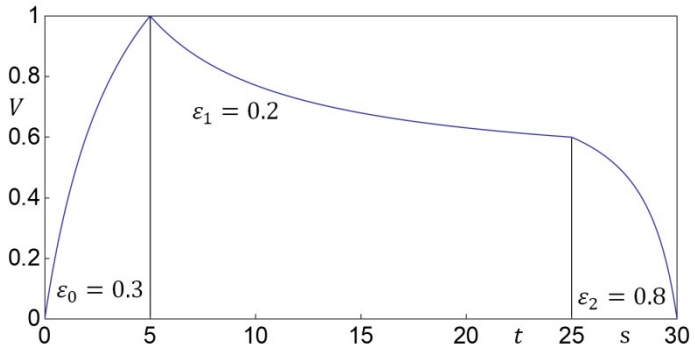

Fig. 3. Volume automation by adopting the piecewise function (31), the control points (0, 0), (5, 1), (25, 0.6), (30, 0) and the parameters ε_0, ε_1, ε_2 in (32)-(34) of value 0.3, 0.2 and 0.8, respectively.

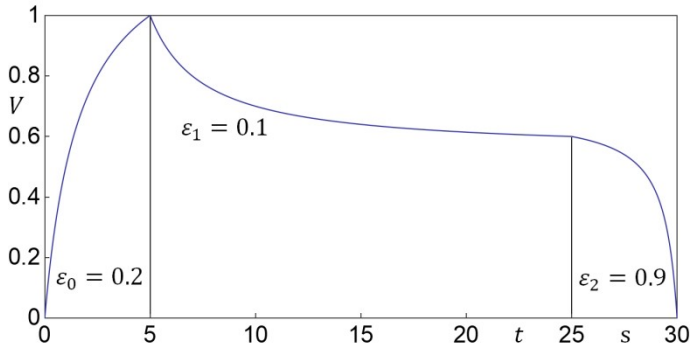

Fig. 4. Volume automation by adopting the piecewise function (31), the control points (0, 0), (5, 1), (25, 0.6), (30, 0) and the parameters ε_0, ε_1, ε_2 in (32)-(34) of value 0.2, 0.1 and 0.9, respectively.

Having in view (15), parameter ε_0 is here directly proportional to the playback time at which the audio volume attains the value of 0.5, parameter ε_1 is directly proportional to the playback position, relative to 5 s, i.e. the instant of fade-in completion, at which the volume during the fade-down process reaches its mean value of 0.8 whilst parameter ε_2 is proportional to the playback position, with respect to 25 s that is the instant of fade-out initiation, at which the volume has the value of 0.3.

In this context, Fig. 1 up to Fig. 4 highlight that the fade-ins received for $\varepsilon_0 \in \{0.7, 0.8\}$ and the fade-outs corresponding to $\varepsilon_2 \in \{0.1, 0.2\}$ act similar to the exponential shape fades just as the fade-ins obtained for $\varepsilon_0 \in \{0.2, 0.3\}$ and the fade-outs corresponding to $\varepsilon_2 \in \{0.8, 0.9\}$ act much the same as the logarithmic fades.

Technium Vol. 4, No. 9 pp.12-22 (2022)
ISSN: 2668-778X
www.techniumscience.com

Furthermore, for parameter $\varepsilon_1 \in \{0.8, 0.9\}$, the resulted fade-down effects follow the pattern of the logarithmic fades, with an absolute value (modulus) of the audio volume rate of change that is smaller at the beginning than at the end of the effect. At the same time, the fade-down effects received for $\varepsilon_1 \in \{0.1, 0.2\}$ resemble the exponential fades, having an absolute value (modulus) of the audio volume rate of change that is smaller at the fade-down ending than at the fade-down initiation.

