Harmonious: An Emotion-Matching System for Intelligent Use of Players' Own Music Libraries with Game Soundtracks

Jim McGowan
j.mcgowan4782@student.leedsmet.ac.uk
jim@bleepsandpops.com

Masters project for:
MSc Sound and Music for Interactive Games
Leeds Metropolitan University

1.CONCEPT AND RATIONALE

In recent years it has been stated in numerous works, both scholarly [18, 22] and otherwise (for example [20]) that many video game players are eschewing the original musical soundtrack in games in favour of listening to their own music, either via music players built into the game platform or external music players, such as iPods. This phenomenon seems surprising considering the efforts made by game developers to create engaging musical soundtracks that can variously adapt to changing game conditions, interact with the player and otherwise aim to enhance the gameplay experience [3]. Indeed, many game composers expressed surprise and disbelief when discussing this phenomenon [7].

By replacing game music, players are deprived of any information or feedback on the game that is provided by such adaptive and dynamic soundtracks.

1.1The Prevalence of Replacing Game Music

None of the work discovered on this topic detailed specific figures on which and how many gamers were replacing game music. To this end a survey was carried out online that questioned gamers about their musical habits.

This survey was carried out with web-based software. The link to the survey was provided in forums and social networks focused on games and/or used by game players. Discussion forum messages were posted on these sites informing the members of the phenomenon being investigated, inviting members to take the survey and inviting any further discussion on the topic within the forum.

The headline result of the survey was this:

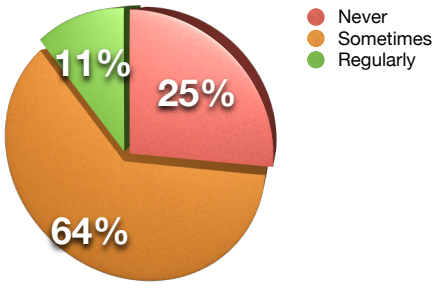


Figure 1: Percentage of players who turn off and replace game music

25% of the respondents never turn off game music, 64% sometimes turn off game music, and 11% turn it off regularly. Breaking this down by age, it was found that turning off game music is more common with younger players.

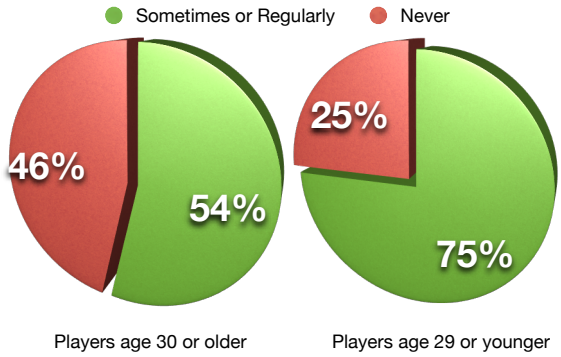


Figure 2: Percentage of players who turn off or replace music by age group

However, it must be noted that the older age group only accounted for 10% of the total number of respondents, so this result is not definitive. A greater difference can be seen from the perspective of how frequently the respondents play games, but again the less frequent group only accounts for just over 10% of the total respondents:

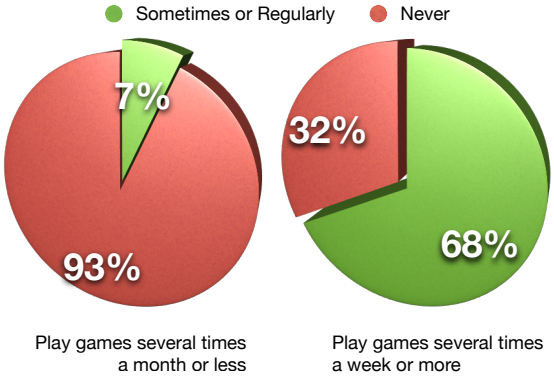


Figure 3: Percentage of players who turn off or replace music by play frequency

The survey asked respondents if there were particular genres of games in which they were more likely to turn off or replace music. The results were as shown in figure 4:

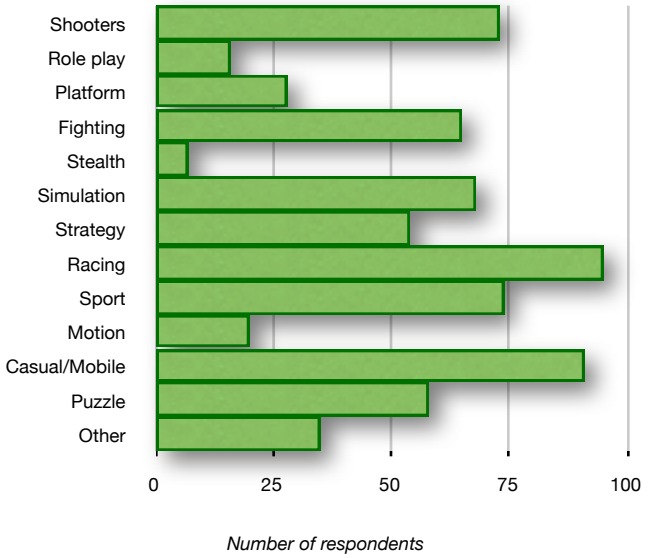


Figure 4: Genres in which players are more likely to turn off or replace music

Unsurprisingly casual/mobile games are frequently muted, which is very likely due to environmental factors. However Racing was the most commonly indicated genre, with 55% of the respondents choosing it. Similarly the survey asked if players were more likely to turn off music in single or multiplayer games. Multiplayer was the most common response.

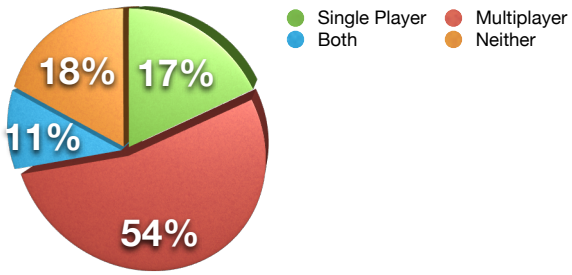


Figure 5: Game play types in which players are more likely to turn off or replace music

Discussions with the respondents started to indicate some common reasons why players turn off game music. A common opinion was that game music is too repetitive, or there simply isn't enough musical material to remain interesting over the course of gameplay. Racing games, shooters, role play and MMOs[1] were commonly charged with this complaint. Another frequent complaint was that music was not enhancing or reflecting gameplay. Multiplayer shooters, MMOs, puzzles and casual games were often cited with this viewpoint. Correspondingly many respondents said that if they felt the music was critical to a narrative they would always listen to it, though many noted that this was only the case on the first play through a game, and subsequent plays would often have the music replaced.

One surprising trend that emerged in discussion with the respondents was that a number of players who turn off game music replace it with spoken word material such as audiobooks or podcasts.

Given the forums used to invite survey participants, the respondent population shared many common demographic qualities - specifically most respondents were very interested in video games and were regular players. The majority of respondents were in their teens or twenties. Therefore, the survey results may potentially be skewed towards the opinions of the so called 'hard-core' gamer, and a wider population may be required to gather data which reflects attitudes of other types of player. Additionally, as discussion around the survey topic was invited in forums, the survey data may suffer from the "diffusion" effect [4]. However, the survey results can be seen as confirming the assertions in other works regarding players eschewing game music.

The complete survey data can be found in Appendix 1

1.2 Issues with Replacing Music

With the design of their XBox 360 game console, Microsoft built in technology that allows players to mute and replace music in games, without affecting the playback of other audio material, such as voice overs and sound effects [18]. Microsoft require that all games released for this console make use of this technology. Other game platform manufacturers, such as Sony, Nintendo and Apple, have similar technologies in place but do not currently enforce their use [1, 22]. Manufacturers are clearly aware of players' desires to replace game music, but the issue remains that information and feedback are being provided to players via game music through cueing, adaptive composition and other techniques. When the music is muted, this information is lost to the player.

2. PREVIOUS WORK

There has been no previous work found that directly addresses this issue, however, there exists a small amount of work that is relevant to understanding the music replacement phenomenon. The earliest such work was by Morris in 2003 [16], a conference paper proposing that multiplayer FPS[2] games should be considered as co-creative media, as the game experience can potentially be customized and personalized by players on each play. This customization could range from swapping out music, to creating new maps and mods[3] for games. As such the paper challenges the notion of a published game as final, complete work, similar to that of a film auteur - rather that it is the beginning of a co-creative

[1] Massively Multiplayer Online games

[2] First Person Shooter

[3] Mods are user/player made modifications of existing games that can range from the creation of new levels or scenarios to 'total conversions', where content and gameplay are completely replaced.

relationship between developer and player. From such a perspective original game music can be considered as a suggested soundtrack for a game, and that player-supplied replacement music can be considered as an equally valid part of the gameplay experience.

In 2005 Microsoft's XBox 360 was released, the first of the seventh generation of games consoles[4]. At the 2006 Game Developer Conference, Brain Schmidt, Architect of the XBox 360 audio system, claimed Microsoft's own research had shown that significant numbers of gamers replaced game soundtracks with their own music [18], though specific figures from this research were not revealed. Schmidt claimed that it was as a result of this research that the XBox 360 system features facilitating music replacement, and the QA requirement that games make use of these features, were developed.

In 2008 Jørgensen [10] did a study on playing games with the entire soundtrack muted: every sound produced by the game, including music, sound effects, and dialogue was removed. Unsurprisingly he found the experience was grossly crippled, as without audio cues, particularly UI and notification sounds, it was easy to miss details in the game.

However the only directly relevant work was a study by Wharton & Collins published in 2011 [22]. This study examined the experience of players when they had free choice over the music to accompany their gameplay. Players played a level from Fallout 3: Operation Anchorage on the XBox 360. At checkpoints in the game players were given the opportunity to choose music from their iPods, MP3 players or CDs.

The key finding of this study was that "players were unable to predict what music would improve their immersion, but were able to choose appropriate music to influence game playing tactics and anxiety levels" [22]. Significantly, the study also found that players still considered the 'replacement' music as an integral part of the game experience, reinforcing Morris' thesis.

3.OVERVIEW

The Harmonious system is a piece of software which can be integrated into video games to facilitate the replacement of game music with players' own music, whilst addressing the issue of loss of information. To this end the objectives of the system design were as follows:

- Allow players to listen to music in their own libraries whilst playing games
- Automate the choice of music from the player's library by matching gameplay mood with the emotional content of the music, thereby:
- Retaining some of the information and feedback provided by the original game music

In order the meet these objectives, the Harmonious system has been designed as shown below:

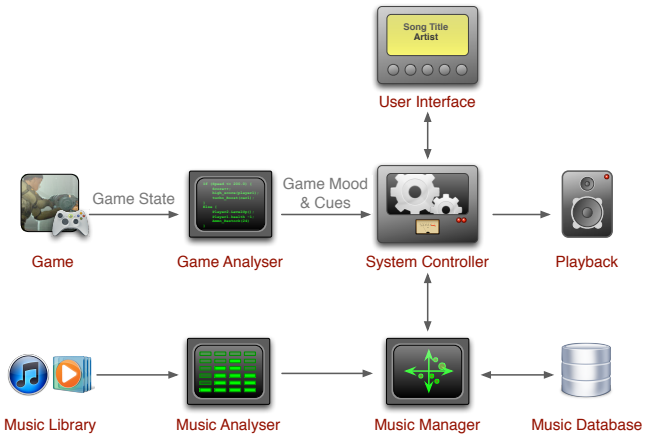


Figure 6: Harmonious system diagram

In this system, a music analyser iterates through the player's music library and determines the emotional content of each piece of music, along with supplementary data, such as key, tempo, bar and section locations, and metadata, such as title, artist and duration. The results of this analysis are stored in a database. Access to this data from the rest of the system is provided via a music manager object.

The system controller object presents both the programmer's API[5] to the game developer and the user interface to the player. The system controller receives cues for music to start, stop and change, and a representation of the current mood of gameplay from a game analyser object. When a new musical selection is required the system controller passes this mood value to the music manager, which determines the best matching music from the player's library. The system controller then plays back this piece of music.

The only component of the system that is game-specific is the game analyser object. This object examines the game's state and determines the current mood of the game, therefore this requires a game-specific implementation. However, a standard interface between a game analyser object and the system controller has been defined.

The controls exposed to the player include standard play/pause and skip controls, as well as controls that can fine tune the mood-matching process and 'teach' the database about the player's preferences.

4.REPRESENTATION OF EMOTIONS

The system relies on having a means of representing emotions in the digital realm. Models for representing emotions can be separated into two types: categorical and dimensional [11]. In a categorical system, analysis of music would result in each piece being allocated to a particular category or categories. The resolution of the system is therefore dependent on the number and type of categories. The model used by the annual MIREX[6] music emotion recognition competition is a typical categorical system and is shown in table 1 below.

[4] The others being the Sony Playstation 3 and the Nintendo Wii, both released in the following year

[5] Application Programming Interface

[6] The Music Information Retrieval Evaluation eXchange, details can be found at http://www.music-ir.org/?q=node/13

Clusters	Mood Adjectives
1	passionate, rousing, confident, boisterous, rowdy
2	rollicking, cheerful, fun, sweet, amiable/good natured
3	literate, poignant, wistful, bittersweet, autumnal, brooding
4	humorous, silly, campy, quirky, whimsical, witty, wry
5	aggressive, fiery, tense/anxious, intense, volatile, visceral

Table 1: MIREX categories from [11]

This approach may be useful in applications such as music streaming services, where the primary concern is the relationship between pieces of music. However, it is limiting in the context of this project, where emotional content of music is matched with emotional content of gameplay, as there is no way to distinguish between different pieces of music in the same category - which song in the 'happy' category is the happiest?

The dimensional approach to emotional modeling maps emotions on a plane, where the axes represent various emotional characteristics, and the continuous nature of the plane eliminates ambiguity as each point represents a unique emotion or level of emotion. The 2-dimensional Thayer/Arousal-Valence model is a simple and effective example of a dimensional model [17, 19]

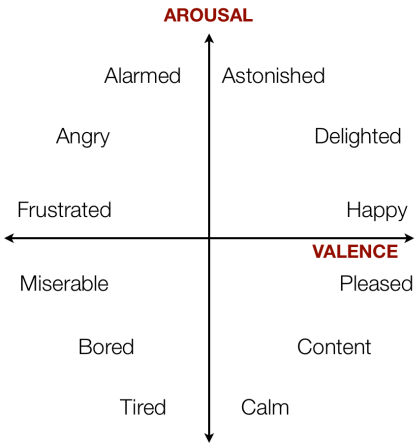


Figure 7: Locations of emotions on a 2-dimensional Aoursal/Valence plane

Though models with larger numbers of dimensions also exist, this 2-dimensional model has proven very effective in musical applications (see [11, 17, 23]).

In this model emotions are represented on a two-dimensional plane, where the axes represent *arousal* (level of energy) and *valence* (positive or negative feeling). Individual emotions are therefore represented as locations or areas on this plane. In the context of the Harmonious system this model offers a significant advantage: it removes the need for the use of subjective language. Gameplay is evaluated in terms of its arousal and valence; likewise music is also evaluated in terms of its arousal and valence. With such data, matching and refinement remains within the mathematical domain, and is not hampered by the translation of such data into subjective language terms, such as 'happy'.

A potential downside to this approach is that various studies have reported that valence is much more difficult for automated systems measure/evaluate than arousal, giving courser grained results. But given the widespread adoption of this approach it is quite likely that improvements will continue to be made in this area.

To maintain a simple and consistent reference throughout this project, the arousal-valence plane is considered to have axes extending 2 units in length, from -1 to 1. The application of unit length to such a dimensional representation is, of course, arbitrary. This range was chosen for simplicity and clarity in calculations.

5.MUSIC ANALYSIS

Several approaches to music analysis were examined for suitability with this project. Many researchers are taking a computational approach to MER[7], making use of custom processes built with lower level MIR[8] tools such as Marsyas[9], LibXtract[10] and the MIR Toolbox[11], or by constructing completely bespoke systems (for examples see [15, 17, 23]). The accuracy of such systems is typically between 45-80% for dimensional emotion representations [11]. However such systems can be extremely computationally expensive and as such the time taken to analyse several thousand tracks on a desktop-class computer system would be prohibitive for a project such as this.

A second approach is mining a pre-existing database of human created descriptions of music. Such databases come in two types: expert and crowd. Projects such as the Music Genome Project[12] and the All Music Guide[13] have databases of tags applied to pieces of music that have been carefully determined by music experts, and present a valuable source of data. However, these expert databases generally remain private or accessible only via commercial

[7] Music Emotion Recognition

[8] Music Information Retrieval

[9] http://marsyas.info

[10] http://libxtract.sourceforge.net

[11] https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox

[12] http://www.pandora.com/mgp.shtml

[13] http://www.allmusic.com

contract, due to the cost and effort of producing them. Crowd-based music databases, such as last.fm[14] are usually more open and accessible. Laurier et al recently proved that for basic emotional tags, such as happy, sad, angry and tender, crowd- or social-based music databases can be reliable sources [12]. However, two issues remain with both expert- and crowd-based databases: No database can contain data on every piece of music that the Harmonious system (or any other) is likely to encounter; and using language tags as a datasource for a geometry-based system requires a subjective mapping between language and numeric values. Such a mapping can only be as accurate as the values shared between the mapper and the tagger. For the Harmonious system it would be preferable to determine arousal and valence values by direct audio analysis.

