Imperial College London

Interim Report

Turning Music Into Game

Supervisor:
Dr. Iain Phillips

Author:
Paulina Koch

2nd Marker:
Dr. Robert Chatley

January 2015

Contents

Chapter 1

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. [3]

However, a limited amount of songs transcribed and adjusted to the gameplay 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. We will try to achieve that by implementing an algorithm that uses the pitch contour characteristics to detect a main melody in a music track and designing and developing an algorithm to map said melody to a series of keyboard presses.

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.

Chapter 2

Background

In this section, we go investigate different types of music games, 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.

2.1 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. [2]

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.

2.1.1 Music Memory Games

The goal of the music memory game is to score 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.

FIGURE 2.1: Screenshot from Dance Dance Revolution, an example of a rhythm music game. [11]

2.1.2 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.

2.1.3 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 score is dynamic.

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 dynamic of the non-musical components of the game.

(A) Screenshot from Guitar Hero - player is attempting to play a song. [8]

(B) A guitar shaped controller used in the game. [7]

FIGURE 2.2: Guitar Hero components

2.2 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.

2.2.1 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.

2.2.2 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 acing certain notes 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.

2.2.3 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 is really difficult, especially for non computer scientists, discouraging 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.

Implementing a feature of uploading some music preferred by the player would definitely improve user satisfaction. However, this might have 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.

2.3 Introduction to Melody Extraction

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. [14]

2.3.1 Melody

The concept of "melody" ultimately relies on the judgement 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 [4], "...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.

2.3.2 Polyphonic Music

Polyphony is a word derived from Greek poluphōnosis 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).

2.3.3 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.

2.3.4 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]

2.3.5 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. [6]

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.

2.4 Main Melody Extraction from Polyphonic Music

2.4.1 Source Separation Based Approach

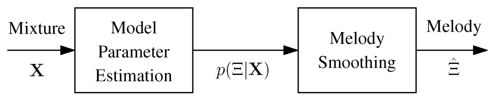
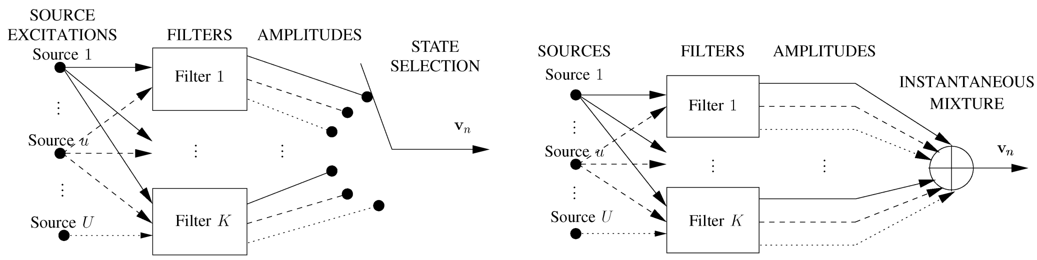


Figure 2.3: Outline of system proposed by Durrieu: X is the STFT of the mixture sygnal, $p(\Xi|X)$ the posterior probability of a given melody sequence, and $\hat{\Xi}$ the desired smooth melody sequence.

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 [5], 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.



(A) Schematic principle of the generative GSMM. Each source u is filtered by each filter k. For frame n, the signal is then multiplied by a given amplitude and a "state selector" then chooses the active state.

(B) Schematic principle of the generative IMM. At each frame, all the U sources, each filtered by K filters, are multiplied by amplitudes and added together to produce the leading voice signal.

Figure 2.4: Diagram of both models presented in the paper.

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 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.

Fig 2.4. 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. [12]

The authors then parametrise the transitions between the possible main melody without disabling jumps from one note to the other.Usign 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.