4. Implementation for real-time processing

In order to validate the real-time audio capabilities of the suggested technique of carrying out the audio volume automation, i.e. by employing rational functions to customize the fades between each two neighboring control points, we have prepared an HTML/JavaScript application. To accomplish the process of discretization, we have proceeded to invoke the "setInterval()" method of the "window" object. On condition that the user provides the needed audio sample, the application carries out the process of volume automation according to the piecewise representation illustrated in Fig. 1, obtained by adopting mapping (31). One has to take into consideration that, in compliance with the current specification of HTML, the "volume" property of an audio object is restricted to the interval [0, 1], where the value of 0 signifies silence [15]. Hence, the audio volume automation is initiated just at the beginning of the audio sample and consists in gradually applying a 5 s fade-in, a 20 s fade-down and a 5 s fade-out effect upon the sample playing. The code of the developed application is presented next.

```
<!DOCTYPE html>
<html>
<head>
   <title>Volume automation</title>
</head>
<body>
<script>

var alpha0, beta0, gamma0;        /* the coefficients of the 1st sub-function of (31) */
var alpha1, beta1, gamma1;        /* the coefficients of the 2nd sub-function of (31) */
var alpha2, beta2, gamma2;        /* the coefficients of the 3rd sub-function of (31) */
var audElm;                       /* the audio element (object) */
var tmrId;                        /* timer id returned by the setInterval() method
                                     and used to call the clearInterval() method */
function alpha012( tI, vI, tI1, vI1, epsI ) {      /* implementation of relation (24) */
   var tauF = tI1 - tI; var denomi = vI + vI1 - vI1 / epsI;
   var alphaI = tauF * vI / denomi;
   return alphaI;
}
function beta012( vI, vI1, epsI ) {                /* implementation of relation (25) */
   var denomi = vI + vI1 - vI1 / epsI;
   var betaI = ( 2.0 - 1.0 / epsI ) / denomi;
   return betaI;
}
function gamma012( tI, vI, tI1, vI1, epsI ) {      /* implementation of relation (26) */
   var tauF = tI1 - tI; var denomi = vI + vI1 - vI1 / epsI;
   var gammaI = tauF / denomi;
   return gammaI;
}
function set0( t0, v0, t1, v1, eps0 ) {
   /* yields the coefficients of the 1st sub-function of mapping (31) */
   alpha0 = alpha012( t0, v0, t1, v1, eps0 ); beta0 = beta012( v0, v1, eps0 );
   gamma0 = gamma012( t0, v0, t1, v1, eps0 );
}
function set1( t1, v1, t2, v2, eps1 ) {
   /* yields the coefficients of the 2nd sub-function of mapping (31) */
   alpha1 = alpha012( t1, v1, t2, v2, eps1 ); beta1 = beta012( v1, v2, eps1 );
   gamma1 = gamma012( t1, v1, t2, v2, eps1 );
}
```