5.1 Echo Nest

It was therefore found that Echo Nest would provide a effective analysis platform for this project. Echo Nest is a music information system that provides a backend for online music services and is used by organizations such as the BBC and MTV. The Echo Nest system is available free of charge for non-commercial uses. The platform combines a server-based music analyser that mixes computational and textual approaches, and a database of "tens of millions" [5] of pre-analysed pieces of music. This gives a large set of existing analysis data that can quickly be retrieved and should hopefully cover a high percentage of the music encountered by the Harmonious system. For music that has not already been analysed by Echo Nest, the system can fall back on using the server-based analyser. This analyser has the advantage of running on dedicated, powerful hardware, and can typically analyse a 3 minute piece of music in under 5 seconds. The analysis time is therefore only limited by the time taken to upload the music to the server.

5.2 Harmonious Analysis Workflow

The process by which the Harmonious system analyses a player's digital music collection consists of three stages:

1. Correctly identifying the music contained in each audio file
2. Fetching Echo Nest's analysis data for that music
3. Parsing the analysis data to create mood shapes (areas of the arousal/valence plane indicating the emotional content of the music) for inclusion in the system database

The first stage is crucial to the entire system. If a piece of music is incorrectly identified, the wrong analysis data will be associated with the audio file, defeating the ability of the system to correctly match music with gameplay. Even the differences between two versions of the same piece of music (for example, and album version and a radio edit) can be significant enough to spoil the data, due to timing differences and other factors.

Harmonious makes use of three methods of identifying a track, starting with the most efficient, and incrementally falling back to the least efficient when necessary (here efficiency refers to the processing time taken by each method). The first method is audio fingerprinting. An audio fingerprint (sometimes referred to as an acoustic fingerprint) is a content-based 'signature' that summarizes the audio content of a digital recording [2]. Identifiable fingerprints can be created from highly compressed audio files, and fingerprinting algorithms are tolerant of audio distortion. Harmonious makes use of an open-source fingerprinting implementation called EchoPrint. As the name suggests, this implementation creates fingerprint codes that are compatible with the Echo Nest platform, and can be used as queries against the Echo Nest database.

Specifically, the Harmonious fingerprint identification process is as follows:

1. 60 seconds of the audio file are read into memory as PCM[15] samples
2. The array of samples is passed to the EchoPrint code to create a fingerprint.
3. The fingerprint code is send to the Echo Nest server to query its database
4. If a match if found for the fingerprint in the database, the matched metadata (artist, title, etc) is retrieved
5. The retrieved metadata is tested against metadata from the local audio file to prevent false positive matches. If the metadata matches to within a fine tolerance, the identification is considered a success.

It was through experimentation that the duration of 60 seconds was found to be the most effective. Counter-intuitively, longer audio segments do not lead to increased accuracy in matches. It was found that for durations of up to 60 seconds, increased duration led to a greater number positive matches. However, with durations over 60 seconds the number of false positives increased greatly with longer durations. 60 seconds was found to give the best balance.

If identification by fingerprinting fails, the system falls back on identification by metadata. In this case the system extracts artist, title and duration data from the audio file (if available). The artist and title strings are used as a search query against the Echo Nest database. This often results in multiple results, as the search engine uses very loose string matching (for example, an artist search for the string "Elvis" may return both "Elvis Presley" and "Elvis Costello" as possible matches).

To make the correct identification amongst the returned possibilities, the artist, title and duration values for each potential match are compared against those from the local audio file. Duration is simply compared numerically, and the difference, in seconds, is noted. The artist and title strings are each compared with fuzzy string matching using the Levenshtein Distance Algorithm [24]. This approach is used to allow the system to correctly identify a pair of strings such as "nick cave & bad seeds" and "Nick Cave And The Bad Seeds" as a positive match. The sum of the two string matching scores and the duration difference is taken as an overall match score, where a lower score indicates a closer match. The system defines a minimum tolerance, and a match score below this tolerance is considered to be a positive identification.

If identification by metadata fails, the system falls back on uploading the audio file and having Echo Nest perform a bespoke analysis. This approach is the most time consuming of the three

[14] http://last.fm

[15] Pulse Code Modulation: uncompressed digital audio

identification methods, but the system can be assured of the accuracy of the analysis results.

After a track has been identified (or uploaded for analysis) Harmonious fetches the full analysis data for the track from the Echo Nest database. This data contains the following information:

- •Track Metadata
- •Duration
- •Time Signature
- •Key
- •Mode (Major or Minor)
- •End of Initial Fade In
- •Start or Final Fade Out
- •Beat Locations
- •Bar Locations
- •Section Locations (sections being high-level divisions, such as verse, chorus, bridge, etc)
- •Segment Locations (segments being short divisions of relatively uniform timbre and harmony, typically less than 1 second)

Included with each segment location are the following:

- •Loudness (in dB)
- •Timbre (as an array of 12 values, each representing a sonic property)
- •Pitch (as an array of 12 values representing the relative dominance of each pitch in the chromatic scale)

(Full descriptions of the analysis data can be found in [6], from which the foregoing list information is summarized).

The first operation carried out with this data is to split tracks into segments of potentially differing emotional content. ('Segments' seemed the most appropriate term for these divisions, and throughout the Harmonious system, in source code and descriptions, these divisions are referred to 'music segments'. However, this clashes slightly with Echo Nest's use of the term 'segment' for the very short tonal based divisions it identifies. To limit confusion, throughout this text the term 'Echo Nest segment' is used when referring to the latter). Echo Nest's section divisions (high-level divisions, such as verse, chorus, etc) provide a convenient framework in which to search for potential significant changes in emotional content.

For each high-level section identified by Echo Nest, the average loudness and timbre values are calculated by iterating through that section's component Echo Nest segment data. Once this data has been collected for each section, each section's averages are compared with those of the sections immediately preceding and following. Experimentation with several known pieces of music revealed that a difference greater than 6.0 in both average loudness and average timbre was a signifier of a potential difference in emotional content. Harmonious then splits the track into music segments at such points of change.

Harmonious then makes a pass through the assembled music segments whilst examining the bar location information for the track. The segment boundaries are each adjusted to lie on bar boundaries, to allow for cleaner transitions at playback time.

After further testing, however, it was found that the criteria used to determine segment breaks would often identify occurrences such as a long sustained chord or note at the beginning or end of a track as a music segment. During gameplay, this would issue manifest itself as playback of several very short pieces of music (often under 10 seconds) amongst the longer musical segments. This was found to be distracting and confusing. Therefore the system limits segments to a minimum of 30 seconds in duration to prevent this.

After each music segment has been identified, an initial circular representation of the segment's emotional content on the arousal/valence plane is created. (These circular shapes are referred to in the source code as 'mood shapes'). The valence value is determined by the mode and mode confidence analysis data. Mode is represented as either major or minor, in the musical sense. Mode confidence is represented as a value between 0.0 and 1.0, with 1.0 indicating complete confidence in the analyser's assessment of the music's mode. If a segment's mode is major, it's valence value is set equal to the confidence value. If a segment's mode is minor, its valence value is set to zero minus the confidence value. This causes clearly major or minor music to be placed towards the extremes of the valence axis, while more ambivalent music tends towards the neutral centre.

The arousal value is determined by the segment's loudness and energy values (energy is a value between 0.0 and 1.0 representing Echo Nests determination of how energetic the music is. This data is provided in the summary information received the point of track identification, rather than in the full analysis data). The energy value is used as the basis for the arousal value, and then modified based on the variation in the segment's loudness values from the overall average loudness of the complete track.

The arousal and valence values are used as the co-ordinate of the centre of a circular mood shape, the radius of which is set to 1 minus the mode confidence, giving segments with lesser confidence a larger scope to be fine-tuned by the user/player (the fine tuning process is discussed in section 5).

All music analysis operations are carried out in the background of the host game's process, allowing the game to be played and to make use of the Harmonious system while analysis is taking place. Analysis operations can also be suspended and resumed when the host game is quit and restarted. The most time consuming aspect of the analysis workflow is the uploading of unidentified tracks for bespoke analysis by Echo Nest, therefore the overall time taken to analyse a music collection depends predominantly on the number of files that require uploading. During testing it was found that an average of 11% of the music encountered by the analysis system required uploading, with 39% being matched with fingerprints and 50% being matched with metadata. Typical time taken to analyse one thousand songs is around 6 hours.

Appendix 2 shows the analysis workflow summarized in a flowchart.

5.3 Measuring Analysis Accuracy

The accuracy of the mood shape analysis is dependent on the accuracy of the Echo Nest analyser. However, the means by which the Echo Nest analyser operates and determines the output values it produces is proprietary and kept in confidence by the company operating the service. There is therefore no means to validate the analyser's methods against other work in the field. However, the analysis results can be validated by measuring them against standard reference sets used in other MIR[16] and MER[17] studies.

A test was devised making use of the CAL500 dataset as a reference. This is a set of 500 pieces of popular western music from the last half-century with human-applied descriptive tags, created by Turnbull et al in 2008 at the Computer Audition Laboratory, University of California San Diego [21]. The set has been used by its authors and third parties as a reference set for various MIR experiments.

Of the 174 possible tags that may be applied to a piece of music in the CAL500 dataset, there are 36 tags describing emotional content. As an initial test of the Echo Nest platform, a comparison between these tags and Echo Nest's mood search was carried out. For each track in the CAL500 dataset the corresponding emotion tags were collected into a set. A dictionary was created mapping the terminology used by Echo Nest's mood search to the corresponding emotion tags from CAL500. A mood search operation was carried out for each Echo Nest mood and limited to music that appears in the CAL500. The results were used to create a second set of emotion tags for each track, representing the emotional content of the track as determined by Echo Nest. Comparing the two sets for each track, the results were as follows:

- 80.07% of the set pairs intersected
- The average complete match between set pairs was 28.74%
- The average complete match between intersecting pairs was 35.89%

This high rate of intersection was encouraging, so so a further test was carried out to compare arousal/valence mood shape representations between CAL500 and Echo Nest. However, as the CAL500 dataset only has textual representations, a means by which to translate these into shapes on the arousal/valence plane was required. To this end a vector was assigned to each of the 36 emotion tags representing the position of the emotion tag on the arousal/valence plane, where the vector origin lies at the origin of the plane, as shown in figure 8 below.

For each track in the dataset the emotion tags were extracted and the average of the corresponding vectors was calculated. The position indicated by this average vector was taken as the centre of a circle. The radius of the circle was based on a confidence value derived from the number of emotion tags applied to that track. This circle then represented the emotional content of track in arousal/valence terms.

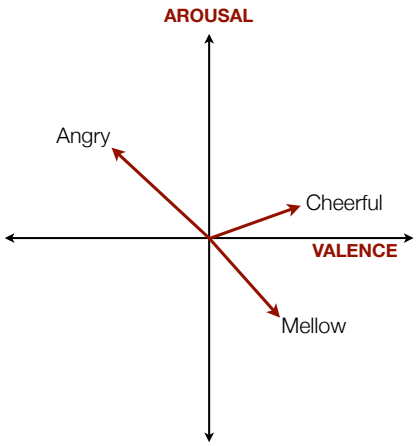


Figure 8: Example mood vectors

The mapping of textual emotion tags to vectors is, of course, a subjective process, as any translation of emotions between categorical and dimensional representations must be. However, the mapping used in this test was based on Thayer's descriptions of the two dimensional properties [19] and the indicative locations of moods on the arousal/valence plane adhered to by many MIR and MER works that derive from Thayer's thesis (for example see [11, 17, 23]). Therefore, while subjective, it can be said that the mapping of textual emotion tags to arousal/valence vectors is at least consistent with other MER work.

A second set of arousal/valence shapes for the tracks in the CAL500 set was collected by parsing the analysis data for each track from Echo Nest, by the same process used by the harmonious system. Comparing the two sets of circular shapes had the following results:

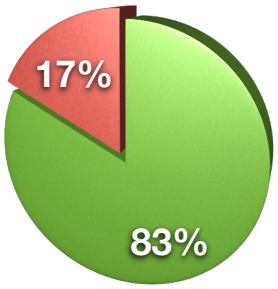


Figure 9: Percentage of circular mood representations overlapping between CAL500 and Echo Nest

- 82.67% of the circle pairs overlapped.
- The average distance between pairs that did not overlap was 10.5% the width of the plane.

These results sit quite favourably with results gained in various recent automated emotion recognition systems, which typically

[16] Music Information Retrieval

[17] Music Emotion Recognition

achieve 45-80% accuracy for dimensional emotion representation [11].

However, the Echo Nest analysis data suffers from the same difficulty found in other MER projects, that valence data is much courser grained than arousal data. Echo Nest data gives musical mode as a binary value with a floating point confidence rating, whereas data contributing to arousal (energy and loudness) are provided as fine grained floating point values. This issue manifests itself in the Harmonious system as a tendency for mood shapes to cluster towards the middle of the valence axis. However, the Echo Nest analyser is under ongoing development, and as many researchers have adopted the arousal/valence model for MER work, it is expected that improvements will be made in this area.

The tag matching dictionary, the mood vector assignments and the code used to carry out the testing are provided in appendix 3.

6.REFINING MUSIC ANALYSIS AND MATCHING MUSIC WITH GAME MOODS

When the Harmonious system is in use with a game, the game analyser object monitors game conditions and sends cues for music to start, transition and stop to the system controller. The system uses gameplay mood information from the game analyser and the music analysis data (music segments and mood shapes) in the database to find an appropriate match between music and gameplay.

The initial musical analysis and mood shape representations for music segments are merely the starting point for the Harmonious system's capability to match music with gameplay. Whenever the system is in use by a game, each music segment's emotional representation can be refined by the user/player, which in turn improves the accuracy the process by which music is matched to gameplay.

Each music segment entry in the database includes two sets of mood shapes, one set for 'included' shapes and one for 'excluded' shapes. The initial shape created by the analysis workflow is added to the included shapes set. During gameplay a player can indicate to the system that the music segment currently playing is particularly appropriate match to the current gameplay. The system then creates a new circular mood shape and adds it to the segment's included shapes set. The location of centre of the new circle is located at the point on the arousal/valence plane that represents the current game mood (this is provided by the game analyser object) The radius of the circle is fixed at 0.2 units, a value found via experimentation. Having additional mood shapes in a segment's included shapes set gives the segment a higher chance of being chosen for the given gameplay mood, as discussed below.

Similarly a player can indicate that the currently playing music segment is a poor or inappropriate match to the current gameplay. In this circumstance, the system creates a new mood shape in the same manner as outlined above and adds this to the segment's excluded shapes set. This prevents the segment from being chosen for the given gameplay mood, as discussed below.

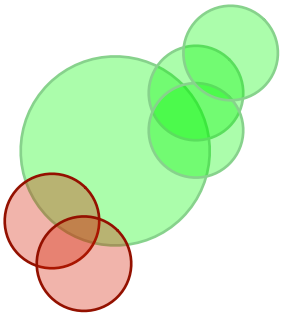


Figure 10: Example representation of a music segment's mood shapes, 'included' shapes in green and 'excluded' shapes in red

When the Harmonious system is required to select a new music segment for playback (due to receiving a cue from the game, or from the currently playing segment coming to an end without a cue to stop music playback), the current gameplay mood, represented as a set of co-ordinates in the arousal/valence plane, is fetched from the game analyser object. This in turn is passed to the music manager object to find an appropriate match in the music segments database. To avoid excessive repetition, the music manager maintains a record of the previous fifteen audio files played, and excludes these from its matching process.

The music manager queries each music segment in the database for a match score against the supplied arousal valence coordinates. A music segment calculates its match score by starting from zero and incrementing the score by one for each shape in its included shapes set which encloses the given co-ordinate, then decrementing the score by one for each shape in its excluded shapes set which encloses the current co-ordinate. A positive match score therefore indicates a possible match between music and gameplay, with a higher score indicating a better match.

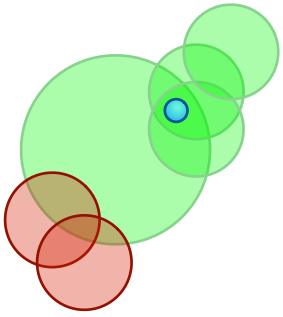


Figure 11: The match score between this music segment and the mood location indicated by the blue dot would be 3

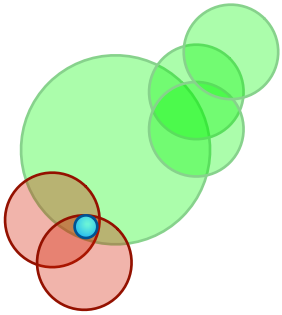


Figure 12: The match score between this music segment and the mood location indicated by the blue dot would be -1

The music manager collects each of the music segments with a positive match score, and assigns them a weighting equal to their score. The weightings are used to influence a pseudo random choice where a segment with a match score of two is twice as likely to be chosen as a segment with a match score of one.

