

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Turning Music Into Game

Author:

[Paulina Koch](#)

Supervisor:

Dr. Iain Phillips

2nd Marker:

Dr. Robert Chatley

May 2015

Chapter 1

Abstract

Music games present a highly pervasive new platform to create, perform and appreciate music. In this project we will attempt creating a music rhythm game which, given a music track, extracts its features to generate a level without human intervention.

This report details the design of such a program and evaluates its effectiveness. The development of the program has lead to the discovery of new and powerful algorithms in music analysis, as well as successfully demonstrating the power of computers in developing creative works.

Chapter 2

Acknowledgments

I would like to thank my supervisor, Iain Phillips, for his sharp insight into the problems encountered, and his uncanny ability to immediately suggest a good solution for each one.

Contents

1	Abstract	1
2	Acknowledgments	2
	Contents	2
3	Introduction	5
4	Background	7
4.1	Music Video Games	7
4.1.1	Music Memory Games	8
4.1.2	Hybrid Music Games	8
4.1.3	Free Form Music Games	9
4.2	Case Study - Guitar Hero	9
4.2.1	The Controller	9
4.2.2	The Gameplay	10
4.2.3	The Critique	10
4.3	Introduction to Music Analysis	11
4.3.1	Pitch, Tones, Fundamental Frequency	11
4.3.2	Polyphonic Music	12
4.3.3	Melody	12
4.3.4	Filter	12
4.3.5	Short Time Fourier Transform	13
4.4	Main Melody Extraction from Polyphonic Music	13
4.4.1	Source Separation Based Approach	13
4.4.2	Salience Based Approaches	15
4.4.3	Comparison of both approaches	18
4.5	Introduction to Neural Networks	19
4.5.1	Models	20
4.6	Mood Detection	21
4.7	Level Generation	21
5	Implementation	22
5.1	Mood Detection	22
5.1.1	Choice of Features	23
5.1.2	Correlation Between Features and Mood Perception	24
5.1.3	Neural Network for Mood Prediction	25
5.2	Main Melody Extraction	27

5.3	The Game	27
5.3.1	Data Storage	27
5.3.2	Menu	28
5.3.3	Level Description	28
5.3.4	Melody Detection as a Game Changer	29
5.3.5	Impact of the Mood on the Level	29
5.4	Main Section 2	29
6	Evaluation	30
6.1	Formative	30
6.1.1	Single-Condition Study	30
6.2	Summative	31
6.2.1	Evaluation of Mood Detection system	31
6.2.2	Comparison to Original Songs	31
6.2.3	Melody Extraction Testing	32
6.2.4	Questionnaires	32

Chapter 3

Introduction

Music and games share a fundamental property: both are playable, offering their listeners and operators an expressive experience with the framework of melody and rhythm [1].

As the quote suggests, both games and music have one thing in common — the act of playing. Just as player's character might die in an attempt to complete a level, causing him to lose the game, the pianist can fail at the attempt of performing a musical piece.

Perhaps this analogy inspired programmers to develop a new genre of games - music games. Music games are games in which players interact with music. Possibly the most commonly known franchises in this genre are Guitar Hero, Rock Band and Dance Dance Revolution. In this type of games user has to follow the indicators on the screen telling him which buttons to hit.

The concept of a music game stormed the industry in 2005, after Guitar Hero was released. The project soon turned into the fastest new video game franchise to reach \$1 billion in retail sales in the history of the business, with Guitar Hero III being the first game to reach \$1 billion [2].

However, a limited amount of songs transcribed and adjusted to the game play soon caused the popularity of such music video games to decline. Some brave fans of the franchises took it upon themselves to transcribe songs to create new levels. The producers, seeing the tendency, started releasing the in-app purchases to enable the players to extend their library and thus, keep the users.

Due to the time consuming and difficult nature of the process of manually adding new songs, most players usually limit themselves to pre-processed songs provided by the game producers, not really taking advantage of the full capabilities of the games.

This project aims to change the way users look at the music rhythm games. We are creating a game which will allow them to upload any song they would like and automatically generate a Guitar Hero-like level corresponding to it.

This will be achieved by implementation of a melody extraction from polyphonic music signals algorithm using pitch contour characterisation. The algorithm consists of four parts - sinusoid extraction, salience function, pitch contour creation and melody selection. In this approach, pitch contours - time continuous sequences of pitch candidates, are grouped using auditory streaming cues. To filter them, we define a set of contour characteristics, which help distinguish between melodic and non-melodic contours. This leads to the development of new voicing detection, octave error minimisation and melody selection techniques [3].

We will then design and develop an algorithm for mapping the extracted to a series of buttons on the screen to create an interesting and challenging game for a user, as no literature describing such problem was found so far.

In addition to this, we will attempt to develop a mood extraction algorithm to dynamically generate surroundings in the game. Specifically, we treat music emotion recognition as a regression problem to predict the arousal and valence values (AV values) of each music sample directly, which then can be used to generate unique surroundings for every level generated. This continuous view of music emotion makes the proposed music emotion recognition system free of the inherent ambiguity issue. In addition to this, because there is more freedom in describing a song compared to defining and assigning mood classes, the subjectivity issue is alleviated to some extent. [4].

The music emotion recognition will be achieved by designing and training a neural network to predict listeners' mean valence and arousal ratings associated with musical pieces.

With this project we would also like to show that sophisticated academic music analysis techniques can be combined together and applied to real world problems in an efficient and reliable manner.

Finally the project aims to be more than just a research study of feasibility. The result of successful completion will be an application of sufficient reliability and quality that it can be released to, and used by, untrained computer users. To our knowledge, it is the only computer game allowing people to generate Guitar Hero-like levels that also generates the surroundings tailored to every music track.

Chapter 4

Background

In this section, we investigate different types of music games [5], along with a deeper look into Guitar Hero, on which we base our main concept for the gameplay. This is followed by a discussion of the most applicable publications in music analysis, on finding the main melody in a musical track in particular.

4.1 Music Video Games

A music video game can be defined as a type of game that uses music or rhythm as an integral part of gameplay. This may involve pressing buttons in time with a song, whether on a conventional controller, and instrument controller or some kind of dance mat, singing into a microphone or creating original music. Players can often perform different parts of the same song together in local multiplayer games or over the Internet, providing enjoyable social experiences [7].

Some games exhibit a sandbox style that encourages a free-form gameplay approach whereas other a hybrid style, which combines musical elements with more traditional genres, for example puzzle games or shooters.

Below we will briefly go over different types of music video games that can be found on the market.

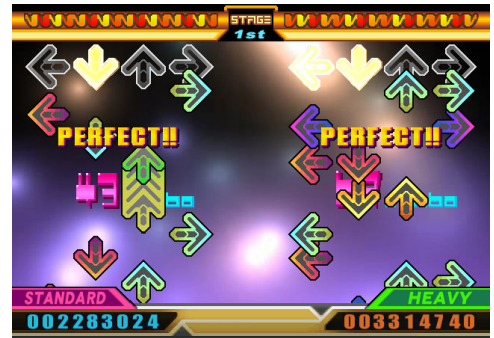


FIGURE 4.1: Screenshot from Dance Dance Revolution, an example of a rhythm music game [6].



(A) is - Internal Section - an example of a generative hybrid music game [8]. (B) SimTunes - an example of a free form music game [9].

FIGURE 4.2: Examples of music video games.

4.1.1 Music Memory Games

The goal of the music memory game is to score a player on their musical memory. Music track is presented to the user who then has to provide an appropriate response to each prompt from the game. Games may be based on different primary musical aspect (whether it is the rhythm, pitch or volume). However, a vast majority of the releases available on the market are rhythm-based.

Rhythm games typically focus on dance or the simulated performance of musical instruments, and require players to press buttons in a sequence dictated on the screen. Doing so causes the game's protagonist or avatar to dance or to play their instrument correctly, which increases the player's score [10]. An example of such games could be Guitar Hero or Dance Dance Revolution.

4.1.2 Hybrid Music Games

Hybrid music games are characterised by substantial and meaningful interactions between a player and the music game in a game that apparently belongs to a non-musical genre. This type of games can be further split into two sub-types.

