E-cecilia: implementation of a music game

Rubén Jesús García Hernández[1,5], Isabel Barbancho Pérez[2], Lorenzo José Tardón García[2], Jon Arambarri[3], Milán Magdics[1,4], Mateu Sbert[1]

[1] University of Girona, Girona, Spain,
[2] University of Málaga, Málaga, Spain,
[3] Virtualware,
[4] Budapest University of Technology and Economics, Budapest, Hungary
[5] iMinds / University of Hasselt, Hasselt, Belgium

Abstract. This paper describes a serious game intended to teach singing to children. The system shows the students a virtual world they can explore, and the evolution of the world is based on their performance. Automatic content generation tecniques are used to present the player with different exercises, and their singing is evaluated using state of the art techniques.

1 Introduction

There is a wide variety of music games [10, 17, 11], but most are focused on keyboard learning, identifying melodies by listening or reading a pentagram or following rhythm and practicing using mouse and keyboard interaction. Evaluating singing requires relatively costly sound processing and analysis, so few examples exist. Sony Singstar [24] evaluates pitching and timing, but is focused on singing complete songs and only runs on PlayStation 3.

The main obstacle in the first stages of learning music by children is the need to perform repetitive exercises during long hours. This can be boring for many children, which abandon their studies. To avoid this, we have developed a serious game which presents these exercises as means to advance in a videogame. The game presents a virtual world with a village which evolves as a function of the players singing performance.

The game is composed of several distinct modules, which will be described in detail in the following sections. Section 2 provides the general description of the game including the design and rendering of and interaction with the virtual world. Section 3 contains the description of the music generation and evaluation subsystems, including a description of related work. Finally section 4 provides some conclusions.

2 The virtual world

The developed music teaching game is designed primarily for children, thus we designed the virtual world, the game logic, the rendering and the interaction accordingly.

The defined scenario is a town, composed of several houses inside a green landscape. It was developed in Unity 4 using the Playmaker library for the implementation of the finite-state machine (FSM) model. The application consists of a scene with Prefabs where you can see several houses — each of them animated by a FSM created with Playmaker, some people modeled with an animation in Idle, and a field. Parameters handled by the application are within the field.

Fig. 1. The scene and the evaluation of houses.

The game is based on a client-server architecture, with three main components: A server, a client and a web server. The server runs on a Microsoft Windows platform and contains the recommendation subsystem and the evaluation subsystem. The client is based on the Unity engine and may run on Windows, OSX or mobile platforms. The communication between the client and the server is based on TCP sockets.

Additional resources are loaded by the client from a web-server which contains the songs, the pentagrams and additional text to describe the exercises. The algorithms for procedural song generation can be seen in section 3.3.

The server runs in a loop, accepting connections from the client, selecting the next three songs to be played by the client, then evaluating the client on the song they choose among the three (in order to give the player different paths to progress through the different stages of the game). A detailed description of the evaluation module can be seen in section 3.4. While the client stays connected, different songs are sent and evaluated repeatedly. After disconnect, the server waits for new connections.

With respect to the client, which runs in the user's computer or mobile device, the game starts by generating a virtual world for the user to explore and modify. Then, information is requested about the user, which is sent to the server and stored server-side for recommendation and evaluation purposes.

The user controls an avatar (in third person view) through which he can move in the virtual world, and can make the world evolve by selecting (i.e. clicking on) a house and singing well to make the house grow. When a house is selected, the three songs mentioned earlier are displayed to the user so that he can listen to

them and choose his preferred song. Then, a pentagram is shown while the song plays. After that, the user is presented with the pentagram and he should sing (or follow the rhythm, depending on the game mode) to the best of his abilities. The audio samples are sent to the server, and an evaluation is returned, which is used to give textual feedback to the user and to evolve the house (figure 2). After that, the user is free to continue the exploration of the world and choose either the same or a different house for further evolution.

Fig. 2. Singing to make houses evolve

The client architecture contains one thread to perform the network communication with the server, while the main thread is used to perform the main activities of the game. This last thread, controlled by the Unity engine, runs the rest of the scripts. A state machine controls the evolution of the game (see Fig. 3). The states are: Initializing, GetName, SendName, GetLevel, SendLevel, TraverseWorld, Ask3Exer, SendSong, AskExercise, PlaySong, RecordAudio, GetAutoeval, SendAutoeval and RecordRhythm, and they follow the game evolution described above. Since we want the user to play as much as he wants, so that he can practice and enhance his singing, there is no final state in the game.

