Applied Computer Science, vol. 15, no. 4, pp. 27–36
doi:10.23743/acs-2019-27

Submitted: 2019-09-17
Revised: 2019-11-06
Accepted: 2019-12-06

audio fade, fade-down, fade-up, real-time processing

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

CUSTOMIZING AUDIO FADES WITH A VIEW TO REAL-TIME PROCESSING

Abstract

To a large extent, an audio fade is distinctly acknowledged as a strict increase (fade-up) or decrease (fade-down) of the volume of an audio content. In this broad context, the widely used fade-in and fade-out sound effects, applied to receive smooth transitions from and down to silence, respectively, appear to be restrictive. Taking into account the increasing demand for multimedia techniques adapted for real-time computing, the present investigation advances straightforward procedures intended for customizing the audio fade-up and fade-down profiles, having at hand well-proven techniques of shaping the fade-in and fade-out audio effects, suitable for fast computing.

1. INTRODUCTION

In audio production, both fade-in and fade-out sound effects are extensively employed not only to smooth the beginning and the ending parts of the audio recordings but also to cross-fade various audio sections (Case, 2007; Langford, 2014; Reiss & McPherson, 2015). On the other hand, the applying of a fade-up or a fade-down sound effect simply results in a strict increase or decrease of the audio volume by a specified amount. Furthermore, the successive applications of fade-up and fade-down effects enable one to control the amplitude envelope of an audio content (Jackson, 2015; Langford, 2014; Schroder, 2011). However, similar to the case of customizing the audio fade-in and fade-out shapes, the adjustable audio fades i.e. the fade-up and fade-down sound effects are usually implemented in the off-line mode, by making use of different transcendental functions (exponential, logarithm, sine) to enforce the time-related evolution of the audio volume between imposed amplitude levels.

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

To boost the computational capabilities with a view to real-time processing, which is actually required by numerous interactive products, the present approach to customizing the audio fades puts forward persuasive methods based on efficient techniques of shaping the audio fade-out and fade-in profiles, which have previously been validated for effectiveness by means of plain JavaScript implementations (Lupsa-Tataru, 2018, 2019).

2. THE AUDIO FADE-DOWN CUSTOMIZING

Assuming that the audio level decreases from the initial value $v_{D,0}$, occurring at the fade initiation, down to the final value $v_{D,f}$, showing at the fade ending, we consider that the evolution of the audio volume during fading-down is given by the output of the following function:

$$v_D(\tau_D) = v_{D,f} + \delta_D(\tau_D), \quad \tau_D \in [0, \tau_{D,f}], \tag{1}$$

wherein the variable

$$\tau_D = t_D - t_{D,ref} \tag{2}$$

is yielded by the difference of the current playback time and the chosen instant of fade-down initiation, whilst $\tau_{D,f}$ stands for the fade-down length. Since quantity $v_{D,f}$ in (1), as the final audio volume, is a constant term, it follows that the rate of change of audio level during fading-down is identical to the rate of change of δ_D in (1).

Thus, customizing the profile of the fade-down effect, provided by the output of (1), is equivalent to shaping the output of function $\delta_D(\tau_D)$ that, from the technical point of view, has to portray a fade-out audio effect. With the purpose of real-time implementation, we plainly consider (Lupsa-Tataru, 2018)

$$\delta_D(\tau_D) = \frac{\tau_D - \alpha_D}{\beta_D \tau_D - \gamma_D}, \quad \tau_D \in [0, \tau_{D,f}]. \tag{3}$$