Generative music video games make use of user's actions. By monitoring interaction with the surroundings in the game, the mechanism generates sounds that are then integrated into the soundtrack, permitting the player's direct interaction with the score. This encourages the creation of a synesthetic experience — when upon stimulation of one sense others activate, causing an involuntary experience. An example of such game could be Rez, which is a simple rail shooter. However, thanks to integrating sounds generated by player completing the normal task of rail-shooting, the musical score is dynamic.



(A) Screenshot from Guitar Hero - player is attempting to play a song [11]. (B) A guitar shaped controller used in the game [12].

FIGURE 4.3: Guitar Hero components

Reactive music games, in contrast to generative one, employ music to determine the gameplay. In such games, the player takes cues from soundtrack to devise his gameplay. For example, *iS* - internal section, uses the music to determine the dynamics of the non-musical components of the game.

4.1.3 Free Form Music Games

In free form music games, the main task of the user is to create content. This form of music game is often compared to non-game music synthesisers. Free form music games are somewhere between generative hybrid music games and non-game utilities, depending on the degree to which their gameplay relies on a driving underlying plot-line. An example of such game could be *SimTunes*, where the user is painting a picture using large pixels and each color represents a musical note.

4.2 Case Study - Guitar Hero

Guitar Hero is one of the most popular franchises in the history of music games. The first of the series was published in 2005 by RedOctane and Harmonix. In the games, players instrument-shaped game controllers to simulate playing the instruments across numerous rock music songs. It is widely considered a highly entertaining game fully embracing the rhythm-based music game.

4.2.1 The Controller

Rather than a typical gamepad, Guitar Hero uses an instrument-shaped controller (guitar in the earlier releases, bass, microphone and drums in more recent ones). Playing the game with the guitar controller simulates playing an actual guitar, except it uses five

coloured "fret buttons" and a "strum bar" instead of frets and strings, and an analogous mapping for the other instruments. They incorporate most of the real life techniques and motions that an instrumentalist would perform on a real instrument.

4.2.2 The Gameplay

The actual game itself works exactly as many other music titles do. At the bottom of the screen, a number of (varying depending of level of difficulty) buttons is shown. In each attempt, a series of notes moves across the screen and when a note aligns with a button, player is supposed to press a corresponding button, gaining points depending on the accuracy. If the player failed to achieve a certain amount of notes — his performance meter stays low for a longer time, he loses the game.

However, there are a couple minor improvements that Harmonix has made to the general music game formula. By pressing buttons with really good accuracy in a song, a player is able to build up Star Power, which when unleashed, doubles up current point multiplier. Star Power also adds a bit of a strategic element - player not only earns more points when it is activated, but he can also raise your performance meter faster, enabling him to last longer when encountering a trickier part of a song.

4.2.3 The Critique

Without a doubt, Guitar Hero features a great selection of music. However, there will always be tracks missing, regardless of how many versions of Guitar Hero are released. People have different tastes and limiting a game to a set of tracks that everybody is supposed to enjoy is a really hard task.

Some more advanced users familiar with Computer Science attempted to transcribe songs and to create new levels. However, this process was really difficult, consisting of many laborious stages and requiring an additional midi files with separated guitar track. This discouraged an average user from fully making use of game's capabilities. The producers, seeing the tendency, started releasing the in-app purchases to enable the players to extend their library and thus, keep the users.

As there is a clear need for custom music extension to the game, implementing a feature of uploading some music preferred by the player would definitely improve user satisfaction. However, this has not been achieved yet as the task itself is quite complex. Moreover, enabling the users to load in some music would deprive the company of their income sources.

4.3 Introduction to Music Analysis

Automatic music analysis is the automated extraction of relevant perceptual information (notes, instruments, etc.) from music files (like mp3s). First attempted in the 1970s at Stanford University [Moorer], it remains an unsolved problem. The problem is highly multifaceted and interdisciplinary, requiring the extraction of musical notes, instruments, percussion, emotion, etc., and drawing from fields as varied as computer science, mathematics, biology, physics, psychology, and electrical engineering. The problem's difficulty lies in a necessity to reverse-engineer the human brain.

For a long time people were researching ways of estimating the fundamental frequency, be it with monophonic music recording or multi-pitch estimation. Melody extraction differs from both of those problems — unlike monophonic pitch estimation it handles polyphonic tracks and in contrast to multi-pitch estimation, it must also include a mechanism for source identification, to spot the voice carrying the melody within the polyphony. To be able to evaluate the performance of the new algorithms, annual Music Information Retrieval Evaluation eXchange (MIREX) has been running since 2005. In this campaign, different models are evaluated against the same sets of music collections in order to obtain a quantitative comparison between methods and assess the accuracy of the current state-of-the-art in melody extraction [13].

4.3.1 Pitch, Tones, Fundamental Frequency

Pitch is the most natural way of ordering sounds on a frequency-related scale. If sounds whose frequency is clear and stable enough to be distinguished from noise, they can be compared among one another as “lower” or “higher”. Pitch is not an objective physical property — it depends on anatomy and physiology of the auditory system, which is a subject of an extensive study called psychoacoustics.

A semitone is the smallest musical interval commonly used in Western tonal music. Two semitones constitute a tone.

The fundamental frequency f_0 is defined as the lowest frequency of a periodic waveform. A harmonic (or a harmonic partial) is any of a set of partials that are whole number multiples of a common fundamental frequency. This set includes f_0 , which is a whole number multiple of itself (1 times itself).

Fundamental frequency can be thought of as the physical property most closely related to perception of pitch. This is why in this context pitch and fundamental frequency can be used interchangeably.

4.3.2 Polyphonic Music

Polyphony is a word derived from Greek *poluphōnōsis* meaning more than one sound — a texture consisting of two or more simultaneous lines of independent melody. This can be contrasted with homophony, where musical parts move generally in the same rhythm and one dominant melodic voice is accompanied by chords or monophony, where only one voice is found.

However, in our case, the term polyphonic will simply refer to any type of music in which two or more notes can be played simultaneously. This can be achieved either by playing in different instruments (for example, voice, guitar and bass) or a single instrument capable of playing more than one more at a time (like a piano).

4.3.3 Melody

The concept of “melody” ultimately relies on the judgment of people listening. This is why it will vary depending on the application context - whether we want to determine symbolic melodic similarity or transcribe a music track.

In order to have a clear framework to work within, the Music Information Retrieval (MIR) community has adopted in recent years the definition proposed by [14], “...the melody is the single (monophonic) pitch sequence that a listener might reproduce if asked to whistle or hum a piece of polyphonic music, and that a listener would recognise as being the ‘essence’ of that music when heard in comparison”.

In practice, research has focused on “single source predominant fundamental frequency estimation” — which means a search for a main melody coming from a single sound source throughout the song analysed. As we can see, the subjective element is still present in this description of a melody as there might not be a definite way of deciding what predominant is. However, it fits well with our project’s objective — generating a game level based on changes in the pitch.

4.3.4 Filter

Any medium through which the music signal passes, whatever its form, can be regarded as a filter. However, we do not usually think of something as a filter unless it can modify the sound in some way.

A digital filter is a filter that operates on digital signals, such as sound represented inside a computer. It is a computation which takes one sequence of numbers (the input signal) and produces a new sequence of numbers (the filtered output signal) [15].

4.3.5 Short Time Fourier Transform

Short-time Fourier transform (STFT), is a signal processing method which is used in analysis of non-stationary signals with statistic characteristics varying with time. In particular, STFT extracts several frames of the signal to be analysed with a window that moves with time. If we set the window size to be narrow enough, each frame extracted can be viewed as stationary so that Fourier transform can be used. With the window moving along the time axis, the relation between the variance of frequency and time can be identified [16].

The short time Fourier transform of a time-domain signal y is denoted by the matrix $F \times N$, F being the Fourier transform size and N the number of analysis frames.

4.4 Main Melody Extraction from Polyphonic Music

In this section we will go over two different approaches to the problem of main melody extraction from polyphonic music, using source separation and a salience function. Then we will compare both methods to determine which one is more suitable for our project.