2.1 Non-photorealistic rendering effects

As the game is primarily intended to be used by children and young students, we designed the rendering style accordingly. The virtual scene model uses simplified geometric shapes and textures. The rendering is made further cartoon-like using artistically motivated non-photorealistic rendering (NPR) effects.

Since the scene model was developed independently of the other parts of the game (including the rendering code), we have chosen to apply only post-processing NPR effects, as these are completely independent of the geometric or lighting model. The effects are executed on the output image stream of the standard rendering pipeline in image space, and are based on state-of-the-art image processing methods. Our supported NPR routines [12] include:

– Image simplification based on edge-preserving blurring and luminance quantization, that creates a cartoon-like style similar to toon shading. The amount of details removed is controllable by artists.

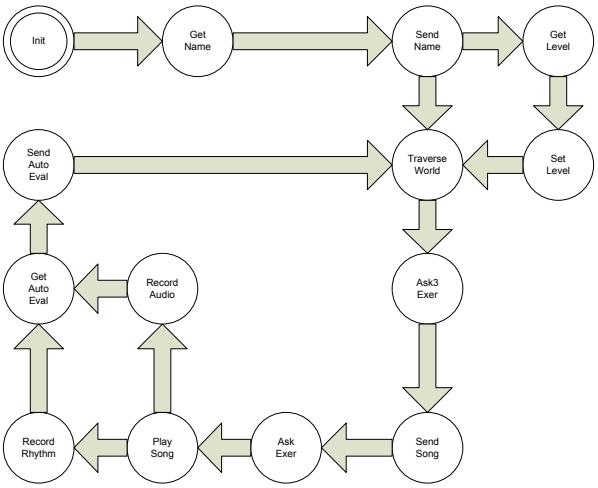

Fig. 3. State Diagram of the game

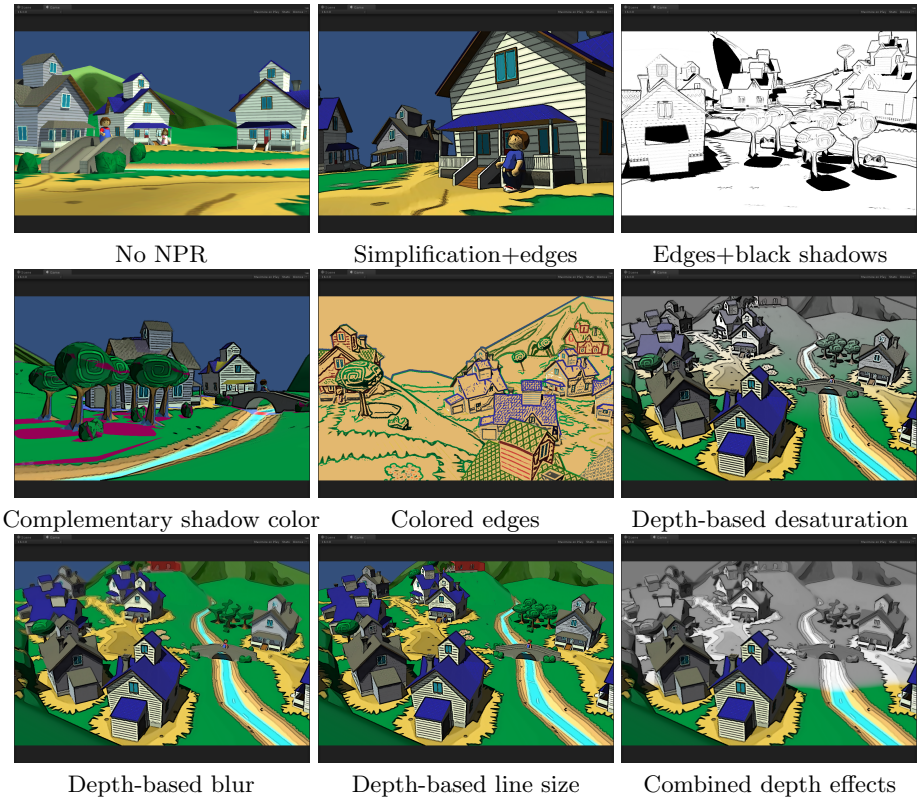

Fig. 4. NPR effects.