2.4.2 Salience Based Approaches

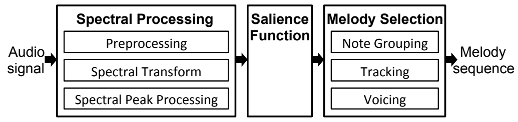


Figure 2.5: Block diagram of four main blocks of the system by Salamon and Gómez: sinusoid extraction, salience function computation, pitch contour creaton and melody selection

This approach has been the most popular so far, with majority of algorithms evaluated at MIREX implementing it and can be split into several smaller stages. In particular, a method implemented in paper [9] seems to be quite promising.

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:

$$X_l(k) = \sum_{n=0}^{M-1} w(n) \times x(n + lH) e^{-j \frac{2\pi}{N} kn} \tag{2.1}$$

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 we 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 salience 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.

Those three steps constitute the spectral processing. But at the core of the salience based algorithms lies the multi pitch representation, i.e. the salience function — a representation of pitch salience 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 us to discard less reliable values and apply further frequency corrections.

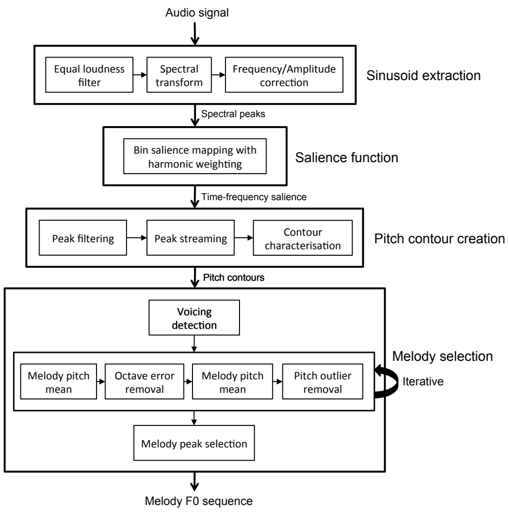


FIGURE 2.6: Block diagram of four main blocks of the system by Salamon and Gómez: sinusoid extraction, salience function computation, pitch contour creaton and melody selection

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 fFgure 2.7.

Having created the pitch contours, we 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 consideration of various contour characteristics means accompanying instruments will not necessarily be selected as melody if they exhibit a certain melodic characteristic. For example, a contour produced by an accompanying violin with vibrato may still be discarded due to its pitch height.

The next main block in this algorithm 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 we take advantage of the contour mean salience distribution. By setting the threshold to a value slightly below the average contour mean salience of

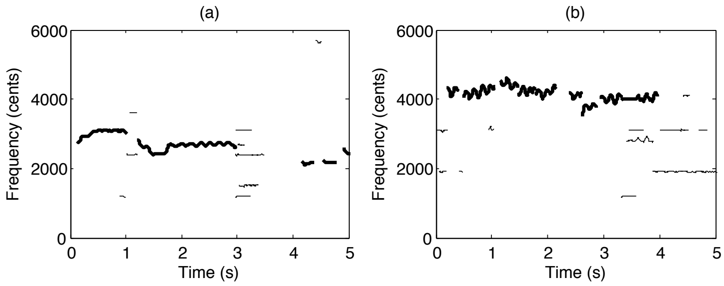


FIGURE 2.7: Pitch contours generated from excerpts of (a) vocal jazz and (b) opera. Melody contours are highlighted in bold.

all contours in the excerpt C_s, we can filter out a considerable amount of non-melody contours. We define the following voicing threshold τ_v based on the distribution mean C_s and its standard deviation $\sigma_{\overline{s}}$:

$$\tau_v = C_s - v \times \sigma_{\overline{s}}$$

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.

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. [13]

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, we use the relationship between neighbouring contours (in time) to decide which of the duplicates is the correct one. Our 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.