4.4.1 Source Separation Based Approach

In polyphonic tracks the main melody can be represented by a specific source/filter model. In case of the leading vocal part, the vocal cords are treated as a source and the voice tract as a linear acoustic filter.

In their paper from 2011 [17], authors presented an algorithm in which they assume that at any given time the signal observed is a mixture of two elementary signals -

one corresponding to the main source and one to the background music. Therefore, the signal can be represented in an equation $x(t) = v(t) + m(t)$, where $v(t)$ stands for the source of the main melody and $m(t)$ is the background music. Interestingly, this equation also holds for the short time Fourier transform (STFT) X , V and M respectively: $X = V + M$. The models proposed by Durrieu essentially aim at constraining the shapes of these STFT using temporal and spectral constraints.

The likelihood of the vocal part V is calculated using two different frameworks.

The first submission uses the source/filter Gaussian scaled mixture model (GSMM). In this model the source element refers to the excitation of the vocal folds and is therefore linked to the fundamental frequency of the sound f_0 , while the filter part is characteristic

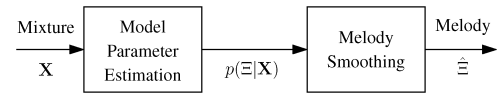


FIGURE 4.4: Outline of system proposed by Durrieu: X is the STFT of the mixture signal, $p(\Xi|X)$ the posterior probability of a given melody sequence, and $\hat{\Xi}$ the desired smooth melody sequence[17].

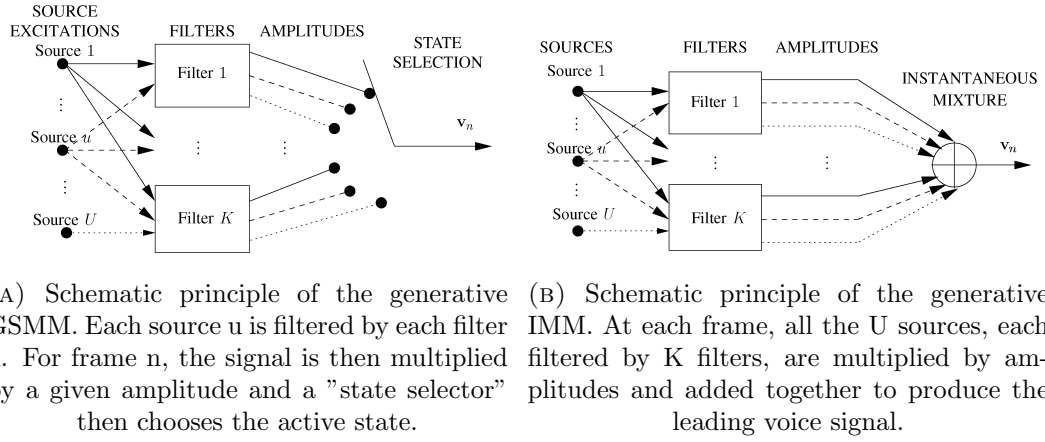


FIGURE 4.5: Diagram of both models presented in the paper[17].

of the vocal tract shape. This space of possibilities is then discretised so that we consider one possible filter frequency response, which is then used to calculate the likelihood of the vocal part knowing the filter and f_0 .

Figure ?? A) shows the diagram of the GSMM model for the main voice part. Each source excitation u is filtered by each filter k . The amplitudes for a frame n and for all the couples (k, u) are then applied to each of the output signals. At last a "state selector" sets the active state for the given frame.

The second model was derived from the first one to find a solution that would be more efficient to compute. The authors came up with a formulation that keep the source/filter model within an instantaneous mixture framework (IMM). In this model, for each source a set of filters is defined and at each frame, once every source is filtered and multiplied by a given amplitude, they are all added together.

The background music signal $m(t)$ can be thought of as a mixture of R independent Gaussian sources $m_r(t)$. Each of the sources is centred and characterised by its power spectral density (PSD), which describes how the power of a signal or time series is distributed over the different frequencies. PSD can be estimated using a Covariance Method. Due to the linearity of the Fourier transform, $M(f, t)$, the STFT of m , is also the instantaneous mixture of the R spectra $M_r(f, t)$ of the sources: $M_r(f, t)$. This together with STFT and an amplitude coefficient associated with each source is used to calculate the likelihood for each of the frequency bins. Let $M_t(f)$ be the STFT of the background signal at frame t and frequency bin f , then we write its likelihood.

Once the parameters are estimated using the maximum likelihood criterion for each of the model, the Viterbi smoothing of the melody line is applied, obtaining a trade-off between the smoothness of the melody and its global energy in the signal. The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states – called the Viterbi path – that results in a sequence of observed events [18].

The authors then parametrise the transitions between the possible main melody without disabling jumps from one note to the other. Using Wiener filtering - digital signal processing reducing the noise, using an statistical estimate of the signal using a desired data without such noise, a framework is implemented to separate the source. This way separated signals are obtained. Computing the energy for each frame of the separated main melody and thereafter thresholding allowed to discriminate between spurious notes and true positives.

4.4.2 Saliency Based Approaches

This approach has been the most popular so far, with majority of algorithms evaluated at MIREX implementing it. It can be split into several smaller stages, as seen in Figure 4.6. In particular, a method implemented in paper [3] seems to be quite promising.

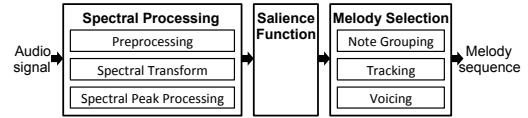


FIGURE 4.6: Block diagram of four main blocks of the system by Salamon and Gómez: sinusoid extraction, saliency function computation, pitch contour creation and melody selection [13].

Usually as a first step, some sort of preprocessing is applied to the audio signal, usually to enhance the frequency content where we expect to find the melody. In particular, Salamon and Gómez apply an equal loudness filter, which enhances the frequencies to which the human ear is more perceptually sensitive, by taking a representative average of the equal loudness curves and filtering the signal by its inverse.

This stage is followed by spectral transform — the signal is chopped into time frames and a transform function is applied to obtain a spectral representation of each frame. This is achieved by applying the Short-Time Fourier Transform given by:

$$X_l(k) = \sum_{n=0}^{M-1} w(n) \times x(n + lH) e^{-j \frac{2\pi}{N} kn} \quad (4.1)$$

with a window length of 46.4ms. Here, $x(n)$ is the time signal, $w(n)$ the windowing function, l the frame number, M the window length, N the FFT length and H the hop size. Thanks to choosing a relatively small hop size, Salamon and Gómez achieve sufficient frequency resolution to identify different notes while maintaining adequate time resolution to track pitch changes in the melody over a short time.

Having done this, we move to frequency/amplitude correction, where the spectral peaks are detected and used to construct a saliency function. To avoid a relatively large error in the estimation of the peak frequency caused by binning them in the process of FFT, peak's instantaneous frequency and amplitude are calculated.

As we can see in Figure 4.6, those three steps constitute the spectral processing. But at the core of the saliency based algorithms lies the multi-pitch representation, i.e. the saliency function — a representation of pitch saliency over time. The peaks of this

function form the f_0 candidates for the main melody. In the algorithm described by Salamon and Gómez, this computation is based on harmonic summation, where the salience of a given frequency is computed as a sum of the weighted energies found at harmonics (integer multiples) of that frequency. Using only the peaks for the summation allows the authors to discard less reliable values and apply further frequency corrections.

The salience function presented in the paper covers a pitch range of nearly five octaves from 55Hz to 1.76kHz.

Peaks of the salience function at each frame are now potential f_0 of the main melody. At this point some methods for melody extraction attempt to track the melody. However, Salamon and Gómez filter out the non-salient peaks, first by comparing them to the highest peak in the frame and then to a value computed using salience mean and standard deviation of all remaining peaks (in all frames). Now the peaks are grouped into pitch contours - time and pitch continuous sequences of salience peaks as shown in Figure 4.7.