Allowing for the possibility of lower scored matches to be chosen (albeit with a lower likelihood) may seem counterintuitive, and early in development the music manager used a selection system focused entirely on the top scored matches. Here the music manager would collect all the joint top matches for the game mood (i.e. if there existed ten segments with a match score of 1, five with a match score of 2 and four with a match score of 3, only those last four would be collected), and from this smaller collection it would choose a segment at random. However, this raised a serious issue when a user/player would begin to use the system for the first few times. In the early levels of the host game used for testing and development, the game mood could be consistently predicted to remain within a small area of the lower right quadrant of the arousal/valence plane. During a player's first attempt at the game, if the player indicated a 'like' of the music chosen at this point in gameplay, that piece of music's score would increase to 2 for that arousal/valence point. On the player's second attempt at the game, this piece of music would always be chosen, as it would be the only highest scored match. Players found that when having multiple attempts at the game in a short timeframe, hearing the exact same music at such points was too repetitive and would often start to mark choices they had previously 'liked' with a 'dislike' for the same gameplay mood just to overcome this repetitiveness. Therefore the weighted choice system was developed and has proven more robust.

In the circumstance where no music segment in the database has a positive match score for the given gameplay mood co-ordinate, the music manager makes a nearest match. This is done by requesting each segment in the database to return a value indicating distance between the given co-ordinate and the nearest edge of a composite shape created from the union of the areas all the circles in the segment's included shapes set with the subtraction of the areas of all the circles in the segment's excluded shapes set. Any segments with an excluded shape enclosing the current gameplay mood point are discarded. From the remaining segments, the one which returns the shortest distance is chosen.

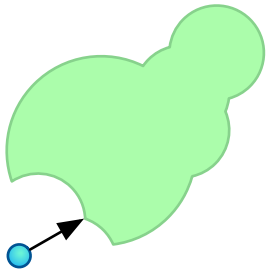


Figure 13: The minimum distance between a mood location and a composite shape

7.THE USER INTERFACE (UI)

The user interface displayed when Harmonious is in use with a game is shown below.

Figure 14: The Harmonious user interface

A text display shows the title and artist of the currently playing music segment and, when appropriate, the status of the music analyser.

As mentioned above, the player has the ability to refine the system and improve the matches between gameplay mood and music. Early in the development of the Harmonious system, this was carried out via the only two controls available to the player, a 'like' button and a 'skip' button. The 'like' button would indicate approval to the system; the 'skip' button would indicate disapproval and cause the system to select an alternate music match. However feedback from early demonstration and testing indicated a need to separate 'dislike' and 'skip' into individual controls, as players may wish to skip a piece of music for reasons other than disapproving of the match with the current gameplay.

This led to the development of a four control system: 'like', 'dislike', 'play/pause' and 'skip'. The former two send the approval and disapproval signals to the system respectively, with the 'dislike' control then triggering a transition to an alternate music segment. The latter two controls act like those on any music player, and have no effect on the system's database.

Feedback from players also indicated the need for a control that would indicate to the system that a particular piece of music should not be used at all by the system. Many of those players' digital music collections contained music acquired by spouses and/or children, and other music that for various reasons they would not normally listen to and did not wish to hear when playing a game.

To meet this issue a fifth control was added, the 'ban' control. This control causes the currently playing music never to be chosen for playback.

Though the Echo Nest music analyser scored very favourably when tested against the CAL500 dataset, 82.67% accuracy still left the potential for almost 1 in 5 tracks to have inaccurate mood shapes at the end of the analysis process. It therefore seemed prudent to include a means to directly manipulate mood shapes. To this end a 'fine tune' interface was added, which normally remains hidden, but can be revealed by clicking a button. This interface consists of two sliders, one labelled Sad/Happy, the other Laid Back/Energetic. The values of these sliders would be set to the valence and arousal values of the segment's original mood shape, and adjusting them would reposition that mood shape on the arousal/valence plane accordingly.

Figure 6: The Harmonious 'fine-tuning' interface

However, during player testing this 'fine tune' control was never used. The reasons for this were twofold. Firstly, a clear understanding of the function of this control required a deeper understanding of the internal operation of the Harmonious system than is necessary for its normal use. Secondly, manipulating this control would typically require pausing the game. This clearly breaks a player's immersion in the game, and as such diminishes the experience and violates the objectives of the project. Additionally, players were content with the results of using the 'like', 'skip' and 'ban' controls to fine tune the music matching process. It therefore seems that the 'fine tune' control may not be required as part of the standard user interface, and further testing and evaluation may indicate that it can safely be removed (the testing and evaluation carried out to-date is discussed in section 7).

Each of the player controls can be activate with the mouse, but this would often prove inappropriate during gameplay, so each is also mapped to a key on the keyboard, and could also be mapped to specific buttons on a game pad controller.

The default keys are:

1 = Like
2 = Dislike
3 = Ban
4 = Play/Pause
5 = Skip

8. THE PROGRAMMER INTERFACE (API)

The Harmonious music system is written in Objective-C and makes use of the FMOD Ex[18] C/C++ library for music playback, the Core Data Framework[19] for the music segments database, and the Cocoa Framework[20] for general operations (such as querying the Echo Nest server) and the user interface. Each of these libraries and frameworks are high-level and general function, and could be substituted for others with similar functionality should the Harmonious system be ported to a platform for which these were unavailable. Similarly, the API provided by the Harmonious system to integrate it into a game is in the Objective-C language, but could easily be adapted to C or C++ if necessary.

Implementing the Harmonious system within a game involves incorporating the system's code into the game, and causing an instance of the HarmoniousSystemController class to be created when the game is launched. The programmer must also create a GameAnalyser object specific to the game.

Though the GameAnalyser must be created for each game, the Harmonious system provides a template API so that it may be easily incorporated into the system. The template API requires the programmer to implement the following methods[21] in the GameAnalyser class:

```
-(id)initWithGameController:(id)theGameController;
```

This method is the designated initialization method for the GameAnalyser class, in which it receives a reference to an object referred to as theGameController. The type and nature of the object is unspecified, and would be an object which the game specific implementation code can communicate with to receive the current game state and any required game variables.

```
-(void)activate;
-(void)deactivate;
-(BOOL)isActive;
```

The first two of the above three methods indicate that to the GameAnalyser that it should start and stop sending cues to the Harmonious system controller. They would typically be invoked when the user/player switches between the game's original soundtrack and the Harmonious system. The third method is simply a means by with other objects can query the active state of

[18] http://www.fmod.org/index.php/products/fmodex

[19] https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/CoreData/cdProgrammingGuide.html

[20] https://developer.apple.com/technologies/mac/cocoa.html

[21] In object-oriented programming, methods are programmatic routines, similar to functions, that are carried out by objects of a specific class

the GameAnalyser. This method should return a boolean value of *true* when the GameAnalyser is active and *false* when it is inactive.

```
-(NSPoint)currentGameplayMoodInArousalValenceCoordinates
```

This method is the means by which other components of the Harmonious system can discover the current gameplay mood in arousal/valence co-ordinates, as determined by the game analyser. The return type, *NSPoint*, is a structure containing two floating point values, labelled 'x' and 'y'. The x value should represent valence, the y value arousal.

```
@property id delegate;
```

Finally, the GameAnalyser must implement this *delegate* property[22] in order to receive a reference to the HarmoniousSystemController object. This reference is the means by which the GameAnalyser can send cues to the system. Two methods are defined in the HarmoniousSystemController class to facilitate these cues. These are:

```
-(void)playNewTrackForArousalValencePoint:(NSPoint)avPt;
-(void)fadeOutAndStopCurrentTrack;
```

The former is used to start music playback and transition to a different piece of music, the latter is used to stop music playback.

As the programmer has free reign over the implementation of the GameAnalyser class, the arousal/valence point it sends to the Harmonious system can be based on any factors appropriate to the game. These may be simple analyses of the number of enemies and the player character's health, through to more sophisticated processes, based on the game's narrative which, for example, could cause the arousal/valence point to indicate tension over a forthcoming event.

The remainder of the programmer's interaction with the Harmonious system is via the API presented by the HarmoniousSystemController class. This class implements three methods:

```
-(id)initWithGameController:(id)gameController;
-(void)startMusicPlayback;
-(void)stopMusicPlaybackWithFade:(BOOL)fadeOut;
```

The first method is the designated initializer for the class. The gameController parameter should be a reference to the object that is in turn passed to the GameAnalyser object upon its initialization. The second method should be called when the user/player has elected to listen to music via the Harmonious system. It in turn causes the GameAnalyser object to activate. The third method should be called when the user/player elects to stop listening to music with the system. This method stops playback and deactivates the GameAnalyser. The boolean *fadeOut* parameter is used to indicate whether the system should fade out any currently playing music, or stop it immediately.

The HarmoniousSystemController class also exposes the methods used by the user interface. These are:

```
-(void)showHarmoniousUIWindow;
-(IBAction)likeMusic:(id)sender;
-(IBAction)dislikeMusic:(id)sender;
-(IBAction)banMusic:(id)sender;
-(IBAction)pauseMusic:(id)sender;
-(IBAction)skipTrack:(id) sender;
```

Should the programmer wish to use an alternate user interface for the system, its controls can call these methods to gain the same functionality as the original interface.

The full interface declarations (header files) for the HarmoniousSystemController class and the GameAnalyser template can be found in Appendix 4.

9.GAME ANALYSER IMPLEMENTATION IN THE SPACE DEFENDER GAME

A small game was created specifically for testing, evaluation and development of the Harmonious system. This game, called '*Space Defender*', is a simple, two dimensional arcade-style game, where the player controls a space craft and must fend off waves of attacking alien craft. The attackers increase in numbers with each new wave, and waves are broken up with 'boss' levels, where a single, large and very strong enemy attacks.

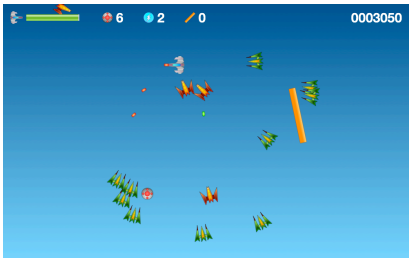


Figure 7: The *Space Defender* game

To create a reference against which the Harmonious system could be compared, a dynamic music system was created for the game using FMOD Designer and implemented using the FMOD Event library/API[23]. The music begins as an ambient synth pad and increases in complexity and intensity by adding further instrument tracks and layers in response to increasing intensity in the game. For the implementation of this music, game intensity is considered a one-dimensional property, defined by the formula:

$$\left(\frac{en}{100}\right)+\left(\frac{100\text{-}h}{100}\right)$$

[22] In the Objective-C language, a property is simply a shortcut for defining a combination of a variable, in this case called 'delegate', and a pair of methods to get and set the value of that variable.

[23] http://www.fmod.org/index.php/products

Where *en* is the maximum number of enemies in the current attack wave, and *h* is the player's level of health (a value between 0 and 100). This intensity value is then halved in the short intervals between a player defeating an attack wave and the beginning of the next wave. The intensity value is therefore a value between 0.0 and 1.0, and this is mapped to a parameter of the same scale in an FMOD event. This event contains a single sound instance of a 12-channel wave file containing each of the music layers. The FMOD event mixes these layers according to the parameter value.

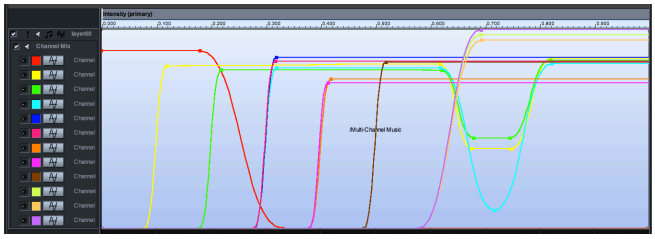


Figure 15: The Music event in the FMOD Designer application

The Harmonious system code was also added to the game, and a GameAnalyser object was created to link the game to the system. The GameAnalyser creates and arousal valence co-ordinate representation of the gameplay mood by the formula:

$$arousal = \left(\frac{en}{35}\right)\text{-}1 \quad valence = \left(\frac{h}{50}\right)\text{-}1$$

which constrain the arousal/valence values between -1 and 1.

A preference option is provided in the game to allow the player to switch between listening to the game's original music and music from the Harmonious system.

10.PLAYER FEEDBACK

The *Space Defender* game was made available to a small group of five players to gather feedback. Players were asked to use the original music whilst playing their first few attempts at the game and becoming familiar with the controls. Once they became comfortable with the game, they would switch to playing with the Harmonious system. The nature of the Harmonious system and its controls were explained to each player beforehand.

The method used to gather this feedback was modeled on that used by Wharton and Collins in a study examining the experience of players given free choice over the music heard during gameplay [22]. In this study, players played the game in the presence of the researcher and used the Think-Aloud method [13] during play. This method is common in user interface testing and evaluation [14] and encourages testers to say out loud what they are thinking as they use and test a device or interface. In the Wharton and Collins study, players were also asked to pause at checkpoints in the game and reflect on their choices of music with the researcher.

Such an approach allows the researcher to gain first hand experience with the participants and allows for any unusual or unexpected occurrences to be observed directly [4]. This approach seemed particularly suitable for early testing of the Harmonious system as it allowed for the gathering of feedback on music choices by the system and their effect on the player as they were happening, and before those effects could be diminished or forgotten by subsequent music choices and game events.

Players were asked to play the *Space Defender* game with the Harmonious system (which had previously analysed the player's music library) and asked to think-aloud on their interactions with the system and the effects of the music on their gameplay. Specifically, players were asked to reveal their thoughts when they made use of any of the harmonious systems controls ('like', 'ban', etc) to make clear their desires and intentions; and to comment of the results of using those controls to reveal whether the system responded as expected. Additionally, players were asked to comment on the matches between music and gameplay and were encouraged to speak any other thoughts they had about the game and the music.

Two players seemed comfortable with the description of the controls before playing, but early in the gameplay they expressed confusion over the 'dislike' and 'ban' controls, saying they were unsure of the difference between the two actions, and therefore unsure of which to use in a particular circumstance. After a further, short explanation of these controls, each of these players' think-aloud comments revealed their understanding. Of the remaining players, one had been involved in earlier testing during the development of the user interface, and therefore found it clear. The remaining two also found the distinction between these two controls clear, as revealed by comments such as 'this song is too slow' before using the 'dislike' control and 'this is one of [my spouse]'s songs' before using the 'ban' control.

One player seemed to make use of the 'like' control to indicate enjoyment of a particular piece of music, rather than approval of the match between music and gameplay, as revealed by think-aloud comments such as 'this is a great song', 'I like this one' and 'this is my kind of music'. When asked about the function of the 'like' control, this player revealed that he was indicating approval of the match between gameplay and music genre, rather than music mood (unfortunately the long term results of using the system in such a manner cannot be yet determined from the amount of testing carried out to-date). The other players showed use of the 'like' control for mood match approval, as revealed by think-aloud comments such as 'this is a better tempo' and 'this is a good fit'.

In later attempts at the game each player's think-aloud comments were less in regard to the use of the controls and more on the music and gameplay in general, seeming to indicate increased confidence and comfort with the controls.

In summary, some players showed some initial misunderstanding of the system's five basic controls, but through coaching and experience they achieved suitable understanding and familiarity. It would seem therefore that this control system is reasonably understandable, but that some initial player induction would be beneficial. Such an induction could perhaps take place in the form of a short in-game tutorial when the system is used for the first time, in a similar manner to gameplay tutorials offered at the beginning of games.

Between attempts at the game (when the player had 'died') players were asked to reflect on how well they felt the Harmonious system was refining itself to their preferences. All players reported that they found they system's music choices had improved on each attempt. Two players had made heavy use of the 'like, 'dislike', and 'ban' controls during their first two attempts at the game, and by the third attempt they reported that they felt the system was making accurate music choices. In later attempts these players made only occasional use of these controls. These players seemed to treat their initial attempts as simply system training and the latter attempts as actual gameplay. The remainder of the players focused

more consistently on gameplay throughout their attempts, and correspondingly made slightly lighter use of the system controls. These players reported a similar satisfaction with match accuracy by their fourth and fifth attempts. An interesting phenomenon was that when Harmonious played music of a player's favourite genre, most players claimed that this type of music was a perfect match to the Space Defender game's style of gameplay.

Based on this mid-game feedback, it would seem that the Harmonious system is meeting the project's objective of automating music choices with mood matching. Players found that, after some refinement of the system, the music choices were accurate and effective. However, further research and testing could prove that the requirement for refinement is discouraging to potential users. As discussed above, the initial matching has an accuracy of around 83%, giving the potential for almost 1 in 5 matches to be inappropriate when a player is using the system for the first time. Should this be the case, the tutorial system suggested above may help mitigate this discouragement.

After playing for around 45 minutes players were asked for feedback relating to the project objectives: how informative the music selections were and how their enjoyment of game and focus on gameplay compared against playing with the original music. Players were asked what they understood the role of the game's original music to be. Every player responded that the link to game intensity was clear, particularly in the later, more intense stages of gameplay where health is low and/or enemies are overwhelming. Players were then asked if they felt the music played by the Harmonious system fulfilled the same role. Again, every player reported that they felt that the music played by Harmonious followed a similar pattern of intensity throughout the game. Players were asked if they felt more or less focused on and engaged with the game when playing with the Harmonious system. Four players reported that they found the higher intensity portions of gameplay were particularly effective with the Harmonious system, where the intensity of the music heightened the gameplay experience and increased their focus on the game. Comments were sparser and more neutral for the lower intensity waves at the beginning of the game. Two players noted that they found themselves firing their weapons in time to the music during medium-high intensity gameplay, which would typically be accompanied by up-beat music. One player however, reported that he was less concerned with matches between music and gameplay, and more with hearing his current favourite tracks. During periods of low activity between attack waves, he would concentrate on skipping through several tracks in the Harmonious UI to find one of his favourites (this being the same player that had used the 'like' control to match gameplay with genre, described above).

