DM

Sonification of Gaming Experiences

MASTER DISSERTATION

Eduardo Câmara Gomes

MASTER IN INFORMATICS ENGINEERING

UNIVERSIDADE da MADEIRA

A Nossa Universidade

www.uma.pt

February | 2018

(Left in blank intentionally)

Sonification of Gaming Experiences

MASTER DISSERTATION

Eduardo Câmara Gomes
MASTER IN INFORMATICS ENGINEERING

SUPERVISOR
Mónica da Silva Cameirão

CO-SUPERVISOR
Sergi Bermúdez I Badia

(Left in blank intentionally)

Universidade da Madeira

Faculdade de Ciências Exatas e da Engenharia

Sonification of Gaming Experiences

Report of Dissertation

Master in Informatics Engineering

Eduardo Câmara Gomes

February 2018

Supervisors:

Mónica da Silva Cameirão

Sergi Bermúdez i Badia

(Left in blank intentionally)

Acknowledgements

To my mother, who always encouraged me to produce quality work and provided me with the possibility to continue doing so.

To my supervisors, Mónica da Silva Cameirão and Sergi Bermúdez i Badia, who always provided me challenges, guidance and feedback, striving to make this work interesting and an amazing learning experience.

To Athanasios Vourvopoulos, Harry Vasanth, John Muñoz and Lucas Pereira, who helped me to get on the right path for data analysis and machine learning.

To my NeuroRehabLab colleagues, for their constant feedback and help during this work.

To Madeira Interactive Technologies Institute and its members that helped and supported me one way or the other.

To University of Madeira, for providing the opportunity to work on this project and learn from it.

To Anna Aljanaki and Yi-Hsuan Yang, who promptly pointed me to their amazing work and helped me get started with feature extraction.

A Heartfelt Thank You!

(Left in blank intentionally)

Resumo

O uso de som em jogos tem-se vindo a tornar uma área de relevância para *designers* de jogos e jogadores. Com este novo interesse e novas tecnologias disponíveis, é pertinente criar ferramentas que facilitem e permitam estender as capacidades atuais de adicionar som a jogos.

A criação de experiências através de áudio consiste na escolha do mesmo, de maneira contextualizada e considerando o seu significado a nível emocional. Este trabalho tem como objetivo a criação de uma ferramenta que permita adicionar áudio a jogos considerando estes aspetos. Contudo, esta abordagem requer poder avaliar áudio a nível emocional e o impacto deste, recorrendo a técnicas de aprendizagem automática (*machine learning)*. A representação da carga emocional pode ser realizada utilizando o amplamente aceite modelo Circumplexo, que representa emoções através das dimensões *Valence* (quão positiva é a emoção) e *Arousal* (quão ativa é a emoção).

Este trabalho fornece então quatro contribuições. A primeira consiste em modelos computacionais que apresentam um aumento significativo de *performance* na análise da dimensão de *Valence*, à custa de um ligeiro decréscimo na dimensão de *Arousal*. A segunda contribuição consiste numa *interface* que permite ao *game designer* e jogador efetuar escolhas de áudio baseadas em emoções alvo e informação contextual. A terceira reside na escolha da interação entre o jogo e seus componentes com o áudio escolhido e como este será influenciado. Finalmente, a quarta contribuição consiste na alteração de características psicométricas do áudio em tempo real para este se adaptar aos objetivos delineados pelo utilizador.

Palavras-chave: Som; Jogos; Emoções; Modelo Circumplexo; Machine Learning

(Left in blank intentionally)

Abstract

Sound in games has increasingly become an area of interest for game designers and players alike. With this renewed interest and new technologies available, it is of relevance to create tools that facilitate and extend current capabilities of adding sound in games.

The creation of experiences through audio lies in its choice, through a contextualised way and considering its significance on an emotional level. The objective of this work was to create a tool that enables the addition of audio considering these aspects. However, such an approach requires the capacity of evaluating of audio on an emotion level and its impact, making use of machine learning techniques. The representation the emotional meaning carried by the audio can be achieved by using the widely accepted Circumplex model, presenting the results along the Valence (how positive the emotion is) and Arousal (how active the emotion is) dimensions.

This work then provides four contributions. The first consists on computational models that reveal a significant increase in terms of performance in the analysis along the Valence dimension at the cost of a slight decrease along the Arousal dimension. The second consists on a tool that allows the game designer and player to choose audio based on target emotions and contextualised information. The third contribution lies on the interaction of the chosen audio and the game itself, its components and how it will respond to it. Finally, the fourth contribution consists on the alteration of audio psychometrical characteristics in real-time, allowing for a better adjustment of audio in face of the user's objectives.