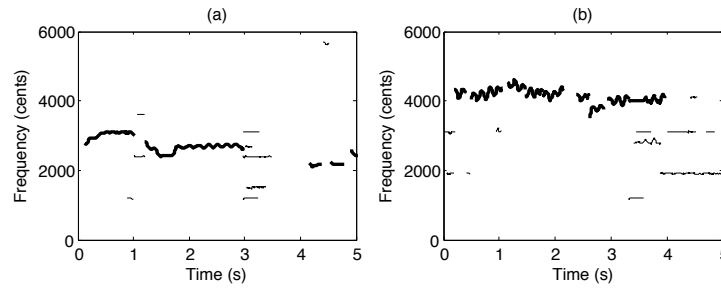


FIGURE 4.7: Pitch contours generated from excerpts of (a) vocal jazz and (b) opera. Melody contours are highlighted in bold[3].

Having created the pitch contours, Salamon and Gómez are faced with the task of determining which one belongs to the main melody. The authors define features based on contour pitch, length and salience.

Given the peaks of the salience function, we now have to determine which pitch values belong to the melody. This process is initiated by grouping peaks into continuous pitch contours, out of which a melody is selected later.

The next main block in this algorithm shown in Figure 4.6 is the melody selection which is comprised of three steps: voicing detection, octave error minimisation/pitch outlier removal, and final melody selection. As the name suggests, the aim of the voicing detection is to determine when the melody is present.

To filter out these contours Salamon and Gómez take advantage of the contour mean salience distribution. By setting the threshold to a value slightly below the average contour mean salience of all contours in the excerpt C_s , we can filter out a considerable amount of non-melody contours. The authors define the following voicing threshold τ_v based on the distribution mean C_s and its standard deviation $\sigma_{\bar{s}}$:

$$\tau_v = C_s - v \times \sigma_{\bar{s}} \quad (4.2)$$

The parameter v determines the lenience of the filtering - a high v value might keep the false melody contours and a low value might filter out the melody contours.

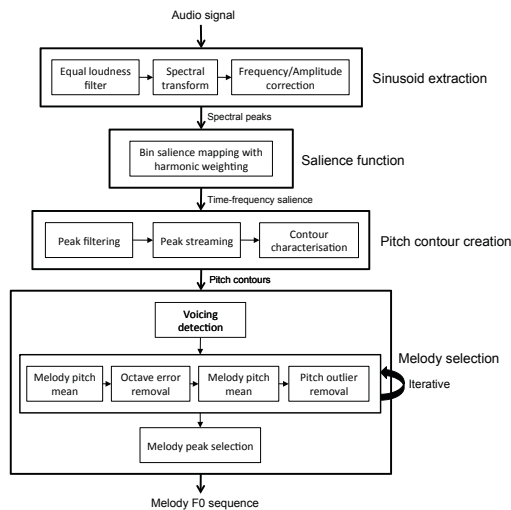


FIGURE 4.8: Block diagram of four main blocks of the system by Salamon and Gómez: sinusoid extraction, salience function computation, pitch contour creation and melody selection [3].

It is also important to note that detecting certain characteristics in the contour increases a probability of it being the melody contour, for example in case of detecting a vibrato - a regular, pulsating change of pitch, used to add expression to vocal and instrumental music. [19]

Next step in the melody selection described by Salamon and Gómez in their paper is octave errors and pitch outliers removal.

In particular, the octave errors are the main sources of errors in melody extraction systems, when a multiple or sub-multiple of f_0 is reported as the main melody.

To detect such errors, contour trajectories are compared by computing distance between their values on a per-frame for the

region they overlap in and computing the mean over this region. If the mean distance is within 1200 ± 50 cents, the contours are considered octave duplicates.

Secondly, Salamon and Gómez use the relationship between neighbouring contours (in time) to decide which of the duplicates is the correct one. Their approach is based on two assumptions: firstly, that most (though not all) of the time the correct contour will have greater salience than its duplicate (the salience function parameters were optimised to this end). Secondly, that melodies tend to have a continuous pitch trajectory avoiding large jumps, in accordance with voice leading principles.

The method iteratively computes the $\overline{P(t)}$ - pitch trajectory that represents the time evolution of the melody’s pitch. It then detects and removes an octave duplicate as well as the "pitch outliers" – contours more than one octave above or below the pitch mean and then it is recalculated. Authors empirically discovered that 2 iterations of this process are enough to get a good approximation of the true trajectory of the melody, which is then passed to the final stage of the model - the final melody selection.

At this stage, there is often only one peak to be chosen as the main melody. When there is still more than one contour present in a frame, the melody is selected as the peak belonging to the contour with the highest total salience $C_{\sum s}$. If no contour is present the frame is regarded as unvoiced.

4.4.3 Comparison of both approaches

In their paper [13], authors attempted to compare multiple melody extraction algorithms created since 2005. One of the methods, used also by MIREX, is based on the per-frame comparison, considering different measures:

Voicing Recall Rate - the proportion of frames labeled as melody frames in the ground truth that are estimated as melody frames by the algorithm.

Voicing False Alarm Rate - the proportion of the frames labeled as non-melody in the ground truth that are mistakenly estimated as melody frames by the algorithm.

Raw Pitch Accuracy - the proportion of melody frames in the ground truth for which f_τ is considered correct (i.e. within half a semitone of the ground truth).

Raw Chroma Accuracy - as raw pitch accuracy, except that both the estimated and ground truth f_0 sequences are mapped onto a single octave. This gives a measure of pitch accuracy which ignores octave errors.

Overall Accuracy - this measure combines the performance of the pitch estimation and voicing detection tasks to give an overall performance score for the system. It is defined as the proportion of all frames correctly estimated by the algorithm, where for non-melody frames this means the algorithm labeled them as non-melody, as for melody frames the algorithm both labeled them as melody frames and provided a correct f_0 estimate for the melody (again, within half a semitone of the ground truth).

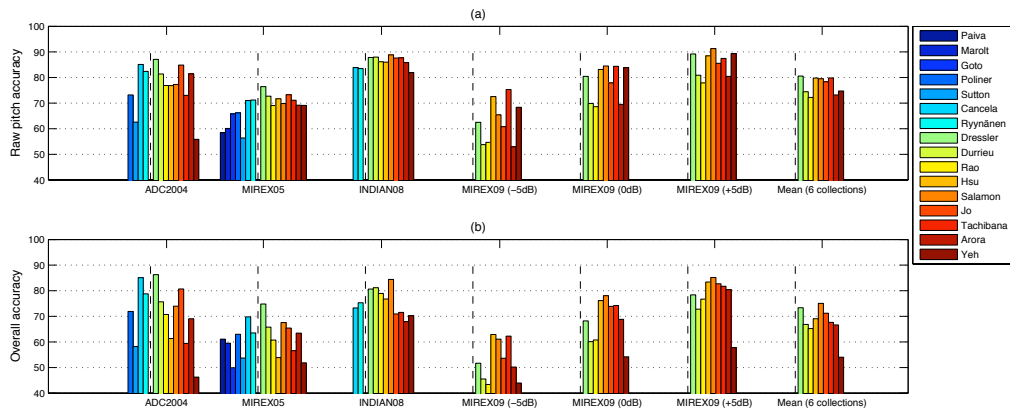


FIGURE 4.9: a) Raw pitch accuracy and b) overall accuracy obtained in MIREX by 16 melody extraction algorithms evaluated in [13]. The vertical dashed line separates the algorithms that were only evaluated on some collections (left of the line) from those evaluated on all six collections (right of the line)[13].

In Figure 2.8. the authors presented results obtained by the algorithms evaluated at MIREX. To get a general idea of the performance of the algorithms, it is sufficient to focus on two evaluation measures. The raw pitch accuracy, presented in Figure 2.8 a) represents how well the algorithm tracks the pitch of the melody. The overall accuracy

on the other hand, as shown in Figure 2.8 b), combines this measure with the efficiency of the algorithm's voicing detection, meaning the voicing-related measures are also reflected in this measure.