This post-game player feedback would seem to indicate that the Harmonious system is capable of meeting the project objective of retaining information and feedback provided by original game music. However, the test game is limited in terms of the emotions it provokes in players, and therefore the gameplay moods which the system is matching music against are also limited. It remains to be seen how effective the system would be with a more emotionally complex game.

During the evaluation, players gave a few further interesting comments. One player noted that he had tried listening to alternate music whilst playing games by playing music on a hi-fi system and a portable music player. He had liked the idea of hearing his own music, but these methods typically made the game's sound effects difficult to hear, and he felt hearing these effects was crucial in the games he played (primarily motor racing and FPS games). He felt the Harmonious system was a better solution as the music playback was integrated within to the game, and could be controlled with the game's own relative volume controls. Two players raised a concern that having to refine the system (via 'like' and 'dislike') multiple times for multiple games would be tedious and likely to discourage their use of the system. Having refinements shared amongst all games using the system would be more appealing. One player also had the concern that without a large and varied music collection, the system would be ineffective.

From this player testing and evaluation, it can be concluded that the Harmonious system has the potential to meet the project's objectives. However, the feedback gathered was from only a small number of players and with a very simple game. Therefore it can only be taken as an early indication. Additionally, the data gathering approach used in this evaluation is known to have drawbacks, specifically that subjects may find the presence of the researcher to be intrusive and that the researcher's presence may bias the subject's feedback [4].

Conclusive proof of the system meeting the project's objectives will require testing with larger numbers of players and games of differing styles, genres and complexity. In such testing, the potential limitations of a researcher-in-attendance approach could be mitigated by the establishment of an Observational Protocol [4] which would allow for the observations and interviewing to be carried out effectively by a third party. Such a protocol could include the gathering of demographic and biographic data on participants to provide a richer data set from which a deeper analysis could be carried out [4]. The feedback gathered to date indicates that such further testing would be worthwhile.

11. FURTHER DEVELOPMENT

11.1 Social Playlist Sharing

At the outset of this project it was envisioned that the system would include a social component, whereby user/players could share playlists amongst their friends[24]. To facilitate this the Harmonious system would keep a record of music that a player 'likes' on a game-by-game basis. These 'liked' tracks would be collected into a playlist entity in the Harmonious database. When a game had a sufficient number of tracks in it's playlist (the number would have to be defined via testing, but is estimated to be at least 50) the playlist would be uploaded to an online database server. When another player begins playing a game with the Harmonious system, that player's system would check the online database for any updated playlists associated with that particular game from any of the player's friends. The player could then be offered the option to play the game with the music that their friends preferred. Should the player not have the playlist's complete set of songs in their music library, they could be presented with an option to purchase the remaining music from an online service appropriate to the game platform. By use of affiliate programmes, this could present a revenue opportunity for the Harmonious project.

[24] The term 'friend' is used here in the social networking sense, indicating a fellow user of an online service with whom a given user has agreed to share information and communications through the service. Popular services in the video game market include Microsoft's XBox Live and Games for Windows Live, and Sony's Playstation Network

Implementing a social playlist sharing component to Harmonious would not be a technically complex undertaking, it was just not possible to complete within the timeframe of the project. As the Harmonious system already makes use of a local database and is processing player 'likes' during gameplay, it would be trivial to have the system build playlist by noting the track (with metadata) associated with the music being 'liked' and add this to a collection in the database labelled with the game title. As each track is added to this collection, the system can check for reaching the number of entries that would signify a 'full' playlist. At this point the playlist would be uploaded to an online database, including track metadata, segment locations and arousal-valence mood shapes.

Creating an online database could be achieved quite simply by using existing technologies, such as Ruby-on-Rails[25] and mySQL[26], which are commonly used for web applications [8]. These technologies would allow for an online database access application with user/password authentication to be created in a very short timeframe [8]. New playlist entries could be uploaded to such a server by encoding the contents of the playlist into the parameters of an HTML POST request[27]. A similar request could be used as the means by which checking for and downloading playlists are achieved.

The Harmonious system would check with the server when gameplay was about to begin, download any new playlists from friends and offer these to the player. Should the player accept, the music matching system would temporarily favour the music segments in the playlist and use the arousal/valence mood shapes from that playlist, rather than those that exist in the local database. Should the player make any 'likes' or 'dislikes' during gameplay with the playlist, these would be merged into the local database for future play.

Sharing features could be activated or deactivated by a local preference setting in the Harmonious system.

11.2 Platform Level Integration

Whilst integrating the Harmonious system into a game is a reasonably simple process, the system could prove more effective if implemented at a platform level as a platform-wide service. As such, music analysis data and user refinement of mood shapes could be easily shared amongst any games played on the platform (player feedback indicated that having to 're-train' the system multiple times for multiple games would be unappealing).

Platform level integration was also allow for simpler access to the platform's online music sales service (if available) and would allow Harmonious to play music files that are subject to any digital rights management restrictions enforced on the platform, a feature that is not currently available to Harmonious as a stand-alone technology.

12. CONCLUSIONS

This project aimed to create a means by which video game players could listen to their own music libraries during gameplay instead of the game's original music, in an integrated, intelligent manner which would help preserve the information and feedback contained in the original music. To this end, the project's stated objectives were:

- Allow players to listen to music in their own libraries whilst playing games
- Automate the choice of music from the player's library by matching gameplay mood with the emotional content of the music, thereby:
- Retaining some of the information and feedback provided by the original game music

The Harmonious system was developed to meet these objectives, and operates around a central principle of emotional analysis of music and gameplay resulting in two-dimensional geometric representations of emotions on a plane where arousal and valence properties form the axes.

The Echo Nest platform is used to perform the bulk of the music analysis operations, and this proved effective. The efficiency of this approach was very high when compared to other efforts in the field of music emotion recognition. However, as found in other studies, valence remains a more difficult property of music to resolve at fine-grained level compared to the evaluation of arousal.

The means by which players could refine the music analysis data during gameplay were developed and fine-tuned with player feedback and have proven effective in quickly and unobtrusively improving the system's matching to the player's tastes. Initial evaluations have shown that the advanced refinement features may be unnecessary, but they remain in place until such time as more comprehensive player testing can take place.

Initial player testing has shown that the Harmonious system operates at a suitable level of efficiency and that players feel the music chosen by Harmonious during gameplay provides an effective means of communicating the information provided by the game's original music. Players found the experience of playing with the Harmonious system pleasurable and found that it contributed to their immersion in the gameplay. However, this initial testing was limited to a small number of players playing a very simple game. Further and more rigorous testing with more involved games and greater numbers of players is required to draw definitive conclusions.

13. REFERENCES

[1] Apple, Inc, 2010. Media Player Framework Reference. [Website]. *iOS Developer Library*, Available from: https://developer.apple.com/library/ios/#documentation/MediaPlayer/Reference/MediaPlayer_Framework/_index.html#//apple_ref/doc/uid/TP40006952 [accessed 4 June, 2010].

[2] Cano, P., Batlle, E., Kalker, T. and Haitsma, J., 2005. A Review of Audio Fingerprinting. *The Journal of VLSI Signal Processing*, **41** (3), p. 271-284.

[3] Collins, K., 2008. *Game Sound: An Introduction to the History, Theory, and Practice of Video Game Music and Sound Design*. Cambridge, Massachusetts: The MIT Press.

[25] http://rubyonrails.org

[26] http://www.mysql.com

[27] A description of the HTML POST request can be found at http://www.w3.org/MarkUp/html-spec/html-spec_8.html#SEC8.2.1

[4] Creswell, J. W., 2003. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. 2nd ed. London: Sage Publications.

[5] The Echo Nest, 2008. *The Intelligent Music Platform*. [Webpage]. Available from: http://the.Echo Nest.com/ [accessed 14 June, 2011]

[6] The Echo Nest, 2008. *How The Echo Nest Works*. [Webpage]. Available from: http://the.Echo Nest.com/platform/how-it-works/ [accessed 14 June, 2011]

[7] Game Audio Pro Discussion Forum Members, 2011. Message Thread: Mod'able or hackable game for music project [Discussion Forum]. *Game Audio Pro, Yahoo Groups*, Available from: http://tech.groups.yahoo.com/group/gameaudiopro/message/15264 [accessed 7 September, 2011].

[8] Holzner, S., 2007. *Beginning Ruby on Rails*. Indianapolis, IN: Wiley Publishing.

[9] Jehan, T., 2010. *The Echo Nest Analyze Documentation*. [White Paper]. Available from http://developer.echonest.com/docs/v4/_static/AnalyzeDocumentation_2.2.pdf [accessed 14 June, 2011]

[10] Jørgensen, K., 2008. Left in the dark: playing computer games with the sound turned off. In K. Collins, ed. *From Pac-Man to Pop Music: Interactive Audio in Games and New Media*. Aldershot: Ashgate.

[11] Kim, Y.E., Schmidt, E.M., Migneco, R., Morton, B.G., Richardson, P., Scott, J., Speck, J.A. and Turnbull, D., 2010. Music emotion recognition: A state of the art review. In: *Proceedings of the 11th International Society for Music Information Retrieval Conference, 9-13 August, 2010, Utrecht, Netherlands*. ISMIR, p. 255-266.

[12] Laurier, C., Sordo, M. Serrà, J. and Herrera, P., 2009. Music Mood Representations from Social Tags. In: *Proceedings of the 10th International Society for Music Information Retrieval Conference, 26-30 October, 2009, Kobe, Japan*. ISMIR, p. 381-386.

[13] Lewis, C., 1982. *Using the "Thinking-Aloud" Method in Cognitive Interface Design*. IBM Research Report RC 9265, IBM Thomas J. Watson Research Center.

[14] Lewis, C. and Rieman, J., 1993. *Task-Centered User Interface Design: A Practical Introduction*. Boulder, Colorado: University of Colorado at Boulder.

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

[16] Morris, S., 2003. WADs, Bots and Mods: Multiplayer FPS Games as Co-creative Media. In: *Level Up Conference Proceedings, November 2003, Utrecht*. University of Utrecht.

[17] Panda, R. & Pavia, R. P., 2011. Using Support Vector Machines for Automatic Mood Tracking in Audio Music. *In: 130th Convention of the Audio Engineering Society, 13–16 May 2011, London, UK*. AES.

[18] Schmidt, B., 2006. An Overview of Xbox 360 Audio. In: *Game Developer's Conference, 2006, San Francisco*. UBM TechWeb.

[19] Thayer, R. E., 1989. *The Biopsychology of Mood and Arousal*. New York: Oxford University Press.

[20] Totilo, S. , 2011. The Year I Gained The Courage To Ignore Video Game Music. *Kotaku*, 11 January. Available from: http://kotaku.com/#!5730637/the-year-i-gained-the-courage-to-ignore-video-game-music [accessed 19 April 2011].

[21] Turnbull, D., Barrington, L.,Torres, D. and Lanckriet, G., 2008. Semantic Annotation and Retrieval of Music and Sound Effects. *IEEE Transactions on Audio, Speech and Language Processing*, **16** (2), p. 467-476.

[22] Wharton, A. & Collins, K., 2011. Subjective Measures of the Influence of Music Customization on the Video Game Play Experience: A Pilot Study. *The International Journal of Computer Game Research*, **11** (2).

[23] Yang, Y.H., Lin, Y.C., Su, Y.F. & Chen, H.H., 2008. A regression approach to music emotion recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, **16** (2), p. 448-457.

[24] Yujian, L. and Bo, L., 2007. A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, **29** (6), p. 1091-1095.

Appendix 1: Game Music Survey Results

ID	7	6	1	2	3 In which genres of games would you be more likely to turn off music or listen to your own music?													
	Age	How often to you play video games?	Do you turn off video game music when playing and/or replace it with your own music?	Are you more likely to turn off game music or listen to your own music when playing single player or multiplayer?	*First/ Third person shooters*	Role play games	Platform games	Fighting games	Stealth games	Sim games	Strategy games	Racing games	Sport games	Motion games	Casual/ Mobile games	Puzzle games	Other
fb4	Teens or younger	Hardly ever or not at all	Never	Neither												1	
fb5	30's	Several times a month	Sometimes	Single Player											1	1	
fb9	40's	Several times a month	Sometimes	Single Player								1					
fb10	20's	Several times a month	Sometimes	Neither						1	1	1			1		
fb11	30's	Several times a month	Sometimes	Single Player						1					1	1	
fb12	Teens or younger	Several times a month	Sometimes	Single Player	*1*	1	1	1			1				1		
fb14	30's	Several times a month	Sometimes	Multiplayer	*1*					1					1	1	
fb15	40's	Several times a month	Regularly	Single Player			1	1	1	1	1		1	1	1	1	
fb16	30's	Several times a month	Never	Neither								1		1	1	1	
6663956	20's	Several times a month	Never	Neither											1		
6657510	20's	Several times a month	Sometimes	Multiplayer								1	1				
6654826	20's	Several times a month	Never	Neither				1				1					
6686062	20's	Several times a month	Never	Neither						1							
6657275	20's	Several times a month	Sometimes	Both	*1*												
6658049	20's	Several times a month	Never	Multiplayer	*1*												
6685660	Teens or younger	Several times a month	Sometimes	Single Player								1	1		1	1	
fb1	40's	Several times a week or more	Regularly	Multiplayer	*1*		1								1		
fb2	20's	Several times a week or more	Sometimes	Multiplayer								1	1		1	1	
fb3	40's	Several times a week or more	Regularly	Single Player									1				
fb7	Teens or younger	Several times a week or more	Regularly	Multiplayer	*1*												
fb8	Teens or younger	Several times a week or more	Never	Neither				1		1		1				1	
fb13	20's	Several times a week or more	Sometimes	Multiplayer	*1*			1			1	1	1				
fb17	20's	Several times a week or more	Sometimes	Single Player													1
6658383	30's	Several times a week or more	Sometimes	Multiplayer								1			1		1
6656765	30's	Several times a week or more	Sometimes	Both													1
6654967	30's	Several times a week or more	Sometimes	Single Player													1
6657821	30's	Several times a week or more	Never	Neither													1
6685675	30's	Several times a week or more	Regularly	Both		1	1	1	1	1	1	1	1	1	1	1	1
6655191	30's	Several times a week or more	Sometimes	Single Player			1	1		1	1	1	1	1	1	1	
6656830	30's	Several times a week or more	Sometimes	Multiplayer								1	1		1		
6655307	30's	Several times a week or more	Regularly	Single Player								1	1				
6657202	20's	Several times a week or more	Regularly	Both	*1*	1	1	1		1	1	1	1		1	1	1
6667841	20's	Several times a week or more	Sometimes	Both						1	1	1			1	1	1
6656809	20's	Several times a week or more	Sometimes	Single Player											1		1
6663921	20's	Several times a week or more	Never	Neither											1	1	1
6663931	20's	Several times a week or more	Sometimes	Multiplayer									1				1
6657075	20's	Several times a week or more	Sometimes	Both	*1*					1	1	1					1
6654757	20's	Several times a week or more	Never	Multiplayer													1
6661363	20's	Several times a week or more	Never	Multiplayer													1
6684423	20's	Several times a week or more	Never	Multiplayer													1
6654685	20's	Several times a week or more	Sometimes	Multiplayer													1
6655490	20's	Several times a week or more	Sometimes	Multiplayer													1
6658446	20's	Several times a week or more	Sometimes	Multiplayer													1
6657690	20's	Several times a week or more	Never	Neither													1
6686089	20's	Several times a week or more	Never	Neither													1
6664107	20's	Several times a week or more	Sometimes	Neither													1
6684444	20's	Several times a week or more	Sometimes	Neither	*1*												1
6654969	20's	Several times a week or more	Regularly	Multiplayer	*1*		1	1		1	1	1	1	1	1	1	1
6664069	20's	Several times a week or more	Never	Multiplayer				1		1	1	1	1	1	1		1
6654760	20's	Several times a week or more	Sometimes	Both	*1*			1		1		1		1	1	1	1
6654635	20's	Several times a week or more	Sometimes	Multiplayer						1	1			1	1	1	1
6664374	20's	Several times a week or more	Regularly	Both	*1*		1	1		1	1	1	1		1	1	1