In order for rational function (3) to describe a fade-out audio effect and, implicitly, in order for (1) to depict a fade-down audio effect, the coefficients of algebraic fraction defining (3) receive the appropriate expressions in terms of imposed maximum (initial) and minimum (final) audio volumes, denoted here by $v_{D,0}$ and $v_{D,f}$, respectively. Hence, one gets (Lupsa-Tataru, 2018)

$$\begin{aligned}\alpha_D &= \tau_{D,f}, \\ \beta_D &= \frac{2\rho_D - 1}{\rho_D \delta_{D,0}} = \left(2 - \frac{1}{\rho_D}\right) / \left(v_{D,0} - v_{D,f}\right), \\ \gamma_D &= \tau_{D,f} / \delta_{D,0} = \tau_{D,f} / \left(v_{D,0} - v_{D,f}\right),\end{aligned} \tag{4}$$

wherein

$$\rho_D = \frac{\delta_D(\tau_{D,f}/2)}{\delta_D(0)} = \frac{\delta_{D,h}}{\delta_{D,0}}; \qquad 0 < \rho_D < 1 \tag{5}$$

or, having in view (1),

$$\rho_D = \frac{v_D(\tau_{D,f}/2) - v_{D,f}}{v_D(0) - v_{D,f}} = \frac{v_{D,h} - v_{D,f}}{v_{D,0} - v_{D,f}}. \tag{6}$$

One can easily observe that, within (4)–(6), we have employed the following auxiliary notations:

$$\begin{aligned}\delta_{D,0} &= \delta_D(0), & v_{D,0} &= v_D(0), \\ \delta_{D,h} &= \delta_D(\tau_{D,f}/2), & v_{D,h} &= v_D(\tau_{D,f}/2).\end{aligned}$$

The technique of customizing the audio fade-out profile by means of rational function (3), which serves as groundwork for the suggested method of shaping the fade-down audio effect, has been validated by a previously advanced implementation in plain (“vanilla”) JavaScript (Lupsa-Tataru, 2018). In the present context, it comes to be obvious that implementing the fade-down audio effect by valuating function (1) to generate the fade profile in real-time is structurally similar to implementing a fade-out audio effect that requires the computation of the output of rational function (3). Generically, a JavaScript implementation of the proposed method of fade-down shaping should include the construction given next.

Listing 1. The function designed for audio fading-down.

```
/* global scope: var ae, alphaD, betaD, gammaD;
   var fadeDown = false; */

function setVolD( tDref, tauDf, vDf, rhoD ) {
   var tauD = ae.currentTime - tDref;
   var vD0 = ae.volume;

   if ( fadeDown ) {
      var deltaD = ( tauD - alphaD ) / ( betaD * tauD - gammaD );
      var vD = vDf + deltaD;
      if ( vD > vDf ) { ae.volume = vD; }
      else { ae.volume = vDf; fadeDown = false; }
   }
   else if ( tauD >= 0.0 && vD0 > vDf ) {
      var deltaD0 = vD0 - vDf;
      alphaD = tauDf;
      betaD = ( 2.0 - 1.0 / rhoD ) / deltaD0;
      gammaD = tauDf / deltaD0;
      fadeDown = true;
   }
}
```

Since global variable “ae” of the provided code is created to refer the audio element (object), the invocation of function “setVolD()” will result in an audio volume updating whenever the playback position within the audio content is greater than the (expected) instant $t_{D,ref}$ of fade-down initiation and the output of function (1), denoted within the code by variable “vD”, is greater than the imposed final volume $v_{D,f}$, designated here by means of parameter “vDf” of function “setVolD()”. One perceives that when the value of (1), i.e. the value of local variable “vD”, is found less than or equal to the imposed final level that is the value of parameter “vDf”, the audio volume is set up just to the imposed final level and the fading-down process is stopped.

To avoid unnecessary valuations of (1), the structure encompasses the global variable “fadeDown”, which receives the value of “true” only when the playback position comes to be greater than or equal to the requested instant of fade-down initiation, denoted here by parameter “tDref”, and the detected audio volume, returned by the “volume” property of the audio object “ae”, remains greater than the value of parameter “vDf” that stores the imposed final volume.