As we can see, some collections are generally hard to analyse (for example MIREX09-5db), in general the collections yield different results for different algorithms. This allows us to spot pros and cons of each approach investigated.

We can also notice that the raw pitch accuracy gradually improved from 2005 to 2009, after which it stayed relatively unchanged. Overall we can see that the average pitch accuracy over a collection lies between 70-80

On the other hand, when it comes to overall accuracy, the performance goes down compared to the raw pitch accuracy for all algorithms due to voicing detection being factored into the results. The importance of this step depends on the intended use of the algorithm. Generally, the overall accuracy results lie between 65-70

Finally, an important factor in assessment of an algorithm is its complexity. While deriving O-notation is too complex for some of the algorithms, generally it is observed that algorithms involving source separation are significantly more computationally complex than salience based approaches. Unfortunately, there is no specific data provided by Salamon and Gómez [9] or by Durrieu [5] on their algorithms.

In conclusion, we believe the solution proposed by Salamon and Gómez is better fitted to the purpose of this project. The paper presents it in a much clearer way and, what is most important, it outperforms the one created by Durrieu significantly, as seen in Figure 2.8.. In addition to this, according to tendency it is less computationally expensive, which is quite important when it comes to game designing as we do not want to keep the user waiting for a long time for his level to generate and load.

4.5 Introduction to Neural Networks

An Artificial Neural Network (ANN) is an information processing paradigm that can be thought of humans' attempt to simulate the brain electronically. Its first conceptual model was developed by Warren S. McCulloch, a neuroscientist, and Walter Pitts, a logician, in 1943. In their paper, "A logical calculus of the ideas imminent in nervous activity," they describe the concept of a neuron, a single cell living in a network of cells that receives inputs, processes those inputs, and generates an output. Their work served as foundation for designing a computational model based on the brain to solve certain kinds of problems.

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. Their most common

application in computing today is to perform one of these “easy-for-a-human, difficult-for-a-machine” tasks, often referred to as pattern recognition. They have higher computational capabilities than any human being - calculating a cube of 9124 in memory is not straightforward for us, but a computer can come up with an answer almost immediately, and thanks to their structure they can tackle problems not easy to solve by a simple computer, like facial recognition or regression analysis. A trained neural network can be thought of as an “expert” in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest and answer “what if” questions.

4.5.1 Models

The computational systems we write are procedural; a program starts at the first line of code, executes it, and goes on to the next, following instructions in a linear fashion. On the other hand, neural networks are “connectionist” computational systems. A true neural network does not follow a linear path. Rather, information is processed collectively, in parallel throughout a network of neurons.

Neural networks are made up of many artificial neurons. There are many different ways of connecting neurons to create a neural network. The number of them depends on a task the network is designed for. .

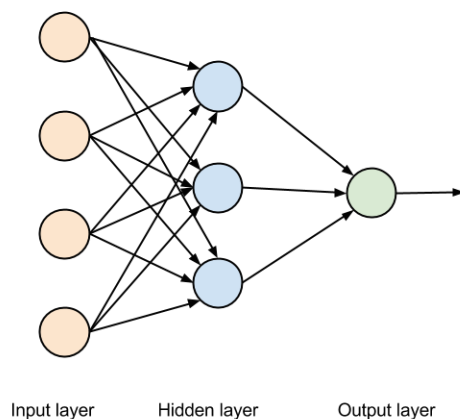


FIGURE 4.10: Diagram of a simple neural network with 4 input nodes, 3 nodes in a hidden layer and one output node.

An example system has three layers. The first layer has input neurons which send data via synapses to the second layer of neurons. Each input into the neuron has its own weight associated with it. A weight is simply a floating point number and they allow us to adjust our network to improve the training outcome. The weights in most neural nets can be both negative and positive, therefore providing excitatory (carrying information) or inhibitory (regulating the activation of excitatory neurons) influences to each input.

As each input enters the nucleus, it is multiplied by its weight. The nucleus then sums all these new input values which gives us the activation. If the activation

is greater than a threshold value, the neuron outputs a signal. If the activation is less than the threshold, the neuron outputs zero. This is typically called a step function.

One type of neural network is called a feedforward network named after the way the neurons in each layer feed their output forward to the next layer until we get the final output from the neural network.

Each input is sent to every neuron in the hidden layer and then each hidden layer's neuron's output is connected to every neuron in the next layer. There can be any number of hidden layers within a feedforward network but one is usually enough to suffice for most problems you will tackle. Also the number of neurons I've chosen for the above diagram was completely arbitrary. There can be any number of neurons in each layer, it all depends on the problem. By now you may be feeling a little dazed by all this information so I think the best thing I can do at this point would be to give you a real world example of how a neural net can be used in the hope that I can get your very own brain's neurons firing!

4.6 Mood Detection

It is well known that music can convey emotion and modulate mood. That is why the relation between musical sounds and their influence on the listener's emotion has been well studied.

One of the first publications on emotion detection in music is credited to Feng, Zhuang, and Pan [20]. They employ Computational Media Aesthetics to detect mood for music information retrieval tasks. The two dimensions of tempo and articulation are extracted from the audio signal and are mapped to one of four emotional categories; happiness, sadness, anger, and fear. After that, feature called relative tempo is calculated, and after the mean and standard deviation of the feature called average silence ratio in the presented computational articulation model are calculated, a simple Back Propagation neural network classifier is trained to detect mood.

In publication by Yang, Lin, Su and Chen [4], the authors presented a tool which recognises a mood in a musical track, allowing a user to then choose the song they want to play by deciding on emotions it is supposed to represent. Specifically, the authors formulate music emotion recognition as a regression problem to predict the arousal and valence values (AV values) of each music sample directly.

Potentially, the second approach described seems more appropriate for our project as it allows for better granularity in the melody emotion detection and, hence, wider variety of changes in the game's environment. However, this area of the project is left to be further researched.

4.7 Level Generation

It is not really surprising that there is no current literature on the problem of automatically generating Guitar Hero buttons given an arbitrary piece of music. However, we believe an algorithm can be developed where the buttons can be mapped to the f_0 in the main melody extracted by main melody extraction algorithm.

Chapter 5

Implementation

In this chapter we will go over the implementation process of the project. We will describe various choices we made, justifying them in the context of our objectives. Mood detection, melody extraction and the game architecture will be our main focus.

5.1 Mood Detection

A common reason for engaging in music listening is that music is an effective means of conveying and evoking emotions. Although they may be subjective, based in part on the listener’s cultural and musical background or preferences, there are commonalities in perceived emotion across different listeners based on the characteristics of the music. Several studies have attempted to predict emotion conveyed during music listening. Some have explored the relationship between physiological activity experienced by a listener and perceived emotion, while others focused on the relationship between perceived emotion and the musical/acoustic features themselves. In our approach, we adapted the latter option, representing emotion using a two-dimensional space with valence on the x-axis and arousal on the y-axis.

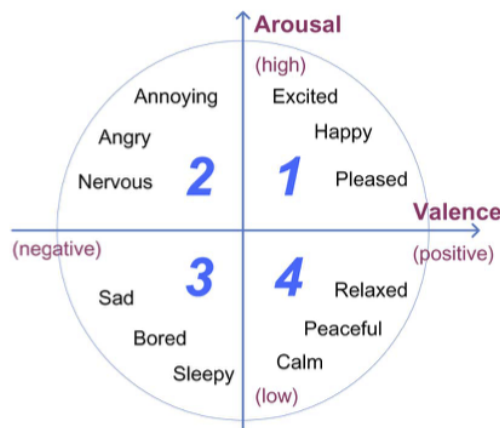


FIGURE 5.1: Thayer’s arousal-valence emotion plane.[4].

In our exploration we decided to base our research on data collected by [21], to avoid personal bias in assessing the mood of the song. The songs in the dataset were annotated by more than 300 crowdworkers on Amazon Mechanical Turk. Each song was annotated for arousal and for valence separately.

Diagram 5.1 depicts the relation between valence and arousal values for a musical track and the moods perceived by people. As we can see, the high arousal is connected to how energetic the music is, whereas valence refers to how positive (or negative) the emotions in the track are.