	7	6	1	2	3 In which genres of games would you be more likely to turn off music or listen to your own music?												
ID	Age	How often to you play video games?	Do you turn off video game music when playing and/or replace it with your own music?	Are you more likely to turn off game music or listen to your own music when playing single player or multiplayer?	*First/ Third person shooters*	Role play games	Platform games	Fighting games	Stealth games	Sim games	Strategy games	Racing games	Sport games	Motion games	Casual/ Mobile games	Puzzle games	Other
6654301	20's	Several times a week or more	Regularly	Multiplayer	*1*		1	1		1	1	1	1	1	1		
6684158	20's	Several times a week or more	Never	Neither						1		1	1	1	1	1	
6663858	20's	Several times a week or more	Sometimes	Multiplayer	*1*	1	1		1			1	1	1	1	1	
6686692	20's	Several times a week or more	Sometimes	Multiplayer				1				1	1	1	1	1	
6684403	20's	Several times a week or more	Sometimes	Multiplayer						1			1	1	1		
6658385	20's	Several times a week or more	Never	Multiplayer		1	1	1		1	1	1	1		1	1	
6655160	20's	Several times a week or more	Sometimes	Multiplayer	*1*		1	1		1	1	1	1		1		
6654396	20's	Several times a week or more	Regularly	Multiplayer	*1*		1	1		1	1	1	1		1	1	
6668530	20's	Several times a week or more	Never	Multiplayer	*1*			1		1	1	1	1		1		
6656354	20's	Several times a week or more	Sometimes	Multiplayer				1		1	1	1	1		1		
6664194	20's	Several times a week or more	Sometimes	Multiplayer	*1*			1		1	1	1	1		1	1	
6684234	20's	Several times a week or more	Sometimes	Multiplayer				1		1	1	1	1		1	1	
6663990	20's	Several times a week or more	Regularly	Multiplayer	*1*			1		1	1	1	1		1		
6663922	20's	Several times a week or more	Never	Multiplayer						1	1	1	1		1	1	
6657938	20's	Several times a week or more	Sometimes	Multiplayer	*1*					1	1	1	1		1		
6661409	20's	Several times a week or more	Sometimes	Multiplayer	*1*					1	1	1	1		1		
6686584	20's	Several times a week or more	Sometimes	Multiplayer	*1*	1		1			1	1	1		1	1	
6664004	20's	Several times a week or more	Sometimes	Multiplayer	*1*			1			1	1	1		1		
6664041	20's	Several times a week or more	Sometimes	Single Player			1	1	1	1		1	1		1	1	
6686652	20's	Several times a week or more	Regularly	Both			1	1		1		1	1		1		
6685727	20's	Several times a week or more	Sometimes	Multiplayer	*1*			1		1		1	1		1		
6659813	20's	Several times a week or more	Sometimes	Single Player						1		1	1		1		
6655010	20's	Several times a week or more	Sometimes	Multiplayer				1				1	1		1	1	
6655559	20's	Several times a week or more	Sometimes	Multiplayer	*1*			1				1	1		1	1	
6656292	20's	Several times a week or more	Sometimes	Multiplayer				1		1			1		1		
6667041	20's	Several times a week or more	Sometimes	Multiplayer	*1*					1			1		1		
6663853	20's	Several times a week or more	Never	Neither						1			1		1	1	
6663812	20's	Several times a week or more	Never	Neither						1	1	1			1	1	
6654494	20's	Several times a week or more	Sometimes	Both	*1*		1				1	1			1	1	
6686600	20's	Several times a week or more	Sometimes	Multiplayer	*1*						1	1			1		
6684274	20's	Several times a week or more	Regularly	Multiplayer	*1*			1		1		1			1	1	
6656226	20's	Several times a week or more	Sometimes	Both			1	1				1			1		
6655597	20's	Several times a week or more	Sometimes	Multiplayer				1				1			1		
6685781	20's	Several times a week or more	Sometimes	Multiplayer	*1*			1				1			1		
6657389	20's	Several times a week or more	Regularly	Single Player	*1*			1				1			1	1	
6655031	20's	Several times a week or more	Regularly	Single Player		1	1			1					1	1	
6658500	20's	Several times a week or more	Sometimes	Multiplayer		1		1							1	1	
6657933	20's	Several times a week or more	Never	Multiplayer	*1*	1									1		
6667306	20's	Several times a week or more	Sometimes	Both											1	1	
6661459	20's	Several times a week or more	Sometimes	Multiplayer	*1*										1		
6657776	20's	Several times a week or more	Sometimes	Single Player											1	1	
6686562	20's	Several times a week or more	Sometimes	Single Player											1	1	
6655017	20's	Several times a week or more	Never	Neither											1	1	
6656763	20's	Several times a week or more	Never	Neither											1		
6663974	20's	Several times a week or more	Never	Neither											1		
6684037	20's	Several times a week or more	Never	Neither											1		
6685630	20's	Several times a week or more	Never	Neither											1		
6657349	20's	Several times a week or more	Sometimes	Both	*1*		1	1		1	1	1	1				
6656815	20's	Several times a week or more	Never	Multiplayer						1	1	1	1				
6663989	20's	Several times a week or more	Never	Multiplayer				1		1		1	1				
6684192	20's	Several times a week or more	Sometimes	Multiplayer	*1*			1		1		1	1				
6660506	20's	Several times a week or more	Sometimes	Single Player						1		1	1				

	7	6	1	2	3 In which genres of games would you be more likely to turn off music or listen to your own music?												
ID	Age	How often to you play video games?	Do you turn off video game music when playing and/or replace it with your own music?	Are you more likely to turn off game music or listen to your own music when playing single player or multiplayer?	*First/ Third person shooters*	Role play games	Platform games	Fighting games	Stealth games	Sim games	Strategy games	Racing games	Sport games	Motion games	Casual/ Mobile games	Puzzle games	Other
6663839	20's	Several times a week or more	Sometimes	Multiplayer	*1*			1				1	1				
6654418	20's	Several times a week or more	Sometimes	Multiplayer								1	1				
6655310	20's	Several times a week or more	Sometimes	Multiplayer								1	1				
6686596	20's	Several times a week or more	Sometimes	Multiplayer	*1*							1	1				
6663953	20's	Several times a week or more	Sometimes	Single Player								1	1				
6656679	20's	Several times a week or more	Never	Multiplayer							1		1				
6658315	20's	Several times a week or more	Never	Multiplayer				1					1				
6658120	20's	Several times a week or more	Sometimes	Both				1			1	1				1	
6654409	20's	Several times a week or more	Sometimes	Multiplayer	*1*						1	1					
6656421	20's	Several times a week or more	Sometimes	Multiplayer	*1*		1	1		1		1					
6684100	20's	Several times a week or more	Sometimes	Multiplayer				1		1		1					
6654954	20's	Several times a week or more	Sometimes	Multiplayer						1		1					
6655041	20's	Several times a week or more	Sometimes	Multiplayer	*1*					1		1					
6657899	20's	Several times a week or more	Sometimes	Both		1		1				1				1	
6686056	20's	Several times a week or more	Sometimes	Multiplayer	*1*			1				1					
6657546	20's	Several times a week or more	Sometimes	Multiplayer	*1*							1					
6654394	20's	Several times a week or more	Never	Neither								1					
6654935	20's	Several times a week or more	Sometimes	Single Player		1				1	1						
6663886	20's	Several times a week or more	Sometimes	Multiplayer							1						
6658450	20's	Several times a week or more	Never	Neither						1							
6663875	20's	Several times a week or more	Sometimes	Single Player	*1*	1			1								
6656798	20's	Several times a week or more	Sometimes	Multiplayer	*1*												
6663978	20's	Several times a week or more	Sometimes	Multiplayer	*1*												
6686019	20's	Several times a week or more	Never	Neither	*1*												
6685344	Teens or younger	Several times a week or more	Sometimes	Multiplayer	*1*			1		1	1				1		1
6684691	Teens or younger	Several times a week or more	Never	Neither								1					1
6657249	Teens or younger	Several times a week or more	Sometimes	Single Player						1							1
6657607	Teens or younger	Several times a week or more	Sometimes	Single Player		1											1
6684333	Teens or younger	Several times a week or more	Sometimes	Multiplayer				1		1		1	1	1	1		1
6685609	Teens or younger	Several times a week or more	Sometimes	Both	*1*		1	1				1	1	1	1	1	1
6656529	Teens or younger	Several times a week or more	Sometimes	Multiplayer						1	1	1	1		1		1
6686090	Teens or younger	Several times a week or more	Sometimes	Multiplayer	*1*			1		1		1	1	1	1		
6661327	Teens or younger	Several times a week or more	Sometimes	Multiplayer						1			1	1	1	1	
6684284	Teens or younger	Several times a week or more	Sometimes	Multiplayer	*1*			1			1	1		1	1		
6654744	Teens or younger	Several times a week or more	Never	Neither								1		1	1		
6684056	Teens or younger	Several times a week or more	Sometimes	Single Player	*1*		1	1	1	1	1	1	1		1		
6654738	Teens or younger	Several times a week or more	Sometimes	Multiplayer			1	1		1	1	1	1		1	1	
6669177	Teens or younger	Several times a week or more	Sometimes	Multiplayer				1				1	1		1		
6684286	Teens or younger	Several times a week or more	Sometimes	Single Player							1		1		1		
6654389	Teens or younger	Several times a week or more	Sometimes	Both	*1*		1			1	1	1			1	1	
6658453	Teens or younger	Several times a week or more	Sometimes	Multiplayer	*1*			1		1		1			1	1	
6654168	Teens or younger	Several times a week or more	Sometimes	Multiplayer				1		1	1				1	1	
6657773	Teens or younger	Several times a week or more	Sometimes	Multiplayer	*1*			1		1	1				1	1	
6667650	Teens or younger	Several times a week or more	Sometimes	Multiplayer	*1*			1			1				1		
6657404	Teens or younger	Several times a week or more	Sometimes	Single Player	*1*						1				1	1	
6654709	Teens or younger	Several times a week or more	Never	Neither											1		
6663859	Teens or younger	Several times a week or more	Sometimes	Multiplayer	*1*		1	1		1		1	1				
6657736	Teens or younger	Several times a week or more	Never	Neither	*1*					1		1	1				
6684462	Teens or younger	Several times a week or more	Sometimes	Multiplayer	*1*		1	1				1	1			1	
6655248	Teens or younger	Several times a week or more	Never	Multiplayer	*1*			1				1	1				
6654222	Teens or younger	Several times a week or more	Sometimes	Multiplayer								1	1			1	
6655406	Teens or younger	Several times a week or more	Sometimes	Multiplayer	*1*							1	1				

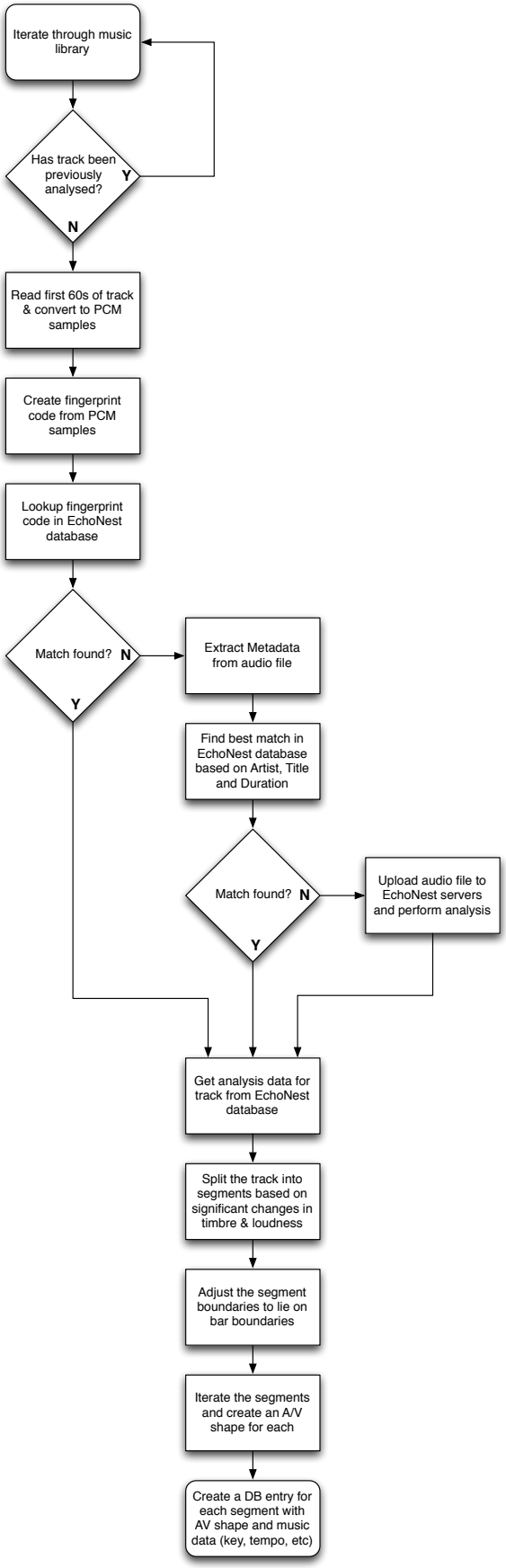
	7	6	1	2	3 In which genres of games would you be more likely to turn off music or listen to your own music?
	4				5
ID	Are there any games in particular that you would always turn off the music for? (optional)				Are there any games in particular that you think have such great music you could never imaging anyone not wanting
fb4	crash bandicoots, farmville				
fb5	Angry Birds				
fb9					
fb10	repetitive non story driven games, strategy games etc				Metal Gear Solid, Final Fantasy, Fallout- more soundscapey than music.
fb11					
fb12	Farmville				Wii
fb14	Farmville, Tetris				Gran Turismo
fb15					
fb16					Halo, Mass Effect 1 + 2, Arkham Asylum - all atmospheric and needed to play properly
6663956					
6657510					
6654826					Street Fighter III: 3rd Strike, Super Mario Galaxy
6686062	Pokemon				
6657275					Mass Effect 2, Super Meat Boy
6658049	Not one that I can think of.				Starcraft 2, Torchlight
6685660					
fb1	Call of Duty: Zombie Maps				Shadow of the Colossus
fb2	The only one that i didnt tick is Prototype, just because the music is pretty rubbish/doesn't add to the gameplay really.				La.Noir, Call of duty (single player), F.E.A.R & FEAR2
fb3					
fb7	pc games				call of duty moder wafer
fb8					
fb13	FIFA (In multiplayer), Unreal Tournament, Burnout				God of War (entire series), Super Mario Galaxy, Heavy Rain, Grand Theft Auto (although I also enjoy the 'User Tracks' option)
fb17					
6658383	games where the music was clearly an afterthought, or music that doesn't support the mood of the game (bullshit cash-in soundtracks)				Shatter, Kings Bounty: Armored Princess
6656765	MMOs.				
6654967	Basically, any genre of games that exceed 50 hours of total play, (grind heavy, endurance races etc)				Demon's Souls, Mass Effect 1&2, Red Dead Redemption for it's dynamic implementation of sound
6657821	none				CHRONO TRIGGER
6685675					Halo series and Gears of War series
6655191					
6656830					
6655307	NFS				FF
6657202	puzzle games				
6667841	MMOs after few weeks				Trine, Puzzle Quest, Super Meat Boy, VVVVVV
6656809	Super Meat Boy. It has great music, but now that I've heard it all, my own music/podcast distracts me from the grind of constant death.				Any sort of interactive experience where you are inhabiting a world.
6663921	Mobile games mostly				Metal Gear series, Silent Hill series, God of War series, Final Fantasy series
6663931					
6657075					Guild Wars, The Elder Scrolls, Final Fantasy, Super Meat Boy
6654757	Diablo 2				Braid
6661363					
6684423	World of Warcraft				Final Fantasy series
6654685	World Of Warcraft				Zelda Franchise
6655490	MMOs				Adventure Games
6658446	World of Warcraft				Persona 3, Persona 4, L.A. Noire, Red Dead Redemption
6657690					
6686089					Chrono Cross, any Zelda game, VVVVVV,
6664107					
6684444					The Witcher, Chrono Trigger, Castlevania: Symphony of the Night.
6654969					
6664069	Sports				

	4	5
ID	Are there any games in particular that you would always turn off the music for? (optional)	Are there any games in particular that you think have such great music you could never imaging anyone not wanting
6654760	Mount and Blade: Warband	
6654635		All the Final Fantasy and Metal Gear Solid games.
6664374	games with bad music	no, those people are stupid
6654301		The Final Fantasy series. And Mass Effect.
6684158		
6663858	Forza	Nope
6686692		
6684403	MMORPGs	Mass Effect 1 and 2, Metal Gear Solid 1, 2, 3, 4, Peace Walker
6658385	Oblivion	S.T.A.L.K.E.R.
6655160		
6654396	Street Fighter, MvC, Starcraft 2 (keep on FX though for warnings)	most RPGs, adventure games, and platformers (read: Final Fantasy, Mario, Zelda, etc.)
6668530		
6656354	Civilizaion because it has such long playsessions	Games where emotional connection is important or setpieces are integral to the design, I find is best experience with the original music on
6664194	I almost always prefer my own music for multiplayer games, games that I've already completed once, or grinding in RPGs.	
6684234		
6663990	Disgaea.	Rez, Fallout series.
6663922	Burnout	Metal Gear Solid
6657938	Starcraft 2 - multiplayer only, Call of Duty franchise - multiplayer only	Morrowind, despite the size and length of the game and the potential for the music to become tedious, I never got tired of listening to it.
6661409	Most MMOs (I listen to podcasts as I play)	Elder Scrolls IV: Oblivion, Fallout 3, Portal 2, Any Mario Game
6686584		
6664004		Super Meat Boy
6664041	If the soundtrack is predominantly death metal.	Anything in the Zelda series or Mario series.
6686652	All sports games after playing for about 20 hours	Final Fantasy games
6685727	FIFA	
6659813	Minecraft, most iOS games, Civilization games	
6655010	Games with bad music	Rhythm/Music games
6655559	World of Warcraft, Spiral Knights, Team Fortress 2 sometimes, etc. Multiplayer games where the focus is not on narrative.	
6656292	Ones with bad music. I rarely turn music off unless its annoying or just plain bad.	
6667041		
6663853		
6663812		
6654494		Rock Band
6686600	League of Legends, online FPS games...	Mass Effect 1 & 2, The Elder Scrolls series, Katamari Damacy, Shadow of the Colossus, Ocarina of Time, Deus Ex (if only for the title theme), Rez, Castlevania: Symphony of the Night
6684274	Halo, Any CoD game.	
6656226	Most PSN games come to mind since they are mostly arcade type games	RPG games, breath of fighter 4 has brilliant music.
6655597		
6685781	Call of Duty Black Ops Multiplayer	Metal gear Solid (1-4)
6657389	burnout paradise	
6655031		
6658500		
6657933		
6667306	Plants vs. Zombies, Peggle, Space Giraffe, Gridrunner Revolution	Bionic Commando Rearmed, Command and Conquer, Red Alert
6661459		
6657776	Tiny Tower	BlazBlue: Continuum Shift
6686562	Impossible Creatures	
6655017		
6656763		
6663974		
6684037		

