Karen Collins: Application for Associate Professor
Audio Engineering Society "AES for Games" 2015 conference paper

VEEMIX: INTEGRATION OF MUSICAL USER-GENERATED CONTENT IN GAMES

KAREN COLLINS[1], ALEXANDER HODGE[1], RUTH DOCKWRAY[2] AND BILL KAPRALOS[3]

[1] *Games Institute, University of Waterloo, Waterloo, ON, Canada*
collinsk@uwaterloo.ca
[2] *Department of Performing Arts, University of Chester, Chester, UK*
r.dockwray@chester.ac.uk
3 *Faculty of Business and IT, University of Ontario Institute of Technology, Oshawa, ON, Canada*
bill.kapralos@uoit.ca

Musical user-generated content (UGC) in games is usually disconnected from gameplay, running in the background rather than being integrated into the game. As a result, players may lose their emotional connection to the game's narrative or action, disrupting immersion and reducing enjoyment. Here we describe a system for integrating user playlists, what we have termed musical UGC, into games. The system, Veemix, allows for the simultaneous integration of music into games using keyword tags and social ranking, while allowing for the collection and storage of semantic data about the music that can be used for music information retrieval purposes. We outline two iterations of the system in the form of a Unity plugin for iOS using streaming and user device music.

INTRODUCTION

Game music has historically been viewed as repetitive and disengaged from game-play. In part as a response to this repetitive looping history, game composition has developed to be dynamic; interactive with both the player's actions and to game-play events. In real time, compositions adjust to in-game parameters, as transitional sequences or cross-fades bring the music from section to section to reflect game events.

Despite the current attempts by composers to create such dynamic soundtracks for games, many players still play games for many hours at a time, and the repetition of the music remains a distraction for some. One solution to this problem, user-customizable music, became a part of game hardware systems around the time of the sixth generation of video game consoles (Xbox and PlayStation 2)[1]. With musical user-generated content, the player is able to connect their music device, and the game music is shut off while the player's music plays instead. This external music is incorporated into a game by playing it as "background music," rather than integrating it into the game. For composers and designers, this trend implies a step backwards with respect to the idea of the music being an integrated and integral part of the game. Moreover, the designers and composers must relinquish control over the musical soundtrack, and therefore in many cases relinquish control over the affective, emotional aspects of the game.

User-generated content (UGC) in games is increasing [2], and solutions are required to better integrate this content into games for more cohesive, engaging and enjoyable gameplay. Here we present the Veemix system, our solution to musical UGC's problems. The remainder of the paper is organized as follows: Section 1 provides background information and the issues regarding musical user-generated content, followed by outlining previous approaches. In Section 2, details regarding our motivation and approach that we have taken are provided. The first iterations of Veemix are introduced in Section 3; A small user study is presented in Section 4; and finally, concluding remarks and plans for future research are presented in Section 5.

1 BACKGROUND

User-generated content is a growing trend in all games (online and offline), and increasingly an important revenue consideration for game companies. However, *how* to incorporate musical UGC as a soundtrack into a game is still an area of debate. In the last decade, console makers such as Microsoft have made it mandatory for developers to allow users to customize their in-game music; that is, the player must be able to plug in their MP3 player and listen to their own music instead of the game music. Musical UGC also occurs as players use their Voice Over IP (VoIP) microphones to input music into a game [3], by streaming music (over the headset) into the game or by setting up in-game chat channels dedicated to music. The desire to set up one's

own streaming channel has led to in-game VoIP radio systems where players can broadcast their own music, effectively becoming virtual world DJs.

Players have also developed customized modifications or "mods" for individual games that let them substitute their own music for the pre-composed music. Players then discuss what music they're listening to on Internet forums. In fact, we found that players were discussing what music they were *going* to listen to in games before the games had even been released.

1.1 Problems with Musical User-Generated Content

While UGC is often desired by players, game designers may be reluctant to provide players with such capabilities, as they may lose control over the emotional content of the game [4]. While there are certainly many positive aspects to customization, the user-led changes in functionality and aesthetics of musical UGC can alter the intended meaning of a scene (see, e.g. [4][5]).