5.1.1 Choice of Features

Using Essentia library [22], we implemented an extractor to retrieve certain features from a song, which we would expect to have certain impact on the perceived mood of a musical piece:

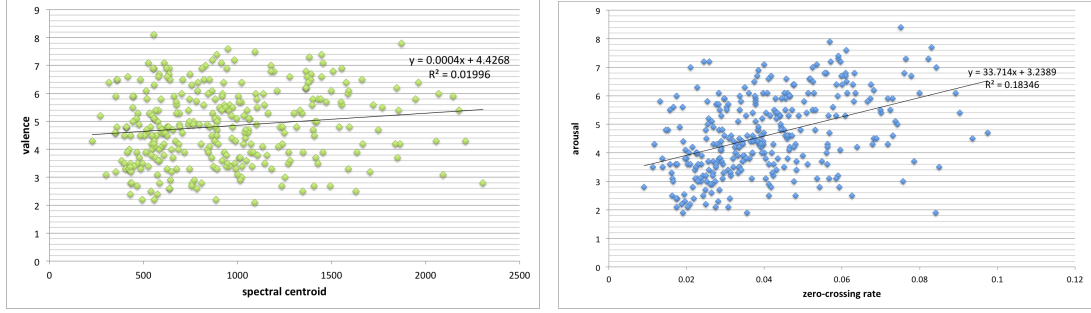
- average loudness - dynamic range descriptor. It rescales average loudness, computed on 2sec windows with 1 sec overlap, into the [0,1] interval. The value of 0 corresponds to signals with large dynamic range, 1 corresponds to signal with little dynamic range. This could indicate the level of the valence, with higher loudness implying higher valence value.
- means and derivatives of variance of rates of silent frames in a signal for thresholds of 20, 30 and 60db. We believe that the values could influence the arousal levels, as the more and the bigger the silent gaps, the sadder / more peaceful the track seems to be, implying the low arousal value.
- dynamic complexity - dynamic complexity computed on 2sec windows with 1sec overlap. The dynamic complexity is the average absolute deviation from the global loudness level estimate on the dB scale. It is related to the dynamic range and to the amount of fluctuation in loudness present in a recording. We believe this feature would have an impact on the arousal level.
- BMP - beats per minute value according to detected beats. This feature should be correlated with the arousal level - the faster the song, the more energetic it seems.
- spectral centroid - centroid statistics describing the spectral shape. It indicates where the "center of mass" of the spectrum is. Perceptually, it has a robust connection with the impression of "brightness" of a sound - an indication of the amount of high-frequency content in a sound. We believe the spectral centroid might be related to both valence and arousal.
- spectral RMS (root mean square). In physics it is a value characteristic of a continuously varying quantity, such as a cyclically alternating electric current, obtained by taking the mean of the squares of the instantaneous values during a cycle. This is the effective value in the sense of the value of the direct current that would produce the same power dissipation in a resistive load. An electric current of given magnitude produces the same heating regardless of the direction of current flow; squaring the quantity measured ensures that alternation of sign does not invalidate the result.

- spectral energy - the energy E_s of a continuous-time signal $x(t)$ defined as:
 $E_s = \langle x(t), x(t) \rangle = \int_{-\infty}^{\infty} |x(t)|^2 dt$ Signal energy is always equal to the summation across all frequency components of the signal's spectral energy density. It could have an impact on both valence and arousal.
- mean and derivative of variance of beat loudness - spectral energy computed on beats segments of audio across the whole spectrum, and ratios of energy in 6 frequency bands. We suspect that the low value of the beat loudness could imply a low arousal.
- scale and key of the key estimated key using Temperley's profile
- scale and key of the chords key taken as the most frequent chord, and scale of the progression, whether major or minor. This and the key features could have an impact on the valence value.
- means of zero-crossing rate - the rate of sign-changes along a signal, i.e., the rate at which the signal changes from positive to negative or back. This feature has been used heavily in music information retrieval, being a key feature to classify percussive sounds. We believe it could be related to the arousal value. ZCR is defined formally as:

$$ZCR = \frac{1}{T-1} \sum_{t=1}^{T-1} \mathbb{I}\{s_t s_{t-1} < 0\}$$
- pitch salience of a spectrum - given by the ratio of the highest auto correlation value of the spectrum to the non-shifted auto correlation value. Pitch salience was designed as quick measure of tone sensation. Unpitched sounds (non-musical sound effects) and pure tones have an average pitch salience value close to 0 whereas sounds containing several harmonics in the spectrum tend to have a higher value. Application: characterizing percussive sounds. We think the value could have an effect on both the valence and arousal.
- mean and derivative of variance of sensory dissonance (to distinguish from musical or theoretical dissonance) of an audio signal given its spectral peaks. Sensory dissonance measures perceptual roughness of the sound and is based on the roughness of its spectral peaks. Given the spectral peaks, the algorithm estimates total dissonance by summing up the normalized dissonance values for each pair of peaks. These values are computed using dissonance curves, which define dissonance between two spectral peaks according to their frequency and amplitude relations. Dissonance could be related to negative valence.

5.1.2 Correlation Between Features and Mood Perception

As a first step towards understanding the pattern by which audio features might account for emotion ratings, we conducted correlational analyses between features and mean valence/arousal ratings from the data set. We performed a bivariate correlation analysis with the valence/arousal ratings as the dependent variable, and each of the 22 features



(A) A graph representing a correlation between spectral centroid and valence values. (B) A graph representing a correlation between zero-crossing rate and arousal values.

FIGURE 5.2: Chosen results of bivariate correlation with multiple regression.

as the explanatory variable. Example of the results we achieved can be seen in Figure 5.2. We found significant correlation between *valence* and *dvar* and *mean silence60*, *dvar of silence30*, *dynamic complexity*, *spectral centroid*, *spectral RMS*, *spectral energy*, *zero-crossing rate*, *pitch salience*, and both *mean* and *dvar* of *dissonance*. For *arousal*, we noticed correlation with *spectral centroid*, *pitch salience*, *zero-crossing rate*, both *mean* and *dvar* of *silence60*, *spectral energy*, *mean dissonance* and *dynamic complexity*.

Values of all the features were then normalized between 0 and 1 to prepare them for the neural network training.

5.1.3 Neural Network for Mood Prediction

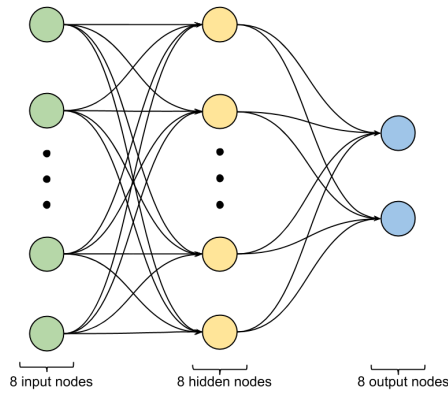


FIGURE 5.3: A diagram depicting the structure of our artificial neural network for mood detection.

Our goal was to train the network to predict mean participant valence and arousal values for musical excerpts. Our first network implementation was a supervised, feedforward network with backpropagation. The input consisted of normalized values of 8 features: *spectral centroid*, *pitch salience*, *zero-crossing rate*, *silence60 mean and dvar*, *mean dissonance*, *dynamic complexity* and *spectral energy*. The network had two outputs - arousal and valence.

As all the training data was normalised, the input and output values were within a range of 0 to 1. The training set consisted of 50 input and output arrays. Each input array had 8 values, one per audio feature, and its corresponding output array had the two desired arousal and valence values.

No. of Nodes	RMSE
1	0.0885458989882
2	0.0881453943244
3	0.0873850620651
4	0.086553779403
5	0.0862146793784
6	0.0860987534316
7	0.0861442944822
8	0.0850460538049
9	0.085143906777
10	0.0852984563213
15	0.085547018396
20	0.0854215840659
50	0.0856472028298

TABLE 5.1: Table showing the root mean square error for training the network for given number of nodes in the hidden layer.