– Edge and contour enhancement based on standard edge detection methods performed either on the color, depth and normal buffers. Thickness and sensitivity of the lines are parameterized. Additionally, line color may be driven by the originally rendered color.
– Re-coloring the rendered shadows allows to change the contrast of the scene. Artists often create dark or completely black shadows, as well as using complementary colors.
– Illusion of depth with varying level of abstraction is commonly used by artists to enhance depth perception. We support varying amount of simplification, line thickness and color saturation. The depth level in focus, the speed of the transition and the number of discrete depth levels can be parameterized.
– Color transfer effects allow to transfer the color mood of an example image to the scene. The artists simply have to specify an exemplar image with the desired color distribution: the color distribution of the rendered image is modified to match that of the exemplar.

2.2 Advanced I/O

We are interested in creating a future-proof game which will be interesting for students in the future. Therefore, we studied how to integrate new interaction devices to provide high immersion and novel interaction. We extended the framework presented in [5] to support additional devices (cylindrical dome and Sony Virtual Reality helmet), to make sure that new devices can be added quickly, as immersive devices are becoming affordable (with a cost competitive with good monitors) and we expect their use to increase rapidly in the following years. Fig. 5 shows the game in the different immersive modes.

We also support the use of Microsoft Kinect gestures to navigate the virtual world and interact with the different objects. The mobile clients can also use Augmented Reality techniques to track the user's hands using the camera and evaluate the rhythm by having the user tap on virtual buttons. The accelerometers provided by the mobile devices allow navigation in the virtual world by tilting the device.

2.3 Implementation difficulties

Complex software developed by relatively-independent subgroups requires careful attention to the definition of the interfaces (the game API) among the different modules, up to a very low level of abstraction. However, when developing software at the edge or beyond the state of the art, it may not be possible to establish a-priori the necessary interfaces. Therefore, a prototype refinement approach is needed. Although the evolution of the API is expected as new functionality is added, some thought should be spent on designing extensible APIs so already-working functionality stays stable.

The unintended interaction among the different techniques can sometimes require specific checks (Fig. 6). As an example, some advanced NPR effects require access to the z-buffer, while the immersive post-processing shaders don't

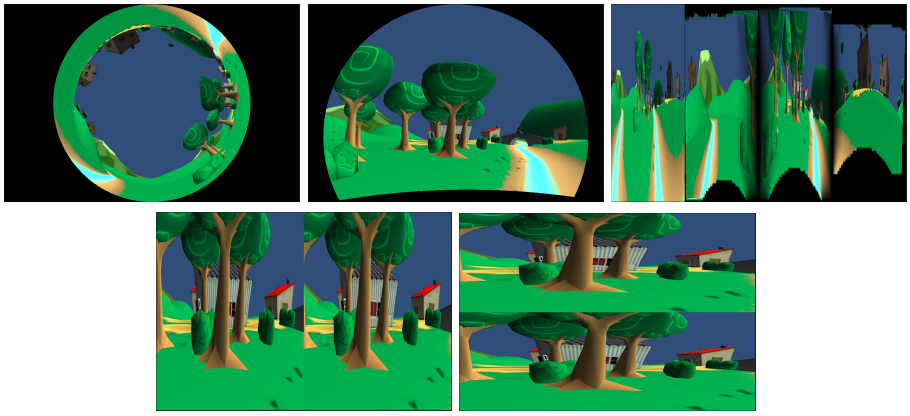

Fig. 5. Visualization in immersive devices: Spherical Dome, Immersapod, Cylindrical Dome, Sony Virtual Reality Helmet (side by side and top-bottom)

update the z-buffer for performance reasons (an update would require many passes over the scene to create a depth cubemap and the final z-buffer).

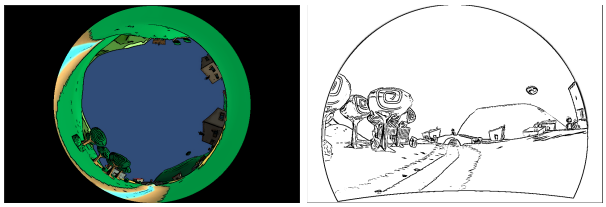

Fig. 6. Visualization using NPR and Immersive devices: spherical dome + colour comic, immersapod + black and white comic

Although the Unity engine advertises support for mobile platforms using the same codebase as the PC versions, we have found a few caveats which affected playability. The caveats might theoretically also apply in the other supported architectures, so the game is more robust in general after taking care of them. The problems in mobile platforms include:

- The internet connection provided by the mobile devices is very unstable, giving timeouts. The game was hardened to detect the situation, retry and give the user feedback.
- The support for advanced graphics is lacking, so some NPR / immersive effects were removed. Performance considerations were also a reason to remove some effects.
- The support for advanced I/O, such as accelerometers to control user movement, are not yet standarized, so the code had to take this fact into account in order to support the different architectures.