Many questions remain regarding the impact that musical UGC may have on the player's experience of a game. Wharton and Collins [4] found that musical UGC altered the perception of immersion, as well as the way that players played the game; that is, their tactics, and their general movement through the game were affected. Most importantly, the change in music altered the "feel" of the game, and the emotional connection to the narrative. Another issue with musical UGC is that players are not always effective at choosing appropriate songs. Wharton and Collins [4] examined the influence that a player's choice of music had on their experience. Despite many players being familiar with the game and incorporating their own selection of songs with which they were familiar, players were not able to consistently predict and select their own musical songs to increase immersion or enjoyment. Players often chose songs by attempting to match *lyrics* to game events, rather than using the *music* to capture an emotion, mood, or general feeling of a game scene. In other words, players lacked experience in understanding the complex interplay between music and a game, and they were unsuccessful in choosing music that would help them to achieve a desired effect.

Indeed, there are many aesthetic consequences to allowing musical UGC. The music composed for games is designed to fit with the aesthetic framework of the world. By allowing players to customize music, this fit can be disrupted, and new meanings and emotional relationships can be generated. Players may choose inappropriate music that contradicts rather than reinforces actions or events, thus turning a potentially moving scene into a comedy, for example. But perhaps the most significant problem with current musical UGC implementation is that the music is played as *background music*, rather than being integrated into the game in any meaningful way.

Despite the problems with musical UGC, players are actively engaged in selecting and sharing musical ideas with other players: there is clearly an equally compelling if different type of experience that occurs with the ability to select and share music in games. The practical problem we are attempting to solve, then, is to match game content with appropriate songs from the device or from a streaming service, in order to maintain some control over the affective, emotional, and semiotic content of the game. In solving this practical problem, we aim to develop insight into the relationship between musical UGC and players' experiences of games.

1.2 Previous Approaches

There have been very few previous attempts to integrate musical UGC into games in a meaningful manner. Rossoff [6] adjusts a playlist to a game by altering the tempo of a selected song to the rate of the gameplay using beat extraction and tempo adaptation. This is, however, unsatisfactory as changing the tempo of one's music may significantly alter the song's mood in a negative way (e.g., vocals will sound odd, phasing effects will be distracting). McGowan [7] used the Echonest API to automate the selection of music to tie to emotional content of the games. While automation is desirable, in this case tying to the content in the game was tedious (the player had to tag the game with emotions and fine-tune selections manually).

Similarly, Deliyannis et al [8] employed an automated system to embed musical preferences into games via connecting the music to a musical framework. The system relies on streaming music from social software sites such as Last.fm to determine the player's musical preferences. While this may, then, select music that the player enjoys, it does not solve the problem of integrating that music with the game's affective, emotional content.

Of course, music recommendation systems for non-game media also exist, using a variety of methods and for a variety of purposes [9][10][11], including commercial recommendation systems to find, for

instance, the appropriate mood for the user.[1] There has also been the opposite approach, where music for games is procedurally generated based on in-game events [12]. To our knowledge, however, there is no existing system that incorporates musical UGC into games by tying the music to game events using tags defined by the game's designers.

2 A NEW APPROACH TO MUSICAL UGC: VEEMIX

Given the benefits of incorporating musical UGC in games, in addition to the problems with current attempts, we have recently begun investigating game-based musical UGC. We have three related goals:

1) Better integration of songs into the game

Rather than playing as "background music" (where the music is not tied to specific locations or events), we believe that it is more effective to select songs (or have songs selected) specifically for game events, thus tying music to action in the game, rather than risk those semiotic mismatches described above. These events are chosen by the game developer and tagged with keywords. For example, a level may begin with "dark, industrial, scary" as keyword tags.

2) Helping the player to make better song selections

As found in Wharton and Collins [4], players became frustrated by their inability to select appropriate music. There are (in our opinion) two primary solutions to this problem: The first involves automatically choosing songs for the player based on audio extraction and recommendation systems. We have not yet implemented this feature for on-device songs, but have implemented such a feature in our streaming iteration. Our second approach is to allow players to share their playlists through a social network. Players can download playlists that other players have uploaded inside the game. Playlists can be rated by other players with a simple thumb-up "like" system, meaning the playlists that are most preferred by other players are easier to find. Moreover, the developers themselves can flag recommended playlists if they find that playlists are suitable to their game, or if they themselves wish to put together a few playlists. In a sense, this function allows developers to choose songs for their game without having to license the songs, since they are not distributing the music themselves.