The network's task was to provide the valence and arousal values based on the 13 audio features. The output values fell within a range of 0 to 1. Since desired outputs were average valence/arousal ratings provided by participants on a scale from 1 to 9, the network outputs were rescaled back. The training set consisted of eight input and output arrays. Each input array had 13 values, one for each audio feature, and its corresponding output array had the two desired arousal and valence values. The connection weights from input to the hidden nodes and from hidden nodes to the output ones were initialised to random numbers.

The network was built, trained, and tested using the pyBrain python library for neural network implementation.

We trained our network for 10000 epochs with many different sizes of the hidden layer and default values for all the other parameters. The performance based on that can be seen in Table 6.1.

As we can see, the optimal solution is the one with 8 nodes in the hidden layer. To avoid overfitting the network, we kept the number of hidden units equal to the number of input units.

Having found the optimal number of nodes in the hidden layer, we moved on to find the learning rate parameter. We started our search by setting it to 0.3 and reducing it over time. The results we found can be found in Table 5.2.

In the end, we came up with the network which can be seen on Figure 5.3.

Learning Rate	RMSE
0.3	0.0846361666437
0.25	0.0827496336912
0.2	0.080513130655
0.15	0.0807476303566
0.1	0.0792817515366
0.05	0.0805650982375
0.01	0.085863454125

TABLE 5.2: Table showing the root mean square error for training the network for given learning rate parameter value.

5.2 Main Melody Extraction

5.3 The Game

In this section, we will go over the architecture and the design choices made when planning and implementing our game.

The game is written mostly in Swift, a multi-paradigm, compiled programming language created by Apple Inc. for iOS and OS X development. It was first introduced at Apple's 2014 Worldwide Developers Conference (WWDC). Swift is designed to work with Apple's Cocoa and Cocoa Touch frameworks, building on the best of C and Objective-C, without the constraints of C compatibility. It adopts safe programming patterns and adds modern features to make programming easier, more flexible, and more fun [23].

We chose this language as we wanted to create a game for the OS X platform. In addition to this, the author also had a personal interest in learning the language.

5.3.1 Data Storage

The game relies on preserving user's scores and the levels generated by them. We need a way of storing them and all the information retrieved when analysing the songs to avoid regenerating the levels for the same song, for example if the user has a music piece they particularly like.

Core Data is the standard way to persist and manage data in both iPhone and Mac applications. It is an object graph and persistence framework provided by Apple in the Mac OS X and iOS operating systems.

Core Data describes data with a high level data model expressed in terms of entities and their relationships plus fetch requests that retrieve entities meeting specific criteria. Code can retrieve and manipulate this data on a purely object level without having to worry about the details of storage and retrieval.

Core Data allows data organised by the relational entity–attribute model to be serialised into XML, binary, or SQLite stores.

Core Data is also a persistent technology, in that it can persist the state of the model objects to disk. But the important takeaway is that Core Data is much more than just a framework to load and save data - it is also about working with the data while it is in memory. We decided to use Core Data rather than a separate database as our game only needs to store data used by the current user, that will be utilised almost immediately after loading into memory.

The model might cause some intensive memory usage if we decide to create a big amount of users, however, as it is an offline game that can be played on a personal machine, in contrast to web application, the number of users should remain relatively small.

5.3.2 Menu

Although not usually adopted in OS X games, we decided to follow the Model-View-Controller design pattern in implementing our application. We believe it was a right choice as the complexity of the main menu would have to be then supported throughout the played level. This would not only be a performance strain, but would also cause the code to be messy.

When first facing the menu, the user has an option of creating an account, logging in as a user or playing a quick game, not requiring any user data. The quick game is essentially an ability of playing one of the predefined levels, without a choice of creating a new one.

Once the user has created an account or chosen an existing one, they can either follow the level creation or level loading option. If they choose to create a new level, they have to select a file from their hard drive they would like to use as the base for their level. Otherwise, they go to the window, where they can select a level and either play it or remove it from their catalogue.

5.3.3 Level Description

Once we move on to playing a game, the `GameViewController` unpacks the `GameScene` - an object representing a scene of content in Sprite Kit.

Sprite Kit provides a graphics rendering and animation infrastructure that can be used to animate arbitrary textured images, or sprites. Sprite Kit uses a traditional rendering loop where the contents of each frame are processed before the frame is rendered. Sprite Kit does the work to render frames of animation efficiently using the graphics hardware. Sprite Kit is optimized so that the positions of sprites can be changed arbitrarily in each frame of animation.

Sprite Kit also provides other functionality that is useful for games, including basic sound playback support and physics simulation. In addition, Xcode provides built-in support for Sprite Kit so that you can create complex special effects and texture atlases directly in Xcode. This combination of framework and tools makes Sprite Kit a good choice for games and other apps that require similar kinds of animation [24].

In the game scene, there is a set of buttons at the bottom of the screen. Players use the strum bar along with the fret buttons to play notes that scroll down the screen. The Easy difficulty only uses the first three fret buttons, that is, the green, red, and yellow. The Medium difficulty uses the blue button in addition to those three, and Hard and Expert use all five buttons.

The score is calculated based on how many scrolling notes we manage to hit. Every time we hit, the performance bar on the right side of the screen goes up, otherwise it goes down. If it hits the minimum, the player loses. However, if the player manages to keep the performance level at the maximum for an appropriate amount of time, the number of the points scored for the new notes gets doubled until he misses a note or wins the level.

The player can at any time pause, stop or replay the game.

5.3.4 Melody Detection as a Game Changer

5.3.5 Impact of the Mood on the Level

5.4 Main Section 2

Sed ullamcorper quam eu nisl interdum at interdum enim egestas. Aliquam placerat justo sed lectus lobortis ut porta nisl porttitor. Vestibulum mi dolor, lacinia molestie gravida at, tempus vitae ligula. Donec eget quam sapien, in viverra eros. Donec pellentesque justo a massa fringilla non vestibulum metus vestibulum. Vestibulum in orci quis felis tempor lacinia. Vivamus ornare ultrices facilisis. Ut hendrerit volutpat vulputate. Morbi condimentum venenatis augue, id porta ipsum vulputate in. Curabitur luctus tempus justo. Vestibulum risus lectus, adipiscing nec condimentum quis, condimentum nec nisl. Aliquam dictum sagittis velit sed iaculis. Morbi tristique augue sit amet nulla pulvinar id facilisis ligula mollis. Nam elit libero, tincidunt ut aliquam at, molestie in quam. Aenean rhoncus vehicula hendrerit.

Chapter 6

Evaluation

Game developers use personal preferences and creative programming techniques and tools to develop games with the hopes of successful market penetration. Often, in the course of development, the needs of the end user are lost.

Evaluation can occur during various times during the design and development life cycle of a game – early, in the middle, late, and at the end. However, not all types of evaluation methods can be applied during all phases of design and development.

6.1 Formative

Formative evaluation is any evaluation that takes place before or during a project's implementation with the aim of improving its design and performance. It is essential for trying to understand why a program works or does not, and what other factors (internal and external) are at work during a project's life. Formative evaluation does require time but it significantly improves the likelihood of achieving a successful outcome through better program design

6.1.1 Single-Condition Study

Throughout the course of the design and development of the game we will be conducting studies by asking small groups of people to play our game. The main aims of this type of study is to learn about the opinion the game causes and to observe reactions of the players while they are testing it. This helps avoiding people being biased. We would like to observe the pace at which they learn the rules without being instructed in person, telling us whether the user interface is intuitive, if they find the game challenging, which would be visible in their scores and their engagement (do they try different songs over and over again or do they get bored after 10 minutes of playing?).