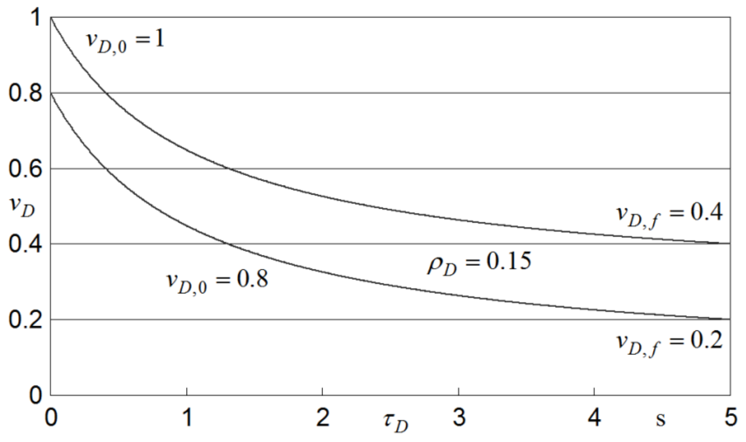

Fig. 1. Fade-down curves for fade length of 5 s, and ratio (5) of 0.15.

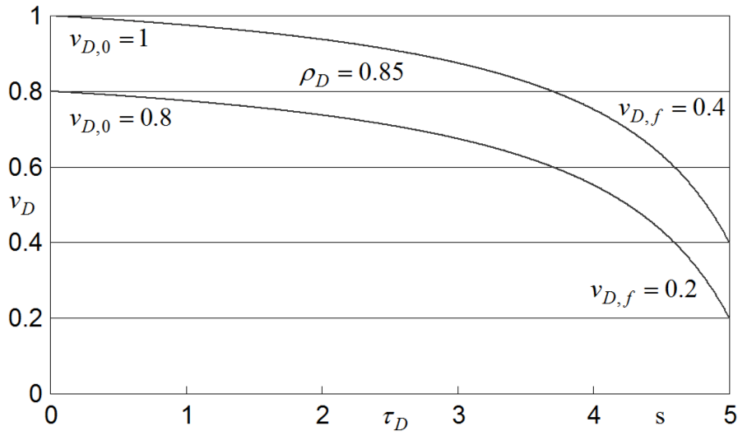

Fig. 2. Fade-down curves for fade length of 5 s, and ratio (5) of 0.85.

Taking into account (1), where function $\delta_D(\tau_D)$ is now provided by relation (3), in Fig. 1 and Fig. 2 we have plotted the fade-down profiles obtained for the fade length of 5 s, the initial levels $v_{D,0} = 0.8$ and $v_{D,0} = 1$, the final levels $v_{D,f} = 0.2$ and $v_{D,f} = 0.4$, and ratio (5) of value $\rho_D = 0.15$ and $\rho_D = 0.85$, respectively. One observes that, regardless of the imposed final level i.e. the value of $v_{D,f}$ in (1), the generated shapes of the fade-down sound effect are crucially decided by the value of quantity (5) that is the ratio between the value of (3) at the fade-down midpoint and the initial value of function (3), occurring at the fade-down initiation.

3. THE AUDIO FADE-UP CUSTOMIZING

Similar to the case of customizing the fade-down shape, we take into account that the evolution of the audio level during fading-up can be represented as the summation of a constant quantity and a function of the playback time. More precisely, we consider here that the function shaping the fade-up profile is brought forth by the summation of the initial audio volume $v_{U,0}$ and a function that technically describes a fade-in audio effect, i.e.

$$v_U(\tau_U) = v_{U,0} + \delta_U(\tau_U), \quad \tau_U \in [0, \tau_{U,f}], \tag{7}$$

where $\tau_{U,f}$ is the length of fade-up audio effect, while the relationship defining the independent variable

$$\tau_U = t_U - t_{U,ref} \tag{8}$$

plainly indicates that, with a view to software implementation, the instant of fade-up initiation has to be subtracted from the current playback time within the audio content (Lupsa-Tataru, 2019).

Since function $\delta_U(\tau_U)$ in (7) has to designate a fade-in audio effect i.e. a strict increasing of the audio level, starting from silence, we straightforwardly employ a rational function that proved to be suitable for real-time implementing. Thus, we deal with the following relation (Lupsa-Tataru, 2019)

$$\delta_U(\tau_U) = \frac{\alpha_U \tau_U^k}{\tau_U + \beta_U}, \quad \tau_U \in [0, \tau_{U,f}], \tag{9}$$

with $k \in \{1,2,3\}$.