	4	5
ID	Are there any games in particular that you would always turn off the music for? (optional)	Are there any games in particular that you think have such great music you could never imaging anyone not wanting
6685630		
6657349	World of Warcraft	
6656815		
6663989		
6684192		Anno 1404: Dawn Of Discovery
6660506		Katamari series
6663839		
6654418		
6655310		Child of Eden/Rez, Red Dead Redemption, Mass Effect 2, Assassin's Creed, VVVVVV
6686596	Less than average shooters	Super Meat Boy
6663953		
6656679		
6658315		
6658120	It's not so much as me caring for the music or not, but the storyline instead. If I don't care about the story line (ex. Vanquish), my music, or podcast, goes on.	Jet Set Radio Future, Chime, Elder Scrolls, Final Fantasy, Halo, Half-Life 2, GTA IV's radio stations
6654409	StarCraft II. I always have my own music or podcasts playing instead.	Burnout 3 has a great soundtrack.
6656421		
6684100	TrackMania Nations	Anything with music composed by Shoji Meguro, Koji Kondo, or Nobuo Uematsu
6654954		
6655041		
6657899		
6686056	Wipeout. But that's because you can make your own playlists for that game.	Persona 3&4
6657546	Black Ops multiplayer	Portal 2
6654394		
6654935		
6663886	Multiplayer PC games.	Shadow Of The Colossus
6658450		
6663875		
6656798	Depends on the soundtrack. If it's really generic, repetitive music, yes.	RPG soundtracks, usually.
6663978		
6686019		
6685344	League of Legends, RIFT, WoW	Mass Effect 2, Super Mario Galaxy 1+2, all of the Legend of Zelda games, Pokemon, Ace Attorney series
6684691		Bioshock, Dead Space
6657249	MMO's	
6657607	World of Warcraft	
6684333	Counter-Strike: Source,Garry's Mod,Trackmania Nations Forever	The Elder Scrolls: Morrowind,Batman: Arkham Asylum,Gemini Rue
6685609	Definitely racing games and multiplayer FPSs. Sometimes in long games like the Elder Scrolls, the music (though good) gets tired after a while so I'll put on the LOTR soundtrack.	Elder Scrolls has really appropriate music
6656529	Starcraft 2, trackmania, COD multiplayer.	The Witcher 2, Super Meat Boy, L.A. Noire, Red Dead Redemption.
6686090		Halo, Shadow of the Colossus, Zelda, Mario
6661327	Minecraft, Tetris	Most games
6684284	Street Fighter, Any Multiplayer Shooter. Games where atmosphere isn't a huge part or there is little dialogue.	Persona, Portal, Splinter Cell, Uncharted
6654744		
6684056		Super Meat Boy
6654738		Bioshock
6669177		Super Mario Bros., Scott Pilgrim, Super Meat Boy, Rhythm games of course.
6684286		Uncharted Series, Metal Gear
6654389		Zelda series, Metroid Prime series
6658453	Games that I play a whole lot of, since the music gets boring after a few hundred hours.	World of Warcraft, until you play it too much.
6654168		
6657773	Any type of MMO	
6667650	Only SC2 multiplayer.	The Persona series.
6657404		

	4	5
ID	Are there any games in particular that you would always turn off the music for? (optional)	Are there any games in particular that you think have such great music you could never imaging anyone not wanting
6654709		
6663859		
6657736		
6684462		
6655248		
6654222		
6655406	Gran Turismo 5	The Final Fantasy games.
6661457		
6661469		
6684236		
6658198	Borderlands	Mass Effect, Mass Effect 2, Super Mario Galaxy, Lost Odyssey, etc.
6663920	Dead or Alive 4	Shatter, Metal Gear Solid 3, Midnight Club: Los Angeles
6661435		Deus Ex
6658232		
6686916		
6657458		Motorstorm, Persona 4, Professor Layton
6663977	Team Fortress 2, Starcraft 2	Audiosurf :P
6661793	If I am playing a third person or first person shooter multiplayer, I will almost always turn the music off.	Bioshock has great atmospheric music that really helps set the game world up. Other games like Super Meat Boy just have cool music.
6656960		Super Smash Brothers Brawl, Super Mario Galaxy, Halo 3 ODST
6657229	Elder Scrolls Fallout	Halo 3: ODST
6658527	No, but I only ever turn music off while replaying a game for achievements/trophies so I can listen to my own music or podcasts	Flower, Pixeljunk Eden, Mirror's Edge, Silent Hill
6684399	Whatever I've already played through or where sound isn't integral to the experience (multiplayer for example)	Not unless the music is a focus of the game and plays a part in the mechanics
fb6	Turn off music for almost all games, cant stand the repetitive loop	Wip3out was the only one I can remember music being a critical part
6663971		

Appendix 2: Analysis Workflow



Appendix 3: Echo Nest/CAL500 Testing

3.1 Dictionary mapping CAL500 Emotion Tags to Echo Nest Moods

CAL500 Emotion Tags	Echo Nest Moods
Angry / Aggressive,	aggressive; angry; angst-ridden; harsh; industrial; rebellious;
Not Angry / Aggressive,	calming; fun; gentle; quiet; spiritual; warm;
Arousing / Awakening,	bouncy; dramatic; energetic; enthusiastic; epic; haunting; hypnotic; intense; lively; ominous; rebellious; rowdy; sexy;
Not Arousing / Awakening,	calming; gentle; mellow; quiet;
Bizarre / Weird,	complex; dark; disturbing; dreamy; eerie; harsh; haunting; manic; mystical; spacey; strange; trippy;
Not Bizarre / Weird,	carefree; cool; gentle;
Calming / Soothing,	ambient; calming; dreamy; gentle; cool; hypnotic; meditation; mellow; peaceful; relax; soothing; spiritual;
Not Calming / Soothing,	aggressive; angry; angst-ridden; cold; disturbing; dramatic; eerie; energetic; harsh; haunting; manic; ominous; rowdy;
Carefree / Lighthearted,	bouncy; carefree; dreamy; fun; funky; cool; humorous; light; playful; sweet; whimsical;
Not Carefree / Lighthearted,	aggressive; angst-ridden; complex; dark; eerie; industrial; intense; ominous; rebellious;
Cheerful / Festive,	cheerful; cool; fun; gleeful; groovy; humorous; joyous; lively; spiritual; sweet;
Not Cheerful / Festive,	cold; dark; gloomy; melancholia;
Emotional / Passionate,	dark; dramatic; energetic; enthusiastic; intense; lively; manic; passionate; poignant; rebellious; romantic; sexy; spiritual;
Not Emotional / Passionate,	cold; harsh; laid-back; mellow; peaceful;
Exciting / Thrilling,	complex; energetic; epic; dramatic; haunting; intense; lively; manic; sexy;
Not Exciting / Thrilling,	ambient; dreamy; gentle; laid-back; meditation; mellow; peaceful; quiet; relax;
Happy,	fun; gleeful; happy; joyous; warm;
Not Happy,	cold; cool; dark; disturbing; gloomy; harsh; dark; gloomy; melancholia; poignant; sad;
Laid-back / Mellow,	ambient; calming; cool; calming; laid-back; meditation; mellow; peaceful; reflective; relax; soothing; spacey; trippy;
Not Laid-back / Mellow,	aggressive; bouncy; complex; disturbing; dramatic; energetic; enthusiastic; harsh; intense; manic;
Light / Playful,	bouncy; fun; funky; groovy; carefree; light; playful; sweet; whimsical;
Not Light / Playful,	complex; dark; eerie; epic; industrial; intense; strange;
Loving / Romantic,	dramatic; elegant; intimate; mystical; passionate; poignant; romantic; sentimental; sexy; sweet; warm;
Not Loving / Romantic,	aggressive; angry; harsh; industrial;
Pleasant / Comfortable,	elegant; gleeful; happy; mellow; spacey; warm;
Not Pleasant / Comfortable,	cold; dark; disturbing; eerie; haunting; ominous;
Positive / Optimistic,	elegant; enthusiastic; funky; happy; warm;
Not Positive / Optimistic,	angst-ridden; gloomy; melancholia; strange;

```
Powerful / Strong,            complex; dramatic; energetic; epic; haunting; hypnotic; industrial;
                              intense; mystical; poignant; rebellious; rowdy; spiritual;

Not Powerful / Strong,        ambient; calming; gentle; laid-back; light; mellow; peaceful; quiet;

Sad,                          cold; cool; dark; disturbing; gloomy; harsh; dark; melancholia; sad;

Not Sad,                      fun; gleeful; happy; carefree; cheerful; joyous; warm;

Tender / Soft,                ambient; calming; gentle; dreamy; intimate; meditation; mellow; peaceful;
                              quiet; romantic; sentimental; soothing; sweet; warm;

Not Tender / Soft,            aggressive; angry; angst-ridden; complex; harsh; industrial; manic;

Touching / Loving,            dramatic; gentle; dreamy; intimate; passionate; poignant; reflective;
                              romantic; sentimental; sexy; spiritual; sweet; warm;

Not Touching / Loving,        aggressive; cold; dark; harsh;
```

3.2 CAL500 to Echo Nest Emotion Tag Comparison Code

```
int main (int argc, const char * argv[])
{
    NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
    NSTimeInterval startTime = [[NSDate date] timeIntervalSinceReferenceDate];
    NSMutableDictionary *combinedSongInformationByID;

    // arg 1 is the csv file with the artist, title and echo nest IDs for each CAL500 song
    NSString *songsFilePath = [NSString stringWithUTF8String:argv[1]];
    NSString *songsList = [NSString stringWithContentsOfFile:songsFilePath
                                                    encoding:NSUTF8StringEncoding
                                                       error:nil];

    NSArray *songs = [songsList componentsSeparatedByCharactersInSet:
                                                        [NSCharacterSet newlineCharacterSet]];

    if ([(NSString *)[songs lastObject] length] == 0)
    {
        songs = [songs subarrayWithRange:NSMakeRange(0, [songs count] -1)];
    }

    combinedSongInformationByID = [NSMutableDictionary dictionaryWithCapacity:[songs count]];


    // arg 2 is the csv file with table of vocab annotations for the CAL500 set
    NSString *annotationsTablePath = [NSString stringWithUTF8String:argv[2]];
    NSString *annotationsTable = [NSString stringWithContentsOfFile:annotationsTablePath
                                                           encoding:NSUTF8StringEncoding
                                                              error:nil];

    NSArray *annotationRows = [annotationsTable componentsSeparatedByCharactersInSet:
                                                                [NSCharacterSet newlineCharacterSet]];
    if ([(NSString *)[annotationRows lastObject] length] == 0)
    {
        annotationRows = [annotationRows subarrayWithRange:NSMakeRange(0, [annotationRows count] -1)];
    }


    // arg 3 is the file with the list of emotion vocab terms
    NSString *calEmotionsTagsPath = [NSString stringWithUTF8String:argv[3]];
    NSString *calEmotionsTagsList = [NSString stringWithContentsOfFile:calEmotionsTagsPath
                                                              encoding:NSUTF8StringEncoding
                                                                 error:nil];

    NSArray *calEmotionTags = [calEmotionsTagsList componentsSeparatedByCharactersInSet:
                                                                [NSCharacterSet newlineCharacterSet]];
    if ([(NSString *)[calEmotionTags lastObject] length] == 0)
    {
        calEmotionTags = [calEmotionTags subarrayWithRange:NSMakeRange(0, [songs count] -1)];
    }
```

```
// iterate through the CAL500 annotations and extract emotion tags per song
NSUInteger rowIndex;
for(rowIndex = 0; rowIndex < [songs count]; rowIndex++)
{
    NSUInteger columnIndex;
    NSMutableSet *cal500tags = [NSMutableSet set];
    NSArray *columnValues = [(NSString *)[annotationRows objectAtIndex:rowIndex]
                                                componentsSeparatedByString:@","];

    for (columnIndex = 0; columnIndex < 36; columnIndex++)
    {
        if([[columnValues objectAtIndex:columnIndex] isEqualToString:@"1"])
        {
            [cal500tags addObject:[calEmotionTags objectAtIndex:columnIndex]];
        }
    }

    NSMutableDictionary *songInformationDictionary = [NSMutableDictionary dictionary];
    [songInformationDictionary setObject:cal500tags forKey:@"cal500tags"];

    [combinedSongInformationByID setObject:songInformationDictionary
                                    forKey:[[[songs objectAtIndex:rowIndex]
                                                componentsSeparatedByString:@","] lastObject]];
}



// arg 4 is the file mapping CAL500 mood tags to echonest mood tags
NSString *moodTagMatchesFilePath = [NSString stringWithUTF8String:argv[4]];
NSString *moodTagMatchesFileContents = [NSString stringWithContentsOfFile:moodTagMatchesFilePath
                                                                  encoding:NSUTF8StringEncoding
                                                                     error:nil];

NSMutableDictionary *enMoodToCAL500MoodTranslations = [NSMutableDictionary dictionary];

for (NSString *line in [moodTagMatchesFileContents componentsSeparatedByCharactersInSet:
                                                        [NSCharacterSet newlineCharacterSet]])
{
    if([line length] > 0)
    {
        NSString *cal500MoodTag = [[line componentsSeparatedByString:@", "] objectAtIndex:0];
        NSArray *echoNestTags = [[[line componentsSeparatedByString:@", "] objectAtIndex:1]
                                                componentsSeparatedByString:@";"];

        for (NSString *echoNestTag in echoNestTags)
        {
            if ([echoNestTag length] > 0)
            {

                if ([enMoodToCAL500MoodTranslations objectForKey:echoNestTag] == nil)
                {
                    [enMoodToCAL500MoodTranslations setObject:[NSMutableSet set]
                                                       forKey:echoNestTag];
                }
                NSMutableSet *matchingCAL500Tags = [enMoodToCAL500MoodTranslations
                                                                    objectForKey:echoNestTag];
                [matchingCAL500Tags addObject:cal500MoodTag];
            }
        }
    }
}


// iterate through the list of echonest moods and see which matches in the
// CAL500 songs set echonest finds
for(NSString *mood in [enMoodToCAL500MoodTranslations allKeys])
{
    NSUInteger serverResultsStartIndex = 0;
    NSUInteger resultsPerQuery = 15;
    BOOL moreResultsAvailable = YES;
```

```
        while (moreResultsAvailable)
        {
            NSString *urlString = [NSString stringWithFormat:@"http://developer.echonest.com/api/v4/
song/search?api_key=D3MKHMOAM5QLRCAT6&mood=%@&bucket=id:CADWAVJ130D48CB527&limit=true&results=%d&start=
%d", mood, resultsPerQuery, serverResultsStartIndex];
            NSData *serverResponse = [NSData dataWithContentsOfURL:[NSURL URLWithString:urlString]];

            NSArray *matchingSongDicts = [[[serverResponse yajl_JSON] valueForKey:@"response"]
                                                                      valueForKey:@"songs"];

            for(NSDictionary *matchingSongDict in matchingSongDicts)
            {
                NSString *songID = [matchingSongDict objectForKey:@"id"];
                NSMutableDictionary *songInfo = [combinedSongInformationByID objectForKey:songID];
                [songInfo setObject:[matchingSongDict objectForKey:@"title"] forKey:@"title"];
                [songInfo setObject:[matchingSongDict objectForKey:@"artist_name"] forKey:@"artist"];

                if ([songInfo objectForKey:@"echoNestTags"] == nil)
                {
                    [songInfo setObject:[NSMutableSet set] forKey:@"echoNestTags"];
                }

                NSMutableSet *echoNestTags = [songInfo objectForKey:@"echoNestTags"];
                [echoNestTags unionSet:[enMoodToCAL500MoodTranslations valueForKey:mood]];
            }

            if([matchingSongDicts count] == resultsPerQuery)
            {
                serverResultsStartIndex += resultsPerQuery;
                NSLog(@"More results available for %@", mood);
            }
            else
            {
                moreResultsAvailable = NO;
            }
        }
    }



    // iterate through the combined data and compare the CAL500 and Echonest tags,
    // prepare the data for output

    NSMutableString *csv = [NSMutableString string];
    [csv appendString:@"Song ID,Artist,Title,CAL500 Tags,Echonest Tags,Match\n"];

    for(NSString *songID in combinedSongInformationByID)
    {
        NSMutableDictionary *songInfo = [combinedSongInformationByID objectForKey:songID];

        [csv appendFormat:@"%@,\"%@\",\"%@\",", songID,
                                                [songInfo objectForKey:@"artist"],
                                                [songInfo objectForKey:@"title"]];

        [csv appendString:@"\""];
        for(NSString *tag in [songInfo objectForKey:@"cal500tags"])
        {
            [csv appendFormat:@"%@, ", tag];
        }
        [csv appendString:@"\","];

        [csv appendString:@"\""];
        for(NSString *tag in [songInfo objectForKey:@"echoNestTags"])
        {
            [csv appendFormat:@"%@, ", tag];
        }
        [csv appendString:@"\","];

        NSMutableSet *commonTags = [NSMutableSet setWithSet:[songInfo objectForKey:@"cal500tags"]];
        [commonTags intersectSet:[songInfo objectForKey:@"echoNestTags"]];
```

```
        double matchScore = (double)([commonTags count]/[[songInfo objectForKey:@"cal500tags"] count]);
        [csv appendFormat:@"%f", matchScore];

        [csv appendString:@"\n"];
    }


    [csv writeToFile:[@"~/Desktop/ComparisonResults.csv" stringByStandardizingPath]
          atomically:YES
            encoding:NSUTF8StringEncoding
               error:nil];


    NSLog(@"Completed in %3f seconds", [[NSDate date] timeIntervalSinceReferenceDate] - startTime);

    [pool drain];
    return 0;
}
```