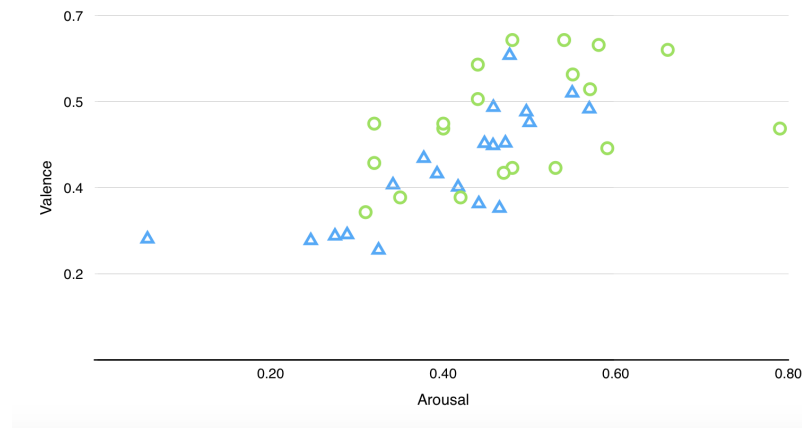


FIGURE 6.1: A plot of the expected and predicted .

6.2 Summative

Summative evaluation looks at the impact of an intervention on the target group. It is outcome-focused more than process focused. Typically, the findings are used to help decide whether a program should be adopted, continued, or modified for improvement

6.2.1 Evaluation of Mood Detection system

We wanted to use neural networks to calculate valence and arousal ratings of songs using audio features we extract.

We computed the mean error between participant ratings and network-predicted outputs across all segments of all test melodies. The network's performance total RMSE was 0.148004365696 on scale from 0 to 1 or 36.4%. The network predicted at an average accuracy of 63.6% for all 20 segments. The plot of the expected and predicted values can be seen in Figure 6.1.

Results from the static network indicate that a network can be trained to identify statistical consistencies across audio features abstracted from music and satisfactorily predict valence/arousal values that closely match mean participant ratings.

6.2.2 Comparison to Original Songs

In order to evaluate the quality of the gameplay generated by our program, we will test our game with songs already existing in the original Guitar Hero game and compare the output we get with its already defined levels. However, to make that possible, the music track we feed to our program must be an instrumental version of the same song as Guitar Hero's songs are mapped onto the key presses by looking at the guitar line of the song, not the main melody. We can then create statistics of correctly identified, false alarm and missed buttons.

expected arousal	expected valence	predicted arousal	predicted valence
0.31	0.3	0.444431	0.317472
0.35	0.33	0.328461	0.223183
0.57	0.55	0.460908	0.435624
0.44	0.53	0.380779	0.409680
0.58	0.64	0.480053	0.619061
0.32	0.48	0.061191	0.245715
0.4	0.47	0.278070	0.251782
0.55	0.58	0.461282	0.513407
0.32	0.4	0.250095	0.242515
0.44	0.6	0.451006	0.439964
0.48	0.39	0.503161	0.482461
0.59	0.43	0.572258	0.510141
0.54	0.65	0.345153	0.355842
0.66	0.63	0.499413	0.504208
0.4	0.48	0.475201	0.440776
0.53	0.39	0.552490	0.542544
0.79	0.47	0.468336	0.308246
0.47	0.38	0.420469	0.351254
0.42	0.33	0.396124	0.378217
0.48	0.65	0.292138	0.254419

TABLE 6.1: Table showing the root mean square error for training the network for given number of nodes in the hidden layer.

6.2.3 Melody Extraction Testing

To evaluate our implementation of the melody extraction algorithms we can use the technique used at Music Information Evaluation eXchange, described in section 2.4.3 of the report. In particular, we can compare the performance of our implementation when tested on the samples used during MIREX to the official statistics presented in papers [3, 13].

Another way of evaluating the game is creating a set of songs and generating levels for them. After that a trained Guitar Hero player can play those levels. If the buttons were consistently on time with the notes then the melody extraction and game synchronisation techniques are considered to work.

6.2.4 Questionnaires

Questionnaires are one of the most common and popular tools to gather data from a large number of people. They generally consist of a limited number of questions that ask participants to rate the effectiveness of various aspects of the activity. The questions should focus on the key points we are trying to evaluate.

Questionnaires tend to be short in order to reduce the amount of time respondents need to complete them, and therefore increase the response rate.

We plan on writing questions that are quantitative and generally consist of close-ended questions (tick the box, or scales), as the open ended questions tend to make data analysis and reporting more difficult.

Bibliography

- [1] Ian Bogost. *How to Do Things with Videogames*. University of Minnesota Press, 2011.
- [2] Guitar hero sales article. URL <http://arstechnica.com/gaming/2009/01/guitar-hero-iii-first-game-to-reach-1-billion-in-sales/>.
- [3] J. Salamon and E. Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Trans. on Audio, Speech and Language Processing*, 20(6), 2012.
- [4] Yi-Hsuan Yang, Yu-Ching Lin, Ya-Fan Su, and Homer H. Chen. A regression approach to music emotion recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(2):448–457, 2008.
- [5] Music video game types. URL http://en.wikipedia.org/wiki/Music_video_game.
- [6] Dance dance revolution screenshot. URL http://cdn-static.gamekult.com/gamekult-com/images/photos/00/00/34/99/ME0000349967_2.jpg.
- [7] Music video game definition. URL <http://psp.about.com/od/pspglossary/a/Music-Game-Definition.html>.
- [8] Internal section screenshot. URL <http://www.hardcoregaming101.net/internalsection/is2.jpg>.
- [9] Simtunes screenshot. URL <http://i.ytimg.com/vi/9CLYAL0930k/hqdefault.jpg>.
- [10] Rhythm game definition. URL http://en.wikipedia.org/wiki/Rhythm_game.
- [11] Guitar hero screenshot. URL <https://images-na.ssl-images-amazon.com/images/G/01/videogames/detail-page/gh3.lor.03.lg.jpg>.
- [12] Guitar hero controller photo. URL http://www.pixelplayer.se/bilder/texter/rec_8576_1_20081117.jpeg.
- [13] J. Salamon, E. Gómez, D. P. W. Ellis, and G. Richard. Melody transcription from music audio: Approaches and evaluation. *IEEE Signal Processing Magazine*, 31(2): 118–134, 2014.

- [14] G. E. Poliner, D. P. W. Ellis, F. Ehmann, E. Gómez, S. Streich, and B. Ong. Melody transcription from music audio: Approaches and evaluation. *IEEE Trans. on Audio, Speech and Language Process*, 15(4):564– 575, 2007.
- [15] Julius O. Smith. *Introduction to Digital Filters with Audio Applications*. W3K Publishing, 2007. URL <http://books.w3k.org/>.
- [16] Short time fourier transform. URL <http://www.originlab.com/index.aspx?go=Products/Origin/DataAnalysis/SignalProcessing/STFT>.
- [17] J.-L. Durrieu, G. Richard, B. David, and C. Fevotte. Source/filter model for unsupervised main melody extraction from polyphonic audio signals. *IEEE Trans. on Audio, Speech and Language Process*, 18(3):564– 57, 2010.
- [18] Viterbi algorithm. URL http://en.wikipedia.org/wiki/Viterbi_algorithm.
- [19] Vibrato definition. URL <http://en.wikipedia.org/wiki/Vibrato>.
- [20] Yazhong Feng, Yueting Zhuang, and Yunhe Pan. Music information retrieval by detecting mood via computational media aesthetics. *Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on*, pages 235– 241, 2003.
- [21] Soleymani, Mohammad, Caro, Micheal N., Schmidt, Erik M., Sha, Cheng-Ya, and Yi-Hsuan Yang. 1000 songs for emotional analysis of music. In *Proceedings of the 2Nd ACM International Workshop on Crowdsourcing for Multimedia*, CrowdMM '13, pages 1–6. ACM, 2013. ISBN 978-1-4503-2396-3. doi: 10.1145/2506364.2506365. URL <http://doi.acm.org/10.1145/2506364.2506365>.
- [22] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, and et al. Essentia: an audio analysis library for music information retrieval. *International Society for Music Information Retrieval Conference (ISMIR 13)*, pages 493–498, 2013.
- [23] Swift documentation. URL https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/index.html#//apple_ref/doc/uid/TP40014097-CH3-ID0.
- [24] Sprite kit documentation. URL https://developer.apple.com/library/ios/documentation/GraphicsAnimation/Conceptual/SpriteKit_PG/Introduction/Introduction.html.