[1] For example, StereoMood (www.stereomood.com), and Musicovery (www.musicovery.com)

3) Collecting rated data on musical UGC for music information retrieval purposes

Information regarding the songs selected for each tag and their ranking by players are being stored, to allow for future improvement with an automated recommendation system. Currently, beyond purely structural aspects (tempo, chords, key, and so on), extraction of auditory information in music for such automation and recommendation systems has focused primarily on *mood* (e.g. [13]). However, there are several problems with relying strictly on mood when it comes to certain search needs. For example, if we wanted to search for music inside a videogame, many popular types of games are limited in the scope of their mood (First Person Shooters for instance are typically built around vengeance and anger). Secondly, even given a specific mood in a game, the context is as important as the emotion ("angry" set in ancient Rome should be musically different than "angry" set in a futuristic world). Thirdly, there are many musical pieces that don't fit well into any single, clearly defined mood. Finally, some songs change significantly in mood as they progress (e.g. Led Zeppelin's "Stairway to Heaven"). There are also aspects of music that are not well served by mood extraction, such as intertextual references to other music or media (e.g. a whistle at the start of Kid Rock's song 'Cowboy' references the opening sequence of *The Good, The Bad and the Ugly*). Such intertextual references may make for interesting synchronization with certain types of games.

Mood is, of course, not the only way to understand musical meaning [14]. It may be possible to combine existing work on mood extraction with genre extraction (see e.g. [15][16]), although genre extraction also has its own problems (categorization being the most significant, but also genre may not relate to what is musically relevant about a piece). What is required, then, is to explore other means of extracting audio information from music, focused on those elements that may be meaningful.

Our project explores meaning in terms of *musemes*. A museme is a 'minimal unit of musical meaning', roughly analogous to a morpheme in linguistics [17]. They are, in other words, units of musical signification. An example of a museme is the first four notes of Beethoven's Fifth Symphony. The phrase is said to represent fate, knocking on the door. One might, in other words, wish to search for the keyword "fate" in music to find such a result.

A museme-based approach to musical information retrieval represents several benefits to the user over a mood or genre-based search. In particular, the user has

the option of more advanced search terms. For example, many stock music companies allows for searching by keywords such as “dark”, “epic”, “fantasy”, “power” amongst others. At the moment, these music pieces must be tagged manually in their meta-data, and if the right keyword is not included, the piece will not be found. Advanced keyword searches are also beneficial for recommendation systems, similarity finding, and musical analysis. Whereas existing tag-based approaches to music information retrieval are based on the entire song, a museme-based approach can apply to a section of a song, allowing for more accurate search. It can also search for the elements inside the song that drive the keyword: for instance, the minor add 9 chord is associated with “madrigalistic woe” [18], and thus we can search for “sad”, or we can search for a more specific *kind* of sad (“madrigalistic woe”), or we can search for the entire museme (“a minor add 9 chord”).

Despite these advantages, there are also some significant challenges with a museme-based recommendation engine. Musemes are complex constructions that may include multiple musical elements such as harmonic or melodic content, rhythm, timbre, and so on. Therefore, a keyword search may require multiple structural components (e.g., timbre, rhythm and harmony) to be examined at once. More challenging, however, is that there is currently no established database of musemes. Collecting existing known musemes involves a time-consuming search of music analysis literature and collating this information, a process which is an ongoing component of our research program.

Our Veemix database, storing music with keyword tags chosen by the game designer, along with songs chosen by the user or by our system, therefore, is in part serving as a means of collecting these musemes. By storing song titles with the keywords that people associate those songs with, we can use our database in the future as a learning set for developing new means of extracting museme information from audio files. If enough data can be collected, then for example, songs that are “dark, industrial, stressful” (Figure 2), can be analyzed for the characteristics that lead to those keyword associations.

Collecting useful annotated data on music using games or social interaction is nothing new of course [19][20][21]. However, this is, to our knowledge, the first time that a plugin that spans multiple games has been used to collect data that may be useful for music information retrieval purposes.