To customize the shape of the fade-up effect, we account here for the ratio between the value of (9) at the fade midpoint and the value of (9) at the end of fading-up i.e.

$$\rho_U = \frac{\delta_U(\tau_{U,f}/2)}{\delta_U(\tau_{U,f})} = \frac{\delta_{U,h}}{\delta_{U,f}}; \quad 0 < \rho_U < 1 \tag{10}$$

or, considering relation (7),

$$\rho_U = \frac{v_U(\tau_{U,f}/2) - v_{U,0}}{v_U(\tau_{U,f}) - v_{U,0}} = \frac{v_{U,h} - v_{U,0}}{v_{U,f} - v_{U,0}}. \tag{11}$$

In order for rational function (9) and, implicitly, function (7) to be strictly increasing, the encompassed parameters get the specific expressions in terms of ratio (10) and the initial and final audio levels $v_{U,0}$ and $v_{U,f}$, where $v_{U,0} < v_{U,f}$ (Lupsa-Tataru, 2019):

$$k = \begin{cases} 3, & 1/8 < \rho_U < 1/4\,; \\ 2, & 1/4 < \rho_U < 1/2\,; \\ 1, & 1/2 < \rho_U < 1; \end{cases} \tag{12}$$

$$\alpha_U(\rho_U) = \begin{cases} \frac{4\rho_U}{8\rho_U - 1}\frac{v_{U,f} - v_{U,0}}{\tau_{U,f}^2}, & 1/8 < \rho_U < 1/4\,; \\ \frac{2\rho_U}{4\rho_U - 1}\frac{v_{U,f} - v_{U,0}}{\tau_{U,f}}, & 1/4 < \rho_U < 1/2\,; \\ \frac{\rho_U}{2\rho_U - 1}\left(v_{U,f} - v_{U,0}\right), & 1/2 < \rho_U < 1; \end{cases} \tag{13}$$

$$\beta_U(\rho_U) = \begin{cases} \frac{1-4\rho_U}{8\rho_U - 1}\tau_{U,f}, & 1/8 < \rho_U < 1/4\,; \\ \frac{1-2\rho_U}{4\rho_U - 1}\tau_{U,f}, & 1/4 < \rho_U < 1/2\,; \\ \frac{1-\rho_U}{2\rho_U - 1}\tau_{U,f}, & 1/2 < \rho_U < 1. \end{cases} \tag{14}$$

It can be observed that, similar to the case of customizing the fade-down audio effect, we have performed several auxiliary notations i.e.

$$\delta_{U,h} = \delta_U\left(\tau_{U,f}/2\right), \quad v_{U,h} = v_U\left(\tau_{U,f}/2\right),$$
$$\delta_{U,f} = \delta_U\left(\tau_{U,f}\right), \qquad v_{U,f} = v_U\left(\tau_{U,f}\right).$$

To facilitate the understanding, in Fig. 3 we have illustrated the fade-up curves received for the fade length of 5 s, the initial audio levels $v_{U,0} = 0.2$ and $v_{U,0} = 0.4$, respectively, the final audio levels $v_{U,f} = 0.8$ and $v_{U,f} = 1$, respectively, and ratio (11) of value $\rho_U = 0.85$.

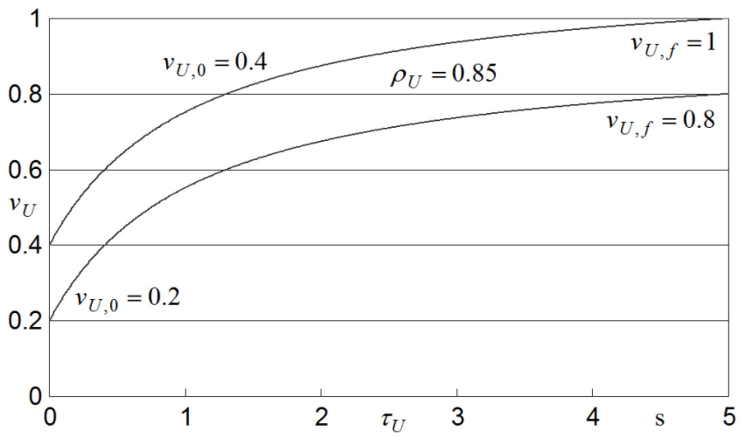

Fig. 3. Fade-up curves for fade length of 5 s, and ratio (11) of 0.85.