Technium Vol. 4, No. 9 pp.12-22 (2022)
ISSN: 2668-778X
www.techniumscience.com

```
function set2( t2, v2, t3, v3, eps2 ) {
   /* yields the coefficients of the 3rd sub-function of mapping (31) */
   alpha2 = alpha012( t2, v2, t3, v3, eps2 ); beta2 = beta012( v2, v3, eps2 );
   gamma2 = gamma012( t2, v2, t3, v3, eps2 );
}
function v012( t, tI, alphaI, betaI, gammaI ) {     /* returns the output of (23) */
   var tau = t - tI;
   var vI_I1 = ( tau - alphaI ) / ( betaI * tau - gammaI );
   if ( vI_I1 < 0.0 ) { vI_I1 = 0.0; }              /* audio volume in [0, 1] */
   else if ( vI_I1 > 1.0 ) { vI_I1 = 1.0; }
   return vI_I1;
}
function currentV( t0, t1, t2, t3 ) {
   /* implementation of piecewise-defined mapping (31) */
   var currentT = audElm.currentTime;
   if ( currentT < t1 ) {
      audElm.volume = v012( currentT, t0, alpha0, beta0, gamma0 );
   }
   else if ( currentT < t2 ) {
      audElm.volume = v012( currentT, t1, alpha1, beta1, gamma1 );
   }
   else if ( currentT <= t3 ) {
      audElm.volume = v012( currentT, t2, alpha2, beta2, gamma2 );
   }
   else {
      /* stops the volume automation */
      audElm.volume = 0.0;
      audElm.pause();
      window.clearInterval( tmrId );
   }
}
function set012( eps0, eps1, eps2 ) {
   /* computes the coefficients of mapping (31) for the control points
      (t0 = 0 s, v0 = 0), (t1 = 5 s, v1 = 1),
      (t2 = 25 s, v2 = 0.6), (t3 = 30 s, v3 = 0) */
   set0( 0.0, 0.0, 5.0, 1.0, eps0 );
   set1( 5.0, 1.0, 25.0, 0.6, eps1 );
   set2( 25.0, 0.6, 30.0, 0.0, eps2 );
}
function automateV() {
   if ( audElm.duration > 30.0 ) {          /* t3 = 30 s */
      audElm.currentTime = 0.0;             /* t0 = 0 s */
      audElm.volume = 0.0;                  /* v0 = 0 */
      tmrId = window.setInterval( currentV, 60, 0.0, 5.0, 25.0, 30.0 );
         /* invokes function currentV() once every 60 ms and passes
            the arguments t0 = 0 s, t1 = 5 s, t2 = 25 s, t3 = 30 s */
   }
   else { alert( “Not enough room to complete volume automation” ); }
}
set012( 0.8, 0.9, 0.1 );           /* eps0 = 0.8, eps1 = 0.9, eps2 = 0.1 */

audElm = document.createElement( “AUDIO” );
audElm.src = “sample.mp3”;         /* audio content is provided by the user */
audElm.controls = true;
audElm.addEventListener( “play”, automateV );
   /* initiates volume automation just when the playback has started */
document.body.appendChild( audElm );

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

The code clearly highlights that the variables of global scope designate the coefficients of the piecewise-defined mapping (31), the audio object and the timer (interval) identifier returned by the "setInterval()" method, used here for discretization. Moreover, one easily observes that functions "alpha012()", "beta012()" and "gamma012()" are just the implementations of relationships (24)-(26) whilst function "v012()" returns the output of mapping (23). When invoked, function "set0()" computes the coefficients of the first sub-function of piecewise mapping (31), which are denoted by variables "alpha0", "beta0" and "gamma0", function "set1()" computes the coefficients of the second sub-function of (31), denoted by variables "alpha1", "beta1" and "gamma1", while function "set2()" computes the values of variables "alpha2", "beta2" and "gamma2" i.e. the coefficients encompassed by the third sub-function of (31).

The calculation of the coefficients interfering in piecewise mapping (31) is carried out by simply calling the function "set012()" and passing the appropriate arguments (the parameters values) required for fade shaping over the three subintervals of mapping (31). As previously mentioned, the application put forward here performs the real-time volume controlling in accordance with the time-related evolution depicted in Fig. 1 and, hence, the parameters of function "set012()" are of value 0.8, 0.9 and 0.1, respectively. One perceives that, in order to follow any of the envelopes depicted in Fig. 2, Fig. 3 and Fig.4, the arguments passed to function "set012()" need to be changed to the appropriate values, clearly indicated in Fig. 2 till Fig.4.

For more versatility, to implement the piecewise-defined mapping (31), function "currentV()" has been defined by introducing the subintervals endpoints as parameters. As the block of function "automateV()" indicates, the "setInterval()" method is employed here to call function "currentV()" once every 60 ms and to pass the corresponding subintervals limits. Thus, parameters "t0", "t1", "t2" and "t3", designating the endpoints of the subintervals of the body of mapping (31), remain local to function "currentV()".