3 IMPLEMENTATION

Veemix has currently been developed with *Unity3D* for iOS games. *Unity3D* was selected given that it is currently the most popular platform for independent developers, it has an established plug-in marketplace, and it offers a cross-platform base that deploys to most major game devices. We chose to implement the system first for iOS given that the iPod music player application is standard across iOS devices, unlike other platforms where music may be stored in different locations (there are many Android music players, for instance, and the user’s music files may be stored in a number of different folders).

Unity3D has a front-end Inspector that enables users to set and adjust public variables, meaning that the developer does not have to do any programming, only set the scripts in the right place by dropping an empty cube mesh where they want the event to occur, and then give the event a name and keywords (Figure 1).

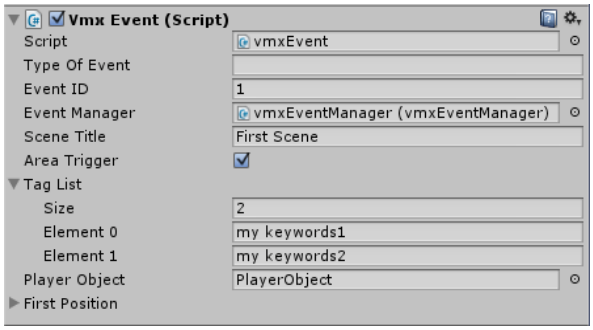

Figure 1. The *Unity3D* Inspector component of the Event script.

Our *Unity3D* scripts were written in C#. The main classes for the *Unity3D* side of the implementation. This involves a *Receiver* script for the player object that handles the options (turn Veemix on or off, turn input touch gesture control on or off), and *Event* scripts that manage the tagged events. As shown in Figure 1, the Event scripts store the event ID number, and allow the developer to specify a name for the event (“Scene Title”), as well as determine how many tags they want to use for that event, and specify those tags.

The rest of the programming is handled on the iOS side, using Objective C via a C bridge to *Unity3D*. Although it would have made future porting to other platforms easier to keep most of the scripting inside *Unity3D* (which is designed as a cross-platform engine), we found that the GUI handling was considerably improved by moving the GUI to iOS, and likewise we found audio processing speeds and playback capabilities improved if these are handled by iOS.

The social networking component was built using PHP, XSLT, XML, JSON, MySQL and a web site. When the player selects a song, we gather all of the relevant track information about the piece (from the iTunes or SoundCloud API), and then add to that music data the event information (tags, game title, etc.). In the future, then, it will be possible for users to query the existing playlists not just for the game that they are looking for, but also by tags. Game developers (who must be registered) can recommend a playlist for their specific game, which sets a binary flag in the database and prioritizes the recommended playlist in the user's view of playlist suggestions.

We created two iterations of Veemix: one using songs on the player's device (iPod, iPad), and one using a streaming service. The player has the option of selecting one or the other at the game's loading.

3.1 On-Device Songs

The first iteration of Veemix uses songs on the player's device, and allows the player to select a song based on a the keyword tags that they uncover.

When the player encounters an event for the first time (though a simple collision detection with an invisible cube mesh placed by the developer where the event should occur), they are provided with the series of tags describing the type of music anticipated by the developer to suit that event. An example is shown in Figure 1.

Figure 2. A Veemix event name with three tags describing the scene. The player first sees just the rightmost orange button, which can be dragged out to see the tags and select the songs.

The music is queued immediately once the player has selected their song(s) for that event. By turning the event-finder off, the player can then have those songs automatically queue the next time that they encounter that event. In other words, if the player's character dies and the level is restarted, the song will play automatically. Thus, while the initial play-through may be disrupted by selecting songs, replaying will be much smoother. When a song is playing and a new event is encountered, the songs will cross-fade over a two-second period. In this way, if the player suddenly encounters for instance a boss level, the player's music can ramp up to reflect the change in status quickly but without being too jarring.

For copyright reasons, the music files are not included with downloaded playlists. The software runs a scan of the player's device to determine if they have the songs available. If they do not, the player is provided with links to the iTunes store to make a purchase (and can sample the 30 seconds provided by iTunes). The downloaded playlists are also editable, so players can select new songs to fill gaps in downloaded playlists for which they have no song. Each event is saved with the tags and a screenshot of the event in the editable playlist screen (see Figure 3). Players can upload their playlists to our server, giving the playlist a name and description. Other players can then "like" the playlist, thus increasing the playlist's ranking.