Keywords: Sound; Games; Emotions; Circumplex Model; Machine Learning;

(Left in blank intentionally)

Contents

List of Tables

(Left in blank intentionally)

List of Figures

Chapter 1: Introduction

Motivation

It is widely accepted and acknowledged that sound and music have an impact in human beings and can help to trigger diverse kinds of reactions. As such, the use of sound and music in games has as its main objective to help maintain the game flow and serve as an enhancer agent to the visual events presented and influence the player towards a certain state of mind, according to the idealization of the game designer.

Although sound and music in games is an area that is being constantly explored, the impact that it has on the emotions perceived by the players is still quite difficult to assess and control, making this the prime reason for the research conducted. The possibility for the game designer to control the emotions of the player, induced by sound and music, is very powerful and arguably adds to the game experience. This could be achieved by making use of the Sonification, a field that has as its primary objective the relay of information through sound, such as emotions. However, one of the biggest challenges lies on what audio is chosen for sending a specific message and there is no definitive answer from the researchers.

While the selection of the correct sound or music plays an important part on the listener's perceived emotion, research also suggest that changing certain audio features can also affect that perception.

Objectives

Considering what has been described above, this thesis has two main objectives. The first consists on the development of techniques for predicting listeners' perceived emotions resulting from an audio clip. The second lies on the ability to change those clips utilizing music features described by research as meaningful from an emotional point of view.

The result of the research is applied into a *Unity3D* project, with the objective of allowing game developers to choose the right sound or music for the right moment of the game, with the aim of enhancing the player's experience. As the prediction of the emotional content of audio is a challenge, the tool developed should also extend to the players, enabling them to choose audio according to their own idealization.

Thesis Outline

The present thesis is composed by a total six Chapters. The current Chapter, Chapter 1 is the Introduction, presenting to the readers an introduction of why this work was conducted and its objectives.

Chapter 2, Background, aims to present the readers with knowledge in the researched area by providing a review of the most relevant literature.

Chapter 3, Feature Extraction and Modelling, provides a detailed description of the methods and tools used to build the solution, as well as the reasons as to why these were chosen.

Chapter 4, Implementation, describes the implementation process of the proposed solution.

Chapter 5, Case Scenarios, shows the application of the tool in two games from different genres.

Chapter 6, Conclusions, draws conclusions from all the work done and provides some future work in this research field.

Chapter 2: Background

Sound, Music and Games

Sonification, an interdisciplinary field that involves Music, Human Computer Interaction, Computer Science and Psychology, among others, has as its main objective the representation of data through sound [1], [2]. Due to Sonification being a field that is composed of several others, it also finds applications in a wide range of situations, even in our daily lives, ranging from the notification sounds of an incoming message, alarm clocks, car locks, to music and speech. Movies are also part of the lengthy application list and is one that particularly benefits from it, immersing the viewer with the rich environment created and brought to life with realistic, quality sounds. For example, in [1] some projects using Sonification were analyzed and the areas of the studies consisted in Data Exploration, Art and Aesthetics, Accessibility, Motion Perception, Monitoring, Complement to Visualization and Study of Psychoacoustics. For each project analyzed, the mapping techniques (representation of a physical dimension, like distance, through a musical feature, such as Pitch) used were distinguished and counted. Some of the findings include the most common features used and the context of their usage.

Sonification in Games

In games, sound can be used to transmit information to represent the status of the player, such as the current health or motion, and the world state, such as the current weather. All of this serves as a complement to the visual component of the game. In contrast, the focus of this work lies in the application of sound and music in games on an emotional level. This potential emotional content has the possibility of supporting the storytelling element of games, building greater engagement levels and enjoyment. Sharing the storytelling element with movies, many authors note the similarities between the two mediums and state that music can be composed using resembling approaches. However, the two diverge on the audience roles, particularly in terms of action/reaction and duration. Games may be designed to be played repeatedly with different interactions, whilst movies are comprised of a fixed script staged once [3], [4]. Although a distinguishing feature, it also poses the problem of audio replayability and subsequent listener annoyance.