5. Conclusion

Carrying out the process of audio volume automation requires the implementation of piecewise-defined functions with the constitutive sub-functions defined so as to impose certain pre-established shapes for the fades occurring between each two neighboring control points. Traditionally, audio volume automation is accomplished in application software like digital audio workstations (DAWs) with the purpose of audio production, i.e. exporting audio files. On the other hand, one observes the increasing demand for integrating interactive audio within entertainment, educational and simulation software, respectively [9]-[15]. Thus, the implementation of audio volume automation is no longer restricted to the range of applications designed for media development only but it has become a major challenge in the process of creating various applications that ask for real-time audio capabilities. From this perspective, having in view that the adjustable fades, i.e. fade-up and fade-down audio effects, are mostly implemented in software DAWs by means of elaborate transcendental functions, to boost the real-time audio experience, we advance a piecewise function designed for audio volume automation, with the constitutive sub-functions, describing just the adjustable fades, being defined by means of rational fractions only. In an effort to emphasize the benefit of adopting the proposed solution to performing volume automation over audio contents being played, a straightforward HTML/JavaScript implementation is provided and discussed in the paper.

References

[1] A. VENKITARAMAN, C. S. SEELAMANTULA: Temporal envelope fit of transient audio signals. *IEEE Signal Processing Letters*, **20** (12), 1191-1194 (2013). doi: 10.1109/LSP.2013.2284971

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

[3] C. SCHRODER: The Book of Audacity. Record, Edit, Mix, and Master with the Free Audio Editor. San Francisco, CA, USA, No Starch Press, 2011.

[4] J. D. REISS, A. McPHERSON: Audio Effects. Theory, Implementation and Application. Boca Raton, FL, USA, CRC Press, 2015.
[5] A. U. CASE: Sound FX. Unlocking the Creative Potential of Recording Studio Effects. Burlington, MA, USA, Focal Press, 2007.
[6] W. JACKSON: Digital Audio Editing Fundamentals. Get Started with Digital Audio Development and Distribution. Berkeley, CA, USA, Apress Media, 2015. doi: 10.1007/978-1-4842-1648-4
[7] The Audacity Team: Audacity(R) Free, Open Source, Cross-platform Audio Software. Audacity Manual, 2021. https://manual.audacityteam.org
Envelope Tool. https://manual.audacityteam.org/man/envelope_tool.html
[8] The Audacity Team: Audacity(R) Free, Open Source, Cross-platform Audio Software. Audacity Manual, 2021. https://manual.audacityteam.org
Adjustable Fade. https://manual.audacityteam.org/man/adjustable_fade.html
[9] M. SWEET: Writing Interactive Music for Video Games. A Composer's Guide. Upper Saddle River, NJ, USA, Addison-Wesley Professional, 2014.
[10] R. L. BLEIDT et al.: Building the world's most complex TV network: a test bed for broadcasting immersive and interactive audio. *SMPTE Motion Imaging Journal*, **126** (5), 26-34 (2017). doi: 10.5594/JMI.2017.2698618
[11] N. HELYER, D. WOO, F. VERONESI: Artful media. The sonic nomadic: exploring mobile surround-sound interactions. *IEEE MultiMedia*, **16** (2), 12-15 (2009). doi: 10.1109/MMUL.2009.38
[12] R. BLEIDT, A. BORSUM, H. FUCHS, S. M. WEISS: Object-based audio: opportunities for improved listening experience and increased listener involvement. *SMPTE Motion Imaging Journal*, **124** (5), 1-13 (2015). doi: 10.5594/j18579
[13] X. GU, M. DICK, Z. KURTISI, U. NOYER, L. WOLF: Network-centric music performance: practice and experiments. *IEEE Communications Magazine*, **43** (6), 86-93 (2005). doi: 10.1109/MCOM.2005.1452835
[14] K. LIANG, B. SEO, A. KRYCZKA, R. ZIMMERMANN: IDM: An indirect dissemination mechanism for spatial voice interaction in networked virtual environments. *IEEE Transactions on Parallel and Distributed Systems*, **24** (2), 356-367 (2013). doi: 10.1109/TPDS.2012.91
[15] I. DEVLIN: HTML5 Multimedia. Develop and Design. Berkeley, CA, USA, Peachpit Press, 2012.
[16] L. LUPSA-TATARU: Novel technique of customizing the audio fade-out shape. *Applied Computer Science*, **14** (3), 5-14 (2018). doi: 10.23743/acs-2018-17