Structurally, the implementation of the fade-up audio effect by valuating the output of (7) is analogous to implementing the fade-in audio effect by employing the rational function (9), which proved to be suitable for fast processing in real-time (Lupsa-Tataru, 2019). Nevertheless, in the present case, the audio volume has to be updated each time the playback position of the audio content is greater than the instant $t_{U,ref}$ of fade-up initiation and the value of function (7) is less than the imposed final audio level $v_{U,f}$. As soon as the output of (7) comes to be greater than or equal to the final audio level $v_{U,f}$, the audio volume has to be set precisely to $v_{U,f}$, and the fading-up process has to be stopped in order to avoid subsequent evaluations of function (7).

Obviously, the computation of parameters (12)–(14), expressed now in terms of ratio (11), and the initial and final audio levels, has to be carried out only once that is the first time the playback position is found greater than or equal to the assumed instant of fade-up initiation. For instance, the JavaScript construction given next has been designed for the case of $1/2 < \rho_U < 1$ in representations (12)–(14). Anyhow, one easily perceives that the calling of function "setVolU()" leads to the computation of coefficients (13), (14), denoted by global variables "alphaU" and "betaU", respectively, only if the fading process is not yet started and the playback position, returned by the audio object "currentTime" property, comes to be greater than or equal to the requested instant of fade-up initiation.

Listing 2. The function designed for audio fading-up.

```
/* global scope: var ae, alphaU, betaU;
   var fadeUp = false; */

function setVolU( tUref, tauUf, vUf, rhoU ) {
   var tauU = ae.currentTime - tUref;
   var vU0 = ae.volume;

   if ( fadeUp ) {
      var deltaU = alphaU * tauU / ( tauU + betaU );
      var vU = vU0 + deltaU;
      if ( vU < vUf ) { ae.volume = vU; }
      else { ae.volume = vUf; fadeUp = false; }
   }
   else if ( tauU >= 0.0 && vU0 < vUf ) {
      var deltaUf = vUf - vU0;
      var auxVar = rhoU + rhoU - 1.0;
      alphaU = rhoU / auxVar * deltaUf;
      betaU = ( 1.0 - rhoU ) / auxVar * tauUf;
      fadeUp = true;
   }
}
```

4. CONCLUSIONS

The present investigation emphasizes the feasibility of customizing and implementing in real-time the fade-down and fade-up audio effects, having at hand techniques of shaping the fade-out and fade-in audio effects, which have been verified for the suitability with real-time computing (Lupsa-Tataru, 2018, 2019).

The audio fades customization is carried out here by taking into account that the evolution of the audio volume during a fading process can straightforwardly be described by the output of a function of the type:

$$v(\tau) = V + \delta(\tau) \tag{15}$$

where: τ – the difference of the playback time and the instant of fade initiation,
V – constant term,
$\delta(\tau)$ – rational function that depicts a fade-out or a fade-in audio effect.

It is pointed out that, by employing a relation of type (15) in order to shape the fade profile with the purpose of real-time processing, the implementation of the fade-down audio effect is similar to implementing the fade-out audio effect whilst the implementation of the fade-up audio effect becomes analogous to implementing the fade-in audio effect. In this context, the essential tasks required by the appropriate real-time implementations are highlighted in the course of presentation.

REFERENCES

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

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

Lupsa-Tataru, L. (2019). Implementing the fade-in audio effect for real-time computing. *Applied Computer Science, 15*(2), 5–18. doi:10.23743/acs-2019-09

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.