3.3 CAL500 Emotion Tag Vector Mapping

CAL500 Emotion Tag	Vector Angle, Magnitude
Emotion-Angry_/_Agressive	: 315.0, 0.75
NOT-Emotion-Angry_/_Agressive	: 135.0, 0.5
Emotion-Arousing_/_Awakening	: 10.0, 0.75
NOT-Emotion-Arousing_/_Awakening	: 190.0, 0.5
Emotion-Bizarre_/_Weird	: 280.0, 0.3
NOT-Emotion-Bizarre_/_Weird	: 150.0, 0.3
Emotion-Calming_/_Soothing	: 155.0, 0.75
NOT-Emotion-Calming_/_Soothing	: 335.0, 0.5
Emotion-Carefree_/_Lighthearted	: 100.0, 0.5
NOT-Emotion-Carefree_/_Lighthearted	: 280.0, 0.5
Emotion-Cheerful_/_Festive	: 80.0, 0.5
NOT-Emotion-Cheerful_/_Festive	: 260.0, 0.5
Emotion-Emotional_/_Passionate	: 30.0, 0.75
NOT-Emotion-Emotional_/_Passionate	: 150.0, 0.5
Emotion-Exciting_/_Thrilling	: 0.0, 0.8
NOT-Emotion-Exciting_/_Thrilling	: 180.0, 0.5
Emotion-Happy	: 80.0, 0.9
NOT-Emotion-Happy	: 260.0, 0.5
Emotion-Laid-back_/_Mellow	: 135.0, 0.75
NOT-Emotion-Laid-back_/_Mellow	: 315.0, 0.5
Emotion-Light_/_Playful	: 80.0, 0.3,
NOT-Emotion-Light_/_Playful	: 260.0, 0.3
Emotion-Loving_/_Romantic	: 110.0, 0.5
NOT-Emotion-Loving_/_Romantic	: 290.0, 0.3
Emotion-Pleasant_/_Comfortable	: 130.0, 0.5

```
NOT-Emotion-Pleasant_/_Comfortable    : 310.0,  0.3
Emotion-Positive_/_Optimistic         : 90.0,   0.75
NOT-Emotion-Positive_/_Optimistic     : 270.0,  0.5
Emotion-Powerful_/_Strong             : 0.0     0.75
NOT-Emotion-Powerful_/_Strong         : 180.0,  0.5
Emotion-Sad                           : 270.0,  0.9
NOT-Emotion-Sad                       : 90.0,   0.5
Emotion-Tender_/_Soft                 : 130.0,  0.5
NOT-Emotion-Tender_/_Soft             : 320.0,  0.3
Emotion-Touching_/_Loving             : 120.0,  0.75
NOT-Emotion-Touching_/_Loving         : 330.0,  0.5
```

3.4 CAL500 to Echo Nest Arousal Valence Shape Comparison Code

```
typedef struct _MWVector {
    double angle;
    double magnitude;
} MWVector; // angle is in degrees

#define MWZeroVector MWMakeVector(0.0, 0.0)

MWVector MWMakeVector(double angle, double magnitude);
MWVector vectorForPoints(NSPoint startPoint, NSPoint endPoint);
NSPoint offsetForVector(MWVector vector);
MWVector averageVectors(MWVector *vectorsArray, NSUInteger arrayCount);

double degreesToRadians(double degreeValue);
double raidansToDegrees(double radianValue);


int main (int argc, const char * argv[])
{
    NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];

    NSTimeInterval startTime = [[NSDate date] timeIntervalSinceReferenceDate];

    // arg 1 is file with vectors for each CAL500 Emotion Tag
    NSString *cal_emotionVectorsFilePath = [NSString stringWithUTF8String:argv[1]];
    NSString *cal_emotionVectorFileContents = [NSString stringWithContentsOfFile:emotionVectorsFilePath
                                                                         encoding:NSUTF8StringEncoding
                                                                            error:nil];

    NSArray *cal_emotionVectorFileLines =
                            [cal_emotionVectorFileContents componentsSeparatedByCharactersInSet:
                             [NSCharacterSet newlineCharacterSet]];

    if ([(NSString *)[cal_emotionVectorFileLines lastObject] length] == 0)
    {
        cal_emotionVectorFileLines = [cal_emotionVectorFileLines subarrayWithRange:
                                                    NSMakeRange(0, [cal_emotionVectorFileLines count] -1)];
    }

    NSUInteger cal_totalEmotionsCount = [cal_emotionVectorFileLines count];


    // Create a vector for each emotion tag
    MWVector *cal_emotionVectors = calloc(cal_totalEmotionsCount, sizeof(MWVector));

    NSUInteger lineNum = 0;
    for (NSString *line in cal_emotionVectorFileLines)
    {
```

```
        double angle, magnitude;

        NSScanner *scanner = [NSScanner scannerWithString:line];
        [scanner scanUpToString:@":" intoString:NULL];
        [scanner scanString:@":" intoString:NULL];
        [scanner scanDouble:&angle];
        [scanner scanString:@"," intoString:NULL];
        [scanner scanDouble:&magnitude];

        MWVector emotionVector = MWMakeVector(angle, magnitude);
        cal_emotionVectors[lineNum] = emotionVector;
        lineNum++;
    }


    // arg 2 is the file with the song names and IDs
    NSString *songNamesFilePath = [NSString stringWithUTF8String:argv[2]];
    NSString *songNamesFileContent = [NSString stringWithContentsOfFile:songNamesFilePath
                                                                encoding:NSUTF8StringEncoding
                                                                   error:nil];

    NSArray *songNamesAndIDs = [songNamesFileContent componentsSeparatedByCharactersInSet:
                                                            [NSCharacterSet newlineCharacterSet]];
    if ([(NSString *)[songNamesAndIDs lastObject] length] == 0)
    {
        songNamesAndIDs = [songNamesAndIDs subarrayWithRange:NSMakeRange(0,[songNamesAndIDs count] -1)];
    }



    // arg 3 is the CAL500 dataset csv
    NSString *cal500FilePath = [NSString stringWithUTF8String:argv[3]];
    NSString *cal500FileContents = [NSString stringWithContentsOfFile:cal500FilePath
                                                              encoding:NSUTF8StringEncoding
                                                                 error:nil];

    NSArray *cal500Rows = [cal500FileContents componentsSeparatedByCharactersInSet:
                                                          [NSCharacterSet newlineCharacterSet]];
    if ([(NSString *)[cal500Rows lastObject] length] == 0)
    {
        cal500Rows = [cal500Rows subarrayWithRange:NSMakeRange(0, [cal500Rows count] -1)];
    }



    // Iterate through the CAL500 set and create and average emotion vector based on the emotion tags
    // for each song.  Use the average vector to calculate an AV point and compare with AV points
    // derived from echonest analyses.

    NSMutableString *outputCSV = [NSMutableString string];
    [outputCSV appendString:@"Song,CAL500 average vector angle,CAL500 average vector magnitude,CAL500
valence,CAL500 arousal, CAL500 confidence,EN Valence,EN arousal,EN confidence,Difference between
centres,Difference between edges,Intersects\n"];

    MWVector *cal_songEmotionVectors = calloc(cal_totalEmotionsCount, sizeof(MWVector));
    NSUInteger rowIndex = 0;
    for (NSString *row in cal500Rows)
    {
        NSLog(@"Song %lu of %lu", rowIndex +1, [cal500Rows count]);

        // for each song iterate the tags and gather all the applicable emotion vectors
        NSArray *columns = [row componentsSeparatedByString:@","];
        NSUInteger columnIndex, cal_songEmotionsCount = 0;
        for (columnIndex = 0; columnIndex < cal_totalEmotionsCount; columnIndex++)
        {
            if([[columns objectAtIndex:columnIndex] integerValue] == 1)
            {
                cal_songEmotionVectors[cal_songEmotionsCount] = cal_emotionVectors[columnIndex];
                cal_songEmotionsCount++;
            }
        }
```

```
        // calculate the average vector for the song and convert to a point in AV space
        MWVector cal_songAverageEmotionVector = averageVectors(cal_songEmotionVectors,
                                                               cal_songEmotionsCount);
        NSPoint cal500AVLocation = offsetForVector(cal_songAverageEmotionVector);
        double cal500Confidence = (double)cal_songEmotionsCount /((double)cal_totalEmotionsCount / 2.0);


        // get the echonest anlysis results for the song
        NSString *songID = [[[songNamesAndIDs objectAtIndex:rowIndex]
                                componentsSeparatedByString:@","] lastObject];

        NSURL *en_audioSummaryURL = [NSURL URLWithString:[NSString stringWithFormat:@"http://
developer.echonest.com/api/v4/song/profile?api_key=D3MKHMOAM5QLRCAT6&format=json&id=
%@&bucket=audio_summary", songID]];

        NSDictionary *en_audioSummary = [[[[[[NSData dataWithContentsOfURL:en_audioSummaryURL]yajl_JSON]
                                          valueForKey:@"response"] valueForKey:@"songs"]
                                              objectAtIndex:0] valueForKey:@"audio_summary"];

        NSURL *en_fullAnalysisURL = [NSURL URLWithString:[en_audioSummary valueForKey:@"analysis_url"]];
        NSDictionary *en_fullAnalysis = [[NSData dataWithContentsOfURL:en_fullAnalysisURL] yajl_JSON];

        double enValenceValue;
        double enConfidence = [[[en_fullAnalysis valueForKey:@"track"] valueForKey:@"mode_confidence"]
                                doubleValue];
        if ([[en_audioSummary valueForKey:@"mode"] integerValue])
        {
            enValenceValue = enConfidence;
        }
        else
        {
            enValenceValue = 0.0 - enConfidence;
        }

        double enEnergyValue = ([[en_audioSummary valueForKey:@"energy"] doubleValue] * 2.0) - 1.0;

        NSPoint enAVLocation = NSMakePoint(enValenceValue, enEnergyValue);


        // compare the 2 AV locations
        // measure the distance between the 2 centres, between the 2 edges, and note overlaps

        double differenceBetweenCentres = vectorForPoints(cal500AVLocation, enAVLocation).magnitude;

        double calRadius = 1.0 - cal500Confidence, enRadius = 1.0 - enConfidence;
        double differenceBetweenEdges = differenceBetweenCentres - (calRadius + enRadius);

        NSInteger intersects = (differenceBetweenEdges <= 0.0) ? 1 : 0;


        // add the data to the output file
        [outputCSV appendFormat:@"%@,%f,%f,%f,%f,%f,%f,%f,%f,%f,%f,%d\n", songID,
cal_songAverageEmotionVector.angle, cal_songAverageEmotionVector.magnitude, cal500AVLocation.x,
cal500AVLocation.y, cal500Confidence, enAVLocation.x, enAVLocation.y, enConfidence,
differenceBetweenCentres,differenceBetweenEdges,intersects];

        rowIndex++;
    }

    NSLog(@"Finished in %2f seconds", [[NSDate date] timeIntervalSinceReferenceDate] - startTime);

    [outputCSV writeToFile:[@"~/Desktop/CAL500-EN AV Comparison.csv" stringByStandardizingPath]
                atomically:YES
                  encoding:NSUTF8StringEncoding
                     error:nil];

    free(cal_songEmotionVectors);
    free(cal_emotionVectors);

    [pool drain];
    return 0;
}
```

```
MWVector MWMakeVector(double angle, double magnitude)
{
    MWVector vector;
    vector.angle = angle;
    vector.magnitude = magnitude;
    return vector;
}


MWVector vectorForPoints(NSPoint startPoint, NSPoint endPoint)
{
    double xOffset = endPoint.x - startPoint.x;
    double yOffset = endPoint.y - startPoint.y;

    if( (xOffset == 0.0) && (yOffset == 0.0) )
    {
        return MWZeroVector;
    }

    if(xOffset == 0.0)
    {
        if (yOffset > 0.0)
        {
            return MWMakeVector(0.0, yOffset);
        }
        else
        {
            return MWMakeVector(180.0, yOffset);
        }
    }

    if(yOffset == 0.0)
    {
        if (xOffset > 0.0)
        {
            return MWMakeVector(90.0, xOffset);
        }
        else
        {
            return MWMakeVector(270, xOffset);
        }
    }


    NSInteger quadrant = 0;

    if( (xOffset > 0.0) && (yOffset < 0.0) )
    {
        quadrant = 1;
    }
    else if( (xOffset < 0.0) && (yOffset < 0.0) )
    {
        quadrant = 2;
    }
    else if( (xOffset < 0.0) && (yOffset > 0.0) )
    {
        quadrant = 3;
    }

    double hypotenuse, opposite, adjacent;
    switch (quadrant)
    {
        case 0:
        {
            adjacent = yOffset;
            opposite = xOffset;
            break;
        }
        case 1:
        {
```

```
            adjacent = xOffset;
            opposite = -yOffset;
            break;
        }
        case 2:
        {
            adjacent = -yOffset;
            opposite = -xOffset;
            break;
        }
        case 3:
        {
            adjacent = -xOffset;
            opposite = yOffset;
            break;
        }
        default:
            break;
    }

    hypotenuse = hypot(adjacent, opposite);

    double angleInRadians = asin(opposite / hypotenuse);
    double angleInDegrees = raidansToDegrees(angleInRadians);

    switch (quadrant)
    {
        case 1:
            angleInDegrees += 90.0;
            break;
        case 2:
            angleInDegrees += 180.0;
            break;
        case 3:
            angleInDegrees += 270.0;
            break;
        default:
            break;
    }

    return MWMakeVector(angleInDegrees, hypotenuse);
}


NSPoint offsetForVector(MWVector vector)
{
    double vAngle = vector.angle, vMagnitude = vector.magnitude;

    if( (vAngle < 0.0) || (vAngle > 360.0) )
    {
        NSLog(@"Invalid vector angle: %f", vAngle);
        return NSZeroPoint;
    }

    if(vAngle == 0.0)
    {
        return NSMakePoint(0.0, (CGFloat)vMagnitude);
    }

    if(vAngle == 90.0)
    {
        return NSMakePoint((CGFloat)vMagnitude, 0.0);
    }

    if(vAngle == 180.0)
    {
        return NSMakePoint(0.0, -(CGFloat)vMagnitude);
    }

    if(vAngle == 270.0)
    {
        return NSMakePoint(-(CGFloat)vMagnitude, 0.0);
```

```
    }

    double workingAngle = vAngle;
    NSInteger quadrant = 0;

    if( (vAngle > 90.0) && (vAngle < 180.0) )
    {
        workingAngle -= 90.0;
        quadrant = 1;
    }
    else if( (vAngle > 180.0) && (vAngle < 270.0) )
    {
        workingAngle -= 180.0;
        quadrant = 2;
    }
    else if( (vAngle > 270.0) && (vAngle < 360.0) )
    {
        workingAngle -= 270.0;
        quadrant = 3;
    }


    double adjacentHypotenuseRatio = cos(degreesToRadians(workingAngle));
    double oppositeHypotenuseRatio = sin(degreesToRadians(workingAngle));
    double hypotenuse = vMagnitude;
    double adjacent = hypotenuse * adjacentHypotenuseRatio;
    double opposite = hypotenuse * oppositeHypotenuseRatio;


    switch (quadrant)
    {
        case 0:
            return NSMakePoint((CGFloat)opposite, (CGFloat)adjacent);
            break;
        case 1:
            return NSMakePoint((CGFloat)adjacent, (CGFloat)-opposite);
            break;
        case 2:
            return NSMakePoint((CGFloat)-opposite, (CGFloat)-adjacent);
            break;
        case 3:
            return NSMakePoint((CGFloat)-adjacent, (CGFloat)opposite);
            break;
        default:
            return NSZeroPoint;
            break;
    }
}


MWVector averageVectors(MWVector *vectorsArray, NSUInteger arrayCount)
{
    if (arrayCount == 0)
    {
        return MWZeroVector;
    }

    NSPoint averageVectorTailPoint = NSZeroPoint;
    NSUInteger i;
    for (i = 0; i < arrayCount; i++)
    {
        averageVectorTailPoint.x += offsetForVector(vectorsArray[i]).x;
        averageVectorTailPoint.y += offsetForVector(vectorsArray[i]).y;
    }

    averageVectorTailPoint.x /= arrayCount;
    averageVectorTailPoint.y /= arrayCount;

    MWVector averageVector = vectorForPoints(NSZeroPoint, averageVectorTailPoint);

    return averageVector;
}
```

```
double degreesToRadians(double degreeValue)
{
    return degreeValue * (M_PI / 180.0);
}


double raidansToDegrees(double radianValue)
{
    return radianValue / (M_PI / 180.0);
}
```