– Microphone use is CPU intensive, so the game can have lowered fps or sub-second freezes while recording. The problem is exacerbated by the fact that Unity forces the microphone access to be performed by the main thread which controls rendering, so latency cannot be hidden by the use of multiple threads.
– The support for playback of audio files depends on the architecture, so multiple versions of the songs in the MP3, OGG and MIDI formats are stored in the web server. The client accesses the specific format that it can play.

The use of augmented reality techniques for obtaining rhythm information from the user requires the use of a front-facing camera to provide an immersive experience. Therefore, this mode is specific to mobile devices, with PC clients using more classical mouse interaction for rhythm. In the future, we expect laptops to provide front-facing cameras; the interaction mode can then be activated in the PC version of the game.

3 Automatic music composition and singing voice assessment for music learning and entertainment

In this section, we present the methods and the structure of a computational scheme of music composition and analysis for music learning by interactive entertainment. The automatically generated singing exercises will be musically meaningful and adapted to the specific skill levels. The singing assessment scheme will be based on the comparison of the user's performance of an automatically generated singing exercise. In Fig. 7, a block diagram of the complete system is shown.

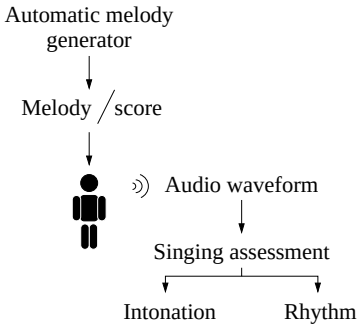

Fig. 7. Scheme of the complete system.

Next, we present related work on melody composition and automatic singing assessment. Then, an automatic generator of singing exercises will be depicted. Later, a scheme for automatic singing assessment will be briefly is described.

3.1 Related Work on Automatic Melody Composition

Music is commonly defined as 'organized sound' [6], thus, systems to generate music must be trained beforehand to learn the logic of the organization of the sound. Consequently, tempo and time signature analysis schemes can be found in previous works [25], [7].

Regarding the specific task of music composition, there exist different schemes based on pattern reallocation and variations. In [13] Markov models are used. Genetic algorithms have also been considered [14] as well as other methods based on probabilistic approaches [2].

Our system is based on the replication of minimal musical structures selected by means of a probabilistic analysis of music and a post-processing based on musical rules according to Western music theory rules and the expectation of Western listeners [19], [18].

3.2 Related Work on Voice Analysis for Assessment

Regarding the automatic evaluation of singing voice, a number of different schemes are found in the bibliography [24, 8, 9].

Typically, this type of application analyses first low-level features (F0, energy, aperiodicity...) in order to segment the audio signal into voiced and unvoiced regions. The most important low-level audio feature for the analysis in our context is the fundamental frequency $F0$ [3]. Later, a note-level segmentation process must be performed [20]. Finally, the segmented waveform will be analysed to assess the user's singing performance.

3.3 Music Composition Scheme

In order to design music composition methods, it is necessary to identify the parameters involved in the composition. Melodies of different complexity have been created by using the parameters extracted by analising excerpts corresponding to different academic levels [23].

Learning Musical Parameters The generation of musical contents is based on pattern repetition with harmonic variations [15]. Thus, rhythm patterns, pitch contours, harmonic progressions and tempo structures must be learned. In Fig. 8, a diagram corresponding to the learning scheme is presented.

Now, we consider the specific estimation stages.

Tempo estimation Tempo estimation can be successfully based on the work presented in [7]. Then, the duration of each measure can be easily obtained.

Time signature estimation The estimation of the time signature can be done using a a multi-resolution analysis scheme based on the analysis of bar repetitions [4].

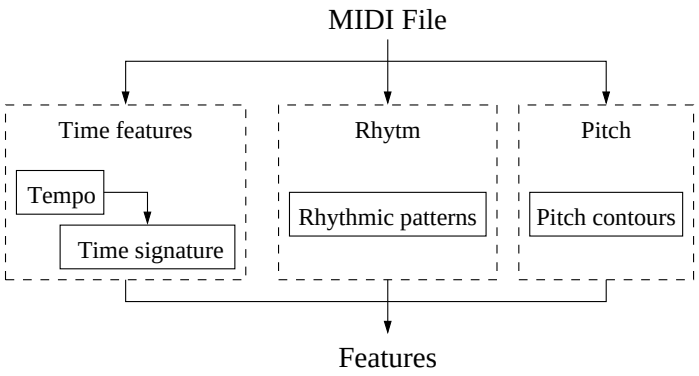

Fig. 8. Illustrative scheme of the music analysis system.