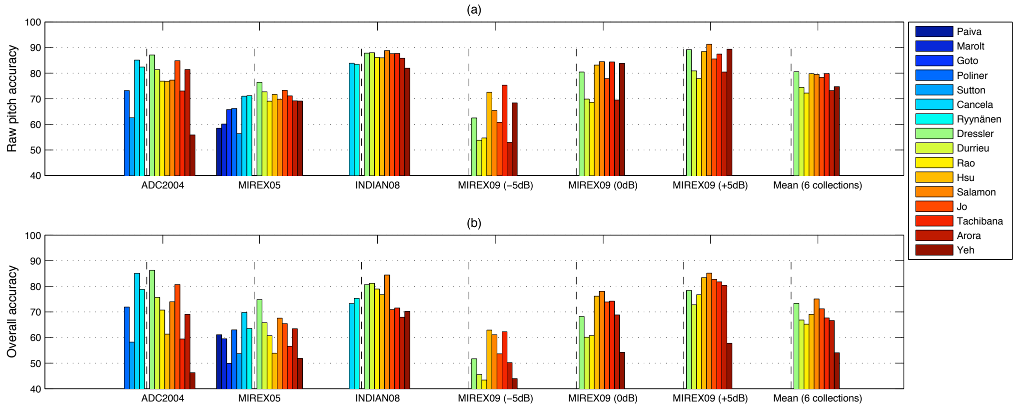


FIGURE 2.8: a) Raw pitch accuracy and b) overall accuracy obtained in MIREX by 16 melody extraction algorithms evaluated in [14]. Rhe vertical dashed line sepatates the algorithms that were only evaluated on some collections (left of the line) from those evaluated on all six collections (right of the line).

2.4.3 Comparison of both approaches

In their paper [14], 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).

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, however 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. 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.

2.5 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 3

Project Plan

In this project, we are attempting to design and develop a rhythm-based computer game where a user's music memory is tested by presenting him with a sequence of key presses corresponding to different characterisations of the music playing in the background.

So far, we have implemented a skeleton OS X application with a simple user interface, which imports and stores tracks chosen by a user for further analysis. We have also researched some of the available libraries useful for audio processing.

We would like to implement a game where user has an ability to log in and store a (possibly limited) amount of their levels. The application would also save the scores obtained at each of the levels. User will be able to personalise their account with a name and a picture/avatar, to distinguish themselves from other players using the same computer.

The game requires an implementation of a main melody extraction. This way we can obtain the melody line which then can be used to generate a level.

The process of the melody extraction will be followed by designing an implementing an algorithm for mapping the melody characteristics to the buttons shown on the screen. Such algorithm will be inspired by the guitar and piano playing techniques. If the next note we encounter has a higher pitch than the one we are currently looking at, the button shown to the user will be the one right of the currently highlighted one. Analogically, if the pitch of the next note is lower, it will activate the button on the right. The gap between the current and the next button will depend on the distance between the two notes analysed. In particular, if the estimated distance is impossible to shown, next button will be allocated in the round robin fashion.For example, in our level we have 5 possible buttons, we are currently looking at the rightmost one and the next one is supposed to be right of the current one, we will highlight the leftmost one.

Initially we hoped that the project could train a neural network to produce the buttons. However, it soon became clear that the music analysis section would require a large portion of time to implement, and that there would be insufficient time to design, train and evaluate a neural network system.

Much thought should be given to the implementation of the actual game. In particular, one of the challenges we will be facing is synchronising the visuals of the game, audio track and the reaction of the player. There are two issues to consider here. The most important one is how to make sure we interpret the player's input correctly, so that they feel like they are being rewarded accurately. In addition to this we should make sure that our graphics match the music, so that it looks like the actions are happening in sync with the music.

Instead of giving you a smooth, consistent output, the playhead/position of an audio file updates in steps. This is why we need to interpolate between those steps by keeping track of the playhead position with your own variable, and automatically adding time to that variable every frame. In order to take our songTime and keep it consistent with the actual playhead position of the audio file, I like to use a basic easing algorithm to apply corrections every time I get a fresh playhead position

In addition to this, we must take into consideration the delays in the rendering pipelines, as well as the delay that happens between the user hitting a key and the keystroke making it back to the program.In most games, this delay is completely negligible, but as we are implementing a rhythm-based video game, perfect accuracy is a must. That is why a visual delay test must be implemented and ran.