Game companies nowadays try to take advantage of sound and compositional methods to bring the experience up a notch, now with access to big orchestras. Regarding sound quality, a big step was taken, taking account the old 8-bit content where the compositions had to be very carefully managed in terms of memory and notes played (*Tetris* for example) [5]. Sonancia is an example of a system, built in *Unity3D*, that ties the game sound to the architecture of a level

and was designed for Horror games [6]. The sound played changes according to the position of the player in a room and according to the relative distance to the level objective. As for the sound event selection, the system builds knowledge of which sound was already played and removes these from the possible choices for the next audio clips. At the moment of publication, the Sonancia system is said to be extremely reliant on sound files but was still not fully ported into Unity. The authors also note that for future work more analysis is required for the intensity to map results more accurately and classify emotions experienced by the player. This task would use a model widely accepted in the literature, the circumplex model, classifying emotions using a pair of values that assess the positiveness and activeness of an emotion [6], [7].

Challenges in Game Music

As mentioned before, there are two main challenges while composing music for games. It needs to both adapt to the user actions and vary each time it plays to enable extended replayability. Considering this, some games already make use of a *stem* system. A *stem* is a short musical segment that can be combined with others, usually triggered by parameters or context, providing in many cases additional layers of sound. For example, in *Red Dead Redemption,* if the player starts to get chased an instrument starts playing and an additional one starts if the chase escalates to a shootout [8]. To accomplish this, all the *stems* were recorded at the same BPM and tonality. To tackle the replayability issue, Collins provides some input on how it is possible to diversify sound playback. The suggestions are as following: Variable Tempo, Variable Pitch, Variable Rhythm, Variable Volume, Variable Timbre, Variable Melodies, Variable Harmony, Variable Mixing, Variable Form (open form), Variable Form (branching parameter-based music). The *Red Dead Redemption stem* system can be considered a Variable Form (open form) since any number of *stems* can be connected seamlessly, as can be seen in Table 1 [4], [8].

Stem	Idle	Pastoral	Suspense	Dramatic	Chase	Chase Intense	Gunfight	Gunfight Intense
1	X	X		X				
2		X	X	X	X	X		
3			X					
4				X				
5					X	X		
6						X		
7								X
8							X	X

Table 1 - Red Dead Redemption Stem System Triggers [8].

Sonification Techniques

There are five techniques that can be considered widely accepted in the field of sonification [1], [2]. These consist of Audificaton, Auditory Icons, Earcons, Parameter Mapping Sonification and Model-Based Sonification [1], [2], [9]. The first technique, Audification, consists on the action of making a sound audible enough to interpret it and draw information from it. An example would be the Heart Rate measurements of the Photoplethysmograph (PPG) machines, by just increasing the amplitude of the sound. Auditory Icons are used to provide information when the user executes an action like taking a picture with a *smartphone*, in which the latter produces sound only for feedback purposes. Earcons are very like Auditory Icons but are sounds that the users have a connection to, or they set them up themselves. Parameter Mapping Sonification consists on mapping data values to acoustic features of a sound and finally, Model-Based Sonification uses data to build a system where the user can trigger the sounds of objects, like the strings of a musical instrument.

Sound Parameters

This section focuses on the analysis of the sound features, elaborating on how certain aspects of sound can be changed and in what way. The features presented here are not purely physical features, but more high-level ones. There are numerous sound features, but the ones that are widely used to describe audio are Pitch, Timbre, Sound Level and Tempo [1]. Depending on the type of information to be represented one can also make use of the Spatial aspect.

Pitch

Pitch, as commonly referred to, is determined by the frequency of a sound, being then perceived as low or high. It is possible to represent pitch as a logarithmic function with each note having an associated frequency in Hertz (Hz) [10]. A sub feature of Pitch is the Pitch Range, being the range of the Pitch over a segment of a sound or music. It is usually quantified as Small or Wide.

Timbre

In cases where two instruments play the same note, Timbre is the sound feature that allows the listener to identify the instruments as different. As such, it is the characteristic that helps to distinguish and separate instruments while using the same Pitch. It is usually classified as Bright, Soft, Sharp, Warm and Round, among others [11], [12].Timbre is a high-level characteristic, being composed of several others such as the Spectral Power and Spectral Flux and is heavily influenced by the sound envelope.

Sound Level

Referring to the Sound Pressure Level, can be measured in decibels (dB). It can be mistaken for Loudness, which is the Amplitude of a sound wave (complementing the Pitch). Sound level can roughly be considered a synonym of “Volume”.

Tempo

Is a measurement on how fast or slow a certain sound or music is, usually represented as Beats Per Minute (BPMs).

Data Transmission Through Sound

Sound is a way of transmitting data. As presented in [1], sound can also be used to map physical quantities and the authors conducted a study to find the