Rhythm Patterns Rhythm is a main musical feature closely related to the structure of music [1]. The parameters described previously can be used to quantize the duration information from an input MIDI file and relate the intervals to figure durations.

The pitch contours of the rhythmic patterns obtained will be stored. Later, the patterns with more contour versions will be selected with higher probability than others by the composition system.

Pitch Progression The pitch contour [1] will be used for music generation. A harmony corrector is used to adapt the melody to the chord progressions.

Melody Generator The Melody Generator will create new melodies that replicate the style or complexity of the songs previously analysed. First of all, the initial tonality, the time signature and the number of bars must be chosen. Then, the patterns selected according to their estimated probability of appearance will be harmonically adapted [22]. Fig. 9 shows a schematic representation of the stages of the melody generation algorithm.

Now, the stages of the melody generation scheme will be briefly described.

Pattern Selection The items stored with a specific user-selected time signature will be pre-selected. Then, those required to build the rhythmic structure (ie. A-B-B-A) will be chosen. Then, a pitch contour is selected randomly among all the pitch contour versions in the database for each of motive.

Harmony Progression A harmonic progression that sounds well will be selected [19]. Finally, a chord transposition scheme will adapt the patterns selected to certain music theory rules.

Chord Transposition The chord transposition stage must modify the sequence of notes so that the desired harmony is followed [22].

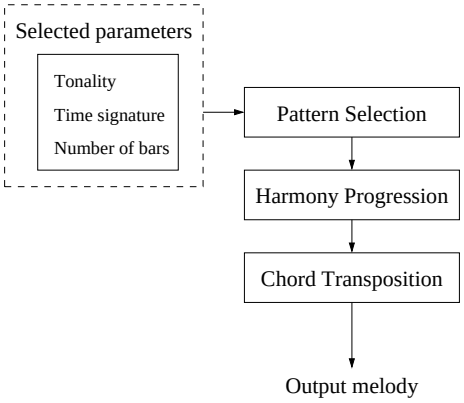

Fig. 9. Scheme of the melody generation system.

3.4 Singing Assessment Scheme

We analyse the user's singing performance by comparing the processed audio against a reference melody. The analysis scheme developed for this game has been found to attain very good performance using inexpensive desktop microphones (like the Trust MC-1200, or similar ones). A scheme of the system is illustrated in Fig. 10. Next, we will briefly describe the steps in the scheme in Fig. 10.

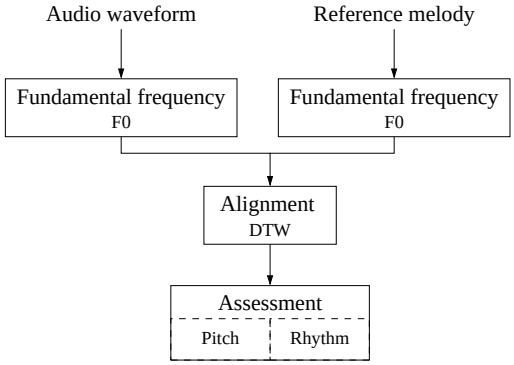

Fig. 10. Singing assessment scheme.

Fundamental frequency (F0) extraction The Yin algorithm [3] is used to extract the F0 vector (temporal sequence of F0 values). Additionally, the original paper by de Cheveigné introduces the aperiodicity measure, which is useful to perform the segmentation of the audio waveform into voiced/unvoiced frames [16].

Assessment of singing voice The F0 of the user's performance must be compared against the reference melody. A suitable method to align these functions is Dynamic Time Warping (DTW) [21]. The path of the optimal alignment contains the necessary pieces of information for singing evaluation. Specifically, the cost of the optimal alignment path found can be used for intonation assessment. On the other hand, the analysis of the deviations of the alignment path with respect to the ideal path (diagonal straight line) provides the desired rhythm assessment information [16].