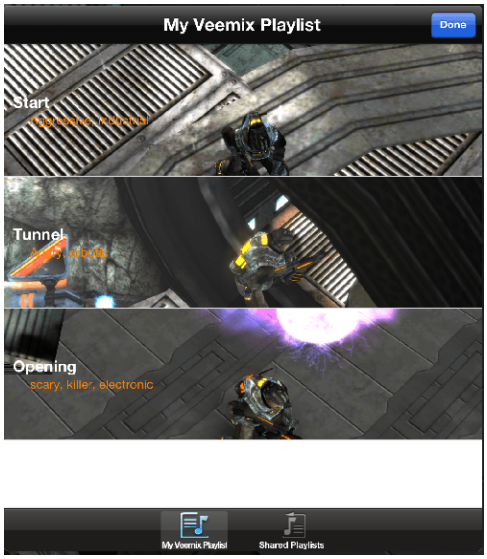

Figure 3. Playlists are editable and show event names, keywords and screenshot. Selecting the event allows the player to choose songs.

3.2 Streaming Songs

Understanding that music streaming is an increasingly important component of the mobile device marketplace, we created a second iteration of Veemix, incorporating a streaming service, *SoundCloud*. *SoundCloud* was selected because it has a large user base (over 200 million listeners), does not require a log-in to the end user to stream music, and is not restricted here in Canada where our development is taking place (*Spotify* and *Pandora*, for instance, are not available in Canada due to licensing regulations). Future work will include the option of selecting streaming service and incorporating other APIs, particularly since the *Spotify*

API is integrated with EchoNest, allowing for some complex and interesting possibilities.

Unlike the "uncover tags and select song" approach taken in the first iteration of Veemix, the streaming version automatically finds songs for the user based on the developer's input tags and a query of the *SoundCloud* API. In other words, when the player encounters an in-game event, the plugin will fetch a song to stream for them. When a user of *SoundCloud* uploads a file, they can tag that file with as many as thirty keyword tags. These tags are then searchable through the *SoundCloud* website or API. However, one of the difficulties with the existing API is that searches are done by using *or* statements: If there are two tags, "scary" and "industrial", for instance, the query finds songs with tags matching "scary" *or* "industrial", rather than matching both "scary" *and* "industrial." Currently, the plugin's web services returns a payload of up to 150 streaming song URLs from multiple API queries using the first three tags for a given game scene. The plugin then loads and begins playing a song using a random selection from the streaming payload.

Figure 4. Using Veemix in streaming mode presents an interface for gamers to rate the song served by the plugin.

While playing within the boundaries of a Veemix event set by the cube mesh, the player is able to accept or reject the plugin's selection of a streamed song for the given scene. Accepting the song (using the "thumbs up" icon) continues playback and adds the song to the Veemix Playlist, while rejecting the song (using "thumbs down") skips to the next candidate song from the SoundCloud payload.

Streaming playback is implemented using the iOS AVPlayer class which is suitable for HTTP Live Streaming of media. However, the AVPlayer class does not support real-time audio mixing for HTTP streams, and as a result the transition from one song to another is done without the crossfading available on the on-device song version of Veemix. The AVPlayer class does, however, expose some degree of real-time control over the stream, for example the playback rate can be set at initialization or during playback, thus speeding up or slowing down the song. The application and usefulness of such a control in gameplay has not been fully explored with this iteration of Veemix. It is our hope that a streaming class that allows for fading in and out will soon be implemented.

In the streaming iteration of Veemix, we do not show the user the tags as described in Figure 2; however, users can still view this information and save, edit and change songs in their playlist. The Veemix Playlist created using the streaming mode is independent of the device-based playlist, and switching between modes will also change the content of the player's playlist.

4 PRELIMINARY USABILITY STUDY OF VEEMIX

In order to determine the effectiveness of Veemix in improving the incorporation of musical user-generated content into games, we ran a small user study to compare both the on-device and streaming service iterations of Veemix. The study was approved by the University of Waterloo Office of Research Ethics.

Participants consisted of ten undergraduate and graduate students at the University of Waterloo, all between the ages of 20 and 35, and enrolled in a sound design course (at the undergraduate or graduate level). Eight out of ten of our participants were avid gamers (playing every day or nearly every day). Seven out of ten of the participants sometimes or usually listened to their own music while playing games.