Appendix 4: Harmonious API

4.1 System Controller Header

```
//
//  SystemController.h
//  Harmonious
//
//  Created by Jim McGowan on 16/07/2011.
//
//     This class is the main interface to the harmonious system.
//     It manages the Window with the system's UI and controls the music
//     database, analysis and playback systems.
//

@interface HarmoniousSystemController : NSWindowController <GameAnalyserDelegate>

// Initializing the system.  The system controller does not hold a reference to the game controller,
// it is just passed on to the game analyser
- (id)initWithGameController:(id)gameController;


// Showing the UI
- (void)showHarmoniousUIWindow;
- (IBAction)toggleFineTuneView:(id)sender;
@property (readonly) BOOL activateUIControls;


// starting and stopping music playback
- (void)startMusicPlayback;
- (void)stopMusicPlaybackWithFade:(BOOL)fadeOut;     // if fadeOut is NO, music stops immediately
- (IBAction)pauseMusic:(id)sender;
- (IBAction)skipTrack:(id)sender;


// triggering playback of a new music segment based on the game analyser's analysis
- (void)playNewTrackForArousalValencePoint:(NSPoint)avPoint;


// updating the arousal valence representation for a music segment
- (IBAction)likeMusic:(id)sender;
- (IBAction)dislikeMusic:(id)sender;
- (IBAction)banMusic:(id)sender;
- (IBAction)adjustHappySadSlider:(id)sender;
- (IBAction)adjustEnergeticLaidBackSlider:(id)sender;


@end
```

4.2 Game Analyser Header

```
//
//  GameAnalyser.h
//  Space Defender + Harmonious
//
//  Created by Jim McGowan on 12/8/11.
//
//     While the rest of the Harmonious system is generic and 'game-agnostic',
//     this game analyser class is written specifically to analyse the
//     host game.  The methods declared here are required by the system,
//     though the implementation details any other functionality are at the
//     programmer's discretion.
//


// These are the methods that the game analyser can call on its delegate (the system controller)
// to trigger cues
@protocol GameAnalyserDelegate <NSObject>
- (void)playNewTrackForArousalValencePoint:(NSPoint)avPoint;
- (void)fadeOutAndStopCurrentTrack;
@end



@interface GameAnalyser : NSObject

// Initializing the analyser and setting the controller & delegate
- (id)initWithGameController:(id)theGameController;
@property (assign) id <GameAnalyserDelegate> delegate;


// Activating the game analyser will cause it to start sending music trigger messages
// (-playNewTrackForArousalValencePoint: & -fadeOutAndStopCurrentTrack) to it's delegate
- (void)activate;
- (void)deactivate;
- (BOOL)isActive;


// Accessing the current gameplay mood as a point in Arousal/Valence space
// For Arousal/Valence locations represented as NSPoints, x = valence, y = arousal
- (NSPoint)currentGameplayMoodInArousalValenceCoordinates;

@end
```

Appendix 5: License Information for Third Party Software used by the Harmonious System

5.1 EchoPrint CodeGen

Whitening, SubbandAnalysis, Fingerprint
Dan Ellis <dpwe@ee.columbia.edu>
Brian Whitman <brian@echonest.com>

AudioBufferInput, AudioStreamInput, Codegen, Common, File, MatrixUtility, Metadata
Tristan Jehan <tristan@echonest.com>
Paul Lamere <paul@echonest.com>
Jason Sundram <jsundram@gmail.com>
Brian Whitman <brian@echonest.com>

Murmurhash2
Austin Appleby

Base64
Rene Nyffenegger

Contributors
Alastair Porter <alastair@porter.net.nz>efsavage
alsuren
artgillespie
yhorng
divan

echoprint-codegen is open source software licensed under the "MIT License"More information about the MIT License: http://en.wikipedia.org/wiki/MIT_License

Copyright (c) 2011 The Echo Nest Corporation

Permission is hereby granted, free of charge, to any person obtaining a copyof this software and associated documentation files (the "Software"), to dealin the Software without restriction, including without limitation the rightsto use, copy, modify, merge, publish, distribute, sublicense, and/or sellcopies of the Software, and to permit persons to whom the Software isfurnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included inall copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS ORIMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THEAUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHERLIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS INTHE SOFTWARE.

libcodegen makes use of the following pieces of software:
- Murmurhash by Austin Appleby (Public Domain / MIT)http://sites.google.com/site/murmurhash/
- Boost (Boost Software License)http://www.boost.org/users/license.html
- Base64.cpp and Base64.h, see source files for licenseCopyright (C) 2004-2008 Rene Nyffenegger

5.2 TagLib

Scott Wheeler <wheeler@kde.org>
Author, maintainer
Ismael Orenstein <orenstein@kde.org>
Xing header implementation
Allan Sandfeld Jensen <kde@carewolf.org>
FLAC metadata implementation
Teemu Tervo <teemu.tervo@gmx.net>
Numerous bug reports and fixes

Please send all patches and questions to taglib-devel@kde.org rather than toindividual developers!

GNU LESSER GENERAL PUBLIC LICENSE Version 2.1, February 1999

Copyright (C) 1991, 1999 Free Software Foundation, Inc. 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by

modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library `Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990 Ty Coon, President of Vice

That's all there is to it!

MOZILLA PUBLIC LICENSE Version 1.1

---

1. Definitions.

1.0.1. "Commercial Use" means distribution or otherwise making the Covered Code available to a third party.

1.1. "Contributor" means each entity that creates or contributes to the creation of Modifications.

1.2. "Contributor Version" means the combination of the Original Code, prior Modifications used by a Contributor, and the Modifications made by that particular Contributor.

1.3. "Covered Code" means the Original Code or Modifications or the combination of the Original Code and Modifications, in each case including portions thereof.

1.4. "Electronic Distribution Mechanism" means a mechanism generally accepted in the software development community for the electronic transfer of data.

1.5. "Executable" means Covered Code in any form other than Source Code.

1.6. "Initial Developer" means the individual or entity identified as the Initial Developer in the Source Code notice required by Exhibit A.

1.7. "Larger Work" means a work which combines Covered Code or portions thereof with code not governed by the terms of this License.

1.8. "License" means this document.

1.8.1. "Licensable" means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

1.9. "Modifications" means any addition to or deletion from the substance or structure of either the Original Code or any previous Modifications. When Covered Code is released as a series of files, a Modification is: A. Any addition to or deletion from the contents of a file containing Original Code or previous Modifications.

B. Any new file that contains any part of the Original Code or previous Modifications.

1.10. "Original Code" means Source Code of computer software code which is described in the Source Code notice required by Exhibit A as Original Code, and which, at the time of its release under this License is not already Covered Code governed by this License.

1.10.1. "Patent Claims" means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process, and apparatus claims, in any patent Licensable by grantor.

1.11. "Source Code" means the preferred form of the Covered Code for making modifications to it, including all modules it contains, plus any associated interface definition files, scripts used to control compilation and installation of an Executable, or source code differential comparisons against either the Original Code or another well known, available Covered Code of the Contributor's choice. The Source Code can be in a compressed or archival form, provided the appropriate decompression or de-archiving software is widely available for no charge.

1.12. "You" (or "Your") means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License or a future version of this License issued under Section 6.1. For legal entities, "You" includes any entity which controls, is controlled by, or

is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. Source Code License.

2.1. The Initial Developer Grant. The Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license, subject to third party intellectual property claims: (a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer to use, reproduce, modify, display, perform, sublicense and distribute the Original Code (or portions thereof) with or without Modifications, and/or as part of a Larger Work; and

(b) under Patents Claims infringed by the making, using or selling of Original Code, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Code (or portions thereof).

(c) the licenses granted in this Section 2.1(a) and (b) are effective on the date Initial Developer first distributes Original Code under the terms of this License.

(d) Notwithstanding Section 2.1(b) above, no patent license is granted: 1) for code that You delete from the Original Code; 2) separate from the Original Code; or 3) for infringements caused by: i) the modification of the Original Code or ii) the combination of the Original Code with other software or devices.

2.2. Contributor Grant. Subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license

(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor, to use, reproduce, modify, display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof) either on an unmodified basis, with other Modifications, as Covered Code and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: 1) Modifications made by that Contributor (or portions thereof); and 2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

(c) the licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first makes Commercial Use of the Covered Code.

(d) Notwithstanding Section 2.2(b) above, no patent license is granted: 1) for any code that Contributor has deleted from the Contributor Version; 2) separate from the Contributor Version; 3) for infringements caused by: i) third party modifications of Contributor Version or ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or 4) under Patent Claims infringed by Covered Code in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Application of License. The Modifications which You create or to which You contribute are governed by the terms of this License, including without limitation Section 2.2. The Source Code version of Covered Code may be distributed only under the terms of this License or a future version of this License released under Section 6.1, and You must include a copy of this License with every copy of the Source Code You distribute. You may not offer or impose any terms on any Source Code version that alters or restricts the applicable version of this License or the recipients' rights hereunder. However, You may include an additional document offering the additional rights described in Section 3.5.

3.2. Availability of Source Code. Any Modification which You create or to which You contribute must be made available in Source Code form under the terms of this License either on the same media as an Executable version or via an accepted Electronic Distribution Mechanism to anyone to whom you made an Executable version available; and if made available via Electronic Distribution Mechanism, must remain available for at least twelve (12) months after the date it initially became available, or at least six (6) months after a subsequent version of that particular Modification has been made available to such recipients. You are responsible for ensuring that the Source Code version remains available even if the Electronic Distribution Mechanism is maintained by a third party.

3.3. Description of Modifications. You must cause all Covered Code to which You contribute to contain a file documenting the changes You made to create that Covered Code and the date of any change. You must include a prominent statement that the Modification is derived, directly or indirectly, from Original Code provided by the Initial Developer and including the name of the Initial Developer in (a) the Source Code, and (b) in any notice in an Executable version or related documentation in which You describe the origin or ownership of the Covered Code.

3.4. Intellectual Property Matters (a) Third Party Claims. If Contributor has knowledge that a license under a third party's intellectual property rights is required to exercise the rights granted by such Contributor under Sections 2.1 or 2.2, Contributor must include a text file with the Source Code distribution titled "LEGAL" which describes the claim and the party making the claim in sufficient detail that a recipient will know whom to contact. If Contributor obtains such knowledge after the Modification is made available as described in Section 3.2, Contributor shall promptly modify the LEGAL file in all copies Contributor makes available thereafter and shall take other steps (such as notifying appropriate mailing lists or newsgroups) reasonably calculated to inform those who received the Covered Code that new knowledge has been obtained.

(b) Contributor APIs. If Contributor's Modifications include an application programming interface and Contributor has knowledge of patent licenses which are reasonably necessary to implement that API, Contributor must also include this information in the LEGAL file.

(c) Representations. Contributor represents that, except as disclosed pursuant to Section 3.4(a) above, Contributor believes that Contributor's Modifications are Contributor's original creation(s) and/or Contributor has sufficient rights to grant the rights conveyed by this License.

3.5. Required Notices. You must duplicate the notice in Exhibit A in each file of the Source Code. If it is not possible to put such notice in a particular Source Code file due to its structure, then You must include such notice in a location (such as a relevant directory) where a user would be likely to look for such a notice. If You created one or more Modification(s) You may add your name as a Contributor to the notice described in Exhibit A. You must also duplicate this License in any documentation for the Source Code where You describe recipients' rights or ownership rights relating to Covered Code. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Code. However, You may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear than any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.6. Distribution of Executable Versions. You may distribute Covered Code in Executable form only if the requirements of Section 3.1-3.5 have been met for that Covered Code, and if You include a notice stating that the Source Code version of the Covered Code is available under the terms of this License, including a description of how and where You have fulfilled the obligations of Section 3.2. The notice must be conspicuously included in any notice in an Executable version, related documentation or collateral in which You describe recipients' rights relating to the Covered Code. You may distribute the Executable version of Covered Code or ownership rights under a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable version does not attempt to limit or alter the recipient's rights in the Source Code version from the rights set forth in this License. If You distribute the Executable version under a different license You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or any Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.7. Larger Works. You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code.

4. Inability to Comply Due to Statute or Regulation.

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Code due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be included in the LEGAL file described in Section 3.4 and must be included with all distributions of the Source Code. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Application of this License.

This License applies to code to which the Initial Developer has attached the notice in Exhibit A and to related Covered Code.

6. Versions of the License.

6.1. New Versions. Netscape Communications Corporation ("Netscape") may publish revised and/or new versions of the License from time to time. Each version will be given a distinguishing version number.

6.2. Effect of New Versions. Once Covered Code has been published under a particular version of the License, You may always continue to use it under the terms of that version. You may also choose to use such Covered Code under the terms of any subsequent version of the License published by Netscape. No one other than Netscape has the right to modify the terms applicable to Covered Code created under this License.

6.3. Derivative Works. If You create or use a modified version of this License (which you may only do in order to apply it to code which is not already Covered Code governed by this License), You must (a) rename Your license so that the phrases "Mozilla", "MOZILLAPL", "MOZPL", "Netscape", "MPL", "NPL" or any confusingly similar phrase do not appear in your license (except to note that your license differs from this License) and (b) otherwise make it clear that Your version of the license contains terms which differ from the Mozilla Public License and Netscape Public License. (Filling in the name of the Initial Developer, Original Code or Contributor in the notice described in Exhibit A shall not of themselves be deemed to be modifications of this License.)

7. DISCLAIMER OF WARRANTY.

COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN

ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

8. TERMINATION.

8.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. All sublicenses to the Covered Code which are properly granted shall survive any termination of this License. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

8.2. If You initiate litigation by asserting a patent infringement claim (excluding declatory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You file such action is referred to as "Participant") alleging that:

(a) such Participant's Contributor Version directly or indirectly infringes any patent, then any and all rights granted by such Participant to You under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively, unless if within 60 days after receipt of notice You either: (i) agree in writing to pay Participant a mutually agreeable reasonable royalty for Your past and future use of Modifications made by such Participant, or (ii) withdraw Your litigation claim with respect to the Contributor Version against such Participant. If within 60 days of notice, a reasonable royalty and payment arrangement are not mutually agreed upon in writing by the parties or the litigation claim is not withdrawn, the rights granted by Participant to You under Sections 2.1 and/or 2.2 automatically terminate at the expiration of the 60 day notice period specified above.

(b) any software, hardware, or device, other than such Participant's Contributor Version, directly or indirectly infringes any patent, then any rights granted to You by such Participant under Sections 2.1(b) and 2.2(b) are revoked effective as of the date You first made, used, sold, distributed, or had made, Modifications made by that Participant.

8.3. If You assert a patent infringement claim against Participant alleging that such Participant's Contributor Version directly or indirectly infringes any patent where such claim is resolved (such as by license or settlement) prior to the initiation of patent infringement litigation, then the reasonable value of the licenses granted by such Participant under Sections 2.1 or 2.2 shall be taken into account in determining the amount or value of any payment or license.

8.4. In the event of termination under Sections 8.1 or 8.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or any distributor hereunder prior to termination shall survive termination.

9. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED CODE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

10. U.S. GOVERNMENT END USERS.

The Covered Code is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Code with only those rights set forth herein.

11. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by California law provisions (except to the extent applicable law, if any, provides otherwise), excluding its conflict-of-law provisions. With respect to disputes in which at least one party is a citizen of, or an entity chartered or registered to do business in the United States of America, any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California, with venue lying in Santa Clara County, California, with the losing party responsible for costs, including without limitation, court costs and reasonable attorneys' fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License.

12. RESPONSIBILITY FOR CLAIMS.

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

13. MULTIPLE-LICENSED CODE.

Initial Developer may designate portions of the Covered Code as "Multiple-Licensed". "Multiple-Licensed" means that the Initial Developer permits you to utilize portions of the Covered Code under Your choice of the NPL or the alternative licenses, if any, specified by the Initial Developer in the file described in Exhibit A.

EXHIBIT A -Mozilla Public License.

``The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.mozilla.org/MPL/

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is _____.

The Initial Developer of the Original Code is _____. Portions created by _____ are Copyright (C) _____ _____. All Rights Reserved.

Contributor(s): _____.

Alternatively, the contents of this file may be used under the terms of the _____ license (the "[___] License"), in which case the provisions of [_____] License are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of the [____] License and not to allow others to use your version of this file under the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the [___] License. If you do not delete the provisions above, a recipient may use your version of this file under either the MPL or the [___] License."

[NOTE: The text of this Exhibit A may differ slightly from the text of the notices in the Source Code files of the Original Code. You should use the text of this Exhibit A rather than the text found in the Original Code Source Code for Your Modifications.]

5.3 NSString-Levenshtein

NSString-Levenshtein.h/m

Created by Rick Bourner on Sat Aug 09 2003.
rick@bourner.com
http://www.merriampark.com/ldobjc.htm

This code is in the public domain.

5.4 YAJL-OBJC

Created by Gabriel Handford on 7/23/09. Copyright 2009. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.