4 Conclusions

In this paper, a complete computational scheme for singing learning using a serious game has been described. The game includes a system for the automatic generation of exercises and a system for the automatic assessment of the user's singing performance. The music generator analyses sample melodies to learn the necessary parameters to automatically generate new melodies corresponding to a music level suitable for music learning. The method for the automatic assessment of singing voice is based on the comparison of the estimated fundamental frequency of the user's performance against the reference fundamental frequency of the automatically generated melodies, evaluating both intonation and rhythm.

These systems interface with a virtual world modelled in the Unity game engine, which provides a catching display using attractive non-photorealistic techniques, novel interaction methods and virtual reality displays to enhance expressiveness, increase game immersion and prevent boredom when performing repetitive exercises.

We expect the system to reduce student drop-off. For future work, we would like to perform a user study to assess the usefulness of the game in real-world scenarios.

5 Acknowledgements

This work has been supported by the research projects IPT-2011-0885-430000, TIN2013-47276-C6-1-R, TIN2013-47276-C6-2-R (Spanish Ministry of Science and Innovation), 2014 SGR 1232 (Catalan Government) and FP7-ICT-2013-10-610005 (European Union).

References

1. Appel, W.: Harvard Dictionary of Music. The Belknap Press of Harvard University, Cambridge, Massachusetts, 2nd edn. (2000)
2. Cope, D.: Computer Models of Musical Creativity. MIT Press Cambridge (2005)
3. De Cheveigné, A., Kawahara, H.: YIN, a fundamental frequency estimator for speech and music. Journal of the Acoustical Society of America 111(4), 1917 (2002)
4. Gainza, M., Barry, D., Coyle, E.: Automatic bar line segmentation. 123rd Convention of Audio Engineering Society Convention Paper (October 2007)

5. García, R.J., Magdics, M., Rodríguez, A., Sbert, M.: Modifying a game interface to take advantage of advanced I/O devices: a case study. Advances in Intelligent Systems Reseearch 58, 128–132 (2013)
6. Goldman, R.: Ionisation; density, 21.5; integrales; octandre; hyperprism; poeme electronique. Musical Quarterly 47(1), 133–134 (1961)
7. Gouyon, F., Herrera, P., Cano, P.: Pulse-dependent analyses of percussive music. Proceedings of ICASSP 2002 4, 396–401 (2002)
8. Hoppe, D., Sadakata, M., Desain, P.: Development of real-time visual feedback assistance in singing training: a review. Journal of computer assisted learning 22(4), 308–316 (2006)
9. Jin, Z., Jia, J., Liu, Y., Wang, Y., Cai, L.: An automatic grading method for singing evaluation. Recent Advances in Computer Science and Information Engineering pp. 691–696 (2012)
10. Learning Games for Kids: Art games and music games for kids. http://www.learninggamesforkids.com/art_and_music_games.html
11. LucasFilm: Loom. http://www.old-games.com/download/1434/loom
12. Magdics, M., Sauvaget, C., García, R., Sbert, M.: Post-processing NPR effects for video games. In: 12th ACM International Conference on Virtual Reality Continuum and Its Applications in Industry (VRCAI 2013) (2013)
13. Merwe, A., Der, V., Schulze, W.: Music generation with markov models. IEEE Multimedia 18(3), 78–85 (2011)
14. Miranda, E.R., Biles, J.A.: Evolutionary Computer Music. Springer (2007)
15. Molina, E.: Hacer música... para aprender a componer. Eufonia. Didáctica de la Música (51), 53–64 (2011)
16. Molina, E., Barbancho, I., Gomez, E., Barbancho, A., Tardon, L.: Fundamental frequency alignment vs. note-based melodic similarity for singing voice assessment (2013)
17. Music Learning Community: Music learning community - free preview – learning music through games. http://www.musiclearningcommunity.com/FreePreview.htm
18. Narmour, E.: The Analysis and Cognition of Melodic Complexity: The Implication-Realization Model. University of Chicago Press (1992)
19. Ottman, R.: Elementary Harmony: Theory and Practice. University of Michigan Press, USA (1998)
20. Ryynänen, M.: Singing Transcription. Springer Science + Business Media LLC (2006)
21. Sakoe, H.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing 26, 43–49 (1978)
22. Schellenberg, E.: Simplifying the implication-realization model of musical expectancy. Music Perception 14(3), 295–318 (1997)
23. Sierra, F.: Lecciones de Entonación. Real Musical (1992)
24. Sony Computer Entertainment Europe: Singstar (2004)
25. Uhle, C., Herre, J.: Estimation of tempo, micro time and time signature from percussive music (2003)