Participants played Veemix implemented into the *Unity3D* "Angry Bots" demonstration game, a third-person science-fiction shooter. Participants then completed a short questionnaire consisting of ten questions using a five-point Likert scale. Questions consisted of five general player background questions (did they own a console, how often they play games, and habits relating to music in games), and five specific questions about our implementation of Veemix (relating to sharing music, interface, and preferences). Participants were also presented with the opportunity to provide comments and feedback.

Our results were split in terms of preference for streaming or on-device song selection, with two preferring on-device song selection, two preferring streaming, and the rest saying that they liked both streaming and on-device song selection equally. Seven out of ten said they would definitely share their playlists with other gamers, and two stated that they would be unlikely to want to share playlists (one had no opinion).

In terms of feedback, the bulk of comments were very positive, with many participants saying that selecting songs added to the enjoyment of the game, and "add a

lot to the player experience in terms of interest and mood, and make the game more personal and engaging". Participants did, however, describe the following potential issues:

1) Selecting songs takes the player out of the gameplay for a moment. This is not an issue with the streaming version, and if we incorporate automation in the future, we will provide the option of automating the selection of on-device songs.
2) Switching between songs in the streamed version was occasionally jarring, since there was no fade out. We are exploring solutions to streamed music fading for future iterations.
3) One participant wanted to select the location in the game where the player could choose songs, rather than leaving that up to the developer.

Participants also noted that the system would work better in some genres than others, with multiplayer games being the favourite choice.

5 CONCLUSIONS

Although incorporating musical user-generated content in games can provide many benefits to the player including increased enjoyment and engagement, the implementation of musical UGC is often overlooked. Implementing musical UGC can be difficult and currently no best practices for doing so exist. Given the benefits of musical UGC and the lack of successful musical UGC approaches, our work has examined a new approach to musical UGC, leading to the development of Veemix, a novel musical UGC system introduced here. Veemix offers a number of benefits over playing musical UGC in the background of a game. We believe that Veemix represents a more effective integration of musical UGC into a game by allowing the player to tie music to in-game events. In addition, the player has the opportunity to learn from other players with regards to being able to select songs that are appropriate to games, essentially learning the skill of the "music supervisor" (the person in films/television that selects music), by sharing playlists, by ranking playlists, and simply through trial and error of their own songs as they play back inside a game.

Despite the benefits of Veemix, work is ongoing, with many future improvements planned. In addition to amassing a large amount of data suitable for developing an automatic song recommendation system, we plan to add several features to Veemix in the future. For example, we are exploring ways of manipulating the songs in real-time inside the game using DSP effects, in order to create an even better fit with the game event. We can adjust songs by sampling and altering the songs in real-time (for instance, sample-and-hold, much like a DJ with a record), and by employing filters and DSP effects like reverb, equalization and phasing (see, e.g.[24]). For example, one could imagine a player slipping underwater in a game; a low-pass filter can be applied to make the song sound like it is being emitted underwater. A similar approach has been applied in the Electronic Arts game *SSX*.[2] The difficulty here is to optimize the processing effects such that they create no perceptible delay on the output, particularly with the mobile devices we are now targeting.

As discussed above in Section 2, one of the aims of Veemix is to examine where and how people make selections of music for particular actions and events. As such, we are storing the playlists (songs selected, tag keywords, and game information) in our database for future use. Not only will this knowledge enable us to create advanced intelligent automation solutions for on-device songs, but by examining the music selected for particular keywords, we can advance music information retrieval techniques beyond mood and genre to incorporate musemes. It is our hope, in other words, that through player use of Veemix, we can increase our understanding of music and image more generally.

6 ACKNOWLEDGEMENTS

The authors would like to thank the Social Sciences and Humanities Research Council of Canada and the Canada Foundation for Innovation for the financial support for this project.

7 REFERENCES

[1] B.L. Schmidt, D.A.W. Pickford and D. H. Smith. Music Replacement in a Gaming System. US Patent No. 7,663,045 B2. Feb 16, 2010.

[2] G. Lastowka."User-Generated Content and Virtual Worlds", *Vanderbilt Journal of Entertainment and Technology Law*, vol. 10, no. 4, pp. 893–917, 2008.