We would also like to implement some additional features to the game to make it more challenging and appealing. Those features could include an accumulator, which would allow the user to gain extra points if they have reacted to the note perfectly. In addition to this, we could make our button generation generate levels of different difficulty, depending on the seed provided by the user.

Index	Week commencing on	Task
1	2.02	Further research of the technologies, libraries.
2	9.02	Start developing the front end for the game.
3	16.02	Continue implementing the front end of the game, ask a few people to initially test the look and feel of the game.
4	23.02	Start implementing the melody detection algorithm - sinusoid extraction.
5	2.03	Finish implementing the Sinusoid extraction and start to implement the salience function.
6	9.03	Continue implementing the salience function. Revision for the exams.
7	16.03	Revision for the exams.
8	23.03	Exams.
9	30.03	Implementing pitch contour creation.
10	6.04	Finish pitch contour creation, implement the melody selection.
11	13.04	Finish the melody selection, further develop the game to fit the retrieved data on, designing the button mapping algorithm.
12	20.04	Further work on integration melody into the game, implementing and testing the button mapping algorithm, testing on a small group of users.
13	27.04	Work on integration melody into the gameplay, adjusting the game to take the delays into consideration.
14	4.05	Work on the final report. Implementing bonuses, achievements and other user centered features.
15	11.05	Further work on the final report.
16	18.05	Do shits
17	25.05	Evaluation - testing with users filling in questionnaires, writing the final report.
18	8.06	Polishing up the final report.
19	15.06	Final Report Due on 16.06 - submission, review and last touches to the code base.
20	22.06	Preliminary Source Code Archive due on 22.06
21	29.06	Final Project Archive due on 29.06

Chapter 4

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.

4.1 Formative

4.1.1 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?).

4.2 Summative

4.2.1 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.

4.2.2 Melody Extraction Testing

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.

4.2.3 Evaluation Framework

4.2.4 Release

4.2.2. interviews, questionnaires, surveys

Chapter 5

Bibliography

[1] Ian Bogost, How to Do Things with Videogames (Minneapolis, MN: University of Minnesota Press, 2011), 36.

[2] http://psp.about.com/od/pspglossary/a/Music-Game-Definition.html

[3] http://arstechnica.com/gaming/2009/01/guitar-hero-iii-first-game-to-reach-1-billio

[4] 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., vol. 15, no. 4, pp. 1247–1256, 2007.

[5] J.-L. Durrieu, G. Richard, B. David, and C. Fevotte, "Source/filter model for unsupervised main melody extraction from polyphonic audio signals," Trans. Audio, Speech and Lang. Proc., vol. 18, no. 3, pp. 564– 575, March 2010.

[6] http://www.originlab.com/index.aspx?go=Products/Origin/DataAnalysis/SignalProcessin STFT

[7] http://www.pixelplayer.se/bilder/texter/rec_8576_1_20081117.jpeg

[8] https://images-na.ssl-images-amazon.com/images/G/01/videogames/detail-page/gh3.lor.03.lg.jpg

[9] J. Salamon and E. Gómez, "Melody extraction from polyphonic music signals using pitch contour characteristics," IEEE Trans. on Audio, Speech and Language Processing, vol. 20, no. 6, Aug. 2012.

[10] http://en.wikipedia.org/wiki/Rhythm_game

[11] http://cdn-static.gamekult.com/gamekult-com/images/photos/00/00/34/99/ME0000349967_2.jpg

[12] http://en.wikipedia.org/wiki/Viterbi_algorithm

[13] http://en.wikipedia.org/wiki/Vibrato1

[14] J. Salamon, E. Gómez, D. P. W. Ellis and G. Richard, "Melody Extraction from Polyphonic Music Signals: Approaches, Applications and Challenges", IEEE Signal Processing Magazine, 31(2):118-134, Mar. 2014.

[15] Smith, Julius O. Introduction to Digital Filters with Audio Applications, W3K Publishing, http://books.w3k.org/, 2007