[3] M.R. Gibbs, K. Hew and G. Wadley. "Social Translucence of the Xbox Live Voice Channel", In *Proceedings of ICEC 2004 3rd International Conference on Entertainment Computing*, pp. 377–385. September 1–3, in Eindhoven, NL, 2004.

[2] http://www.audiorecordingschool.com/blog/2012/07/19/eas-realtime-user-music-remix-system-rumr/

[4] A. Wharton and K. Collins. “Subjective Measures of the Influence of Music Personalization on Video Game Play: A Pilot Study” *Game Studies* ,vol. 11 no. 2, 2011. http:// http://gamestudies.org/1102/articles/wharton_collins

[5] A.J. Cohen. “How Music Influences the Interpretation of Film and Video: Approaches from Experimental Psychology”. In *Selected Reports in Ethnomusicology: Perspectives in Systematic Musicology 12*, pp. 15–36. Los Angeles: UCLA Ethnomusicology Publications, 2005.

[6] S. Rossoff. “Adapting Personal Music based on Game Play” MSc thesis, Northwestern University, 2007.

[7] J. McGowan. “Harmonious: An Emotion-Matching System for Intelligent use of player’s own music libraries with game soundtracks.” MSc project, Leeds Metropolitan University, 2011.

[8] I. Deliyannis, I. Karydis and K. Anagnostou. “Enabling Social Software-Based Musical Content for Computer Games and Virtual Worlds.” *4th International Conference on Internet Technologies and Applications, (ITA2011)*, Wrexham, UK. September 2011.

[9] A. Nanopoulos, D. Rafailidis, P. Symeonidis, and Y. Manolopoulos, “Musicbox: Personalized music recommendation based on cubic analysis of social tags”, *IEEE Transactions on Audio, Speech & Language Processing*, vol. 18, pp. 407- 412, 2010.

[10] C. Basu, H.Hirsch, and W. Cohen. “Recommendation as Classification: Using Social and Content-Based Information in Recommendation” *AAAAI-98 Proceedings*, http://www.aaai.org/Papers/AAAI/1998/AAAI98-101.pdf, 1998

[11] O. Celma. *Music recommendation and discovery: The long tail, long fail, and long play in the digital music space*. Berlin: Springer, 2010.

[12] J. Robertson, A. D. Quincey, T. Stapleford, and G. Wiggins, “Real-Time Music Generation for a Virtual Environment”, presented at the ECAI-98 Workshop on AI/Alife and Entertainment, Brighton, England, 1998.

[13] D. Liu, L. Lu and H.J. Zhang. “Automatic mood detection from acoustic music data”. In: *Proceedings of the International Symposium on Music Information Retrieval, 26-30 October 2003, Baltimore, Maryland*. ISMIR, 2003

[14] P.N. Juslin and J. A. Sloboda, (eds.). *Handbook of music and emotion: Theory, research, applications*. Oxford University Press, 2010.

[15] J.J. Burred and A. Lerch. “A hierarchical approach to automatic musical genre classification”. In *Proceedings of the 6th International Conference on Digital Audio Effects’ 03*, 8–11, 2003.

[16] G. Tzanetakis and P. Cook. “Musical genre classification of audio signals” *Speech and Audio Processing, IEEE transactions on*, vol. 10 no.5. 293-302, 2002.

[17] P. Tagg. *Kojak: Fifty Seconds of Television Music—Toward the Analysis of Affect in Popular Music*. New York: Mass Media Music Scholars’ Press, 2000.

[18] P. Tagg. “Film Music, Anti-Depressants and Anguish management”. IASPM-AL Conference, Rio de Janeiro, June 2004. http://tagg.org/articles/jochen0411.html

[19] P. Lamere. “Social tagging and music information retrieval”. *Journal of New Music Research*, vol 37 no. 2, pp.101-114, 2008.

[20] E. Law and L. Von Ahn. “Input-agreement: a new mechanism for collecting data using human computation games.” *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2009.

[21] D. Turnbull, R. Liu, L. Barrington, and G.R. Lanckriet. “A Game-Based Approach for Collecting Semantic Annotations of Music.” In *Proceedings of ISMIR 2007*, vol. 7, pp. 535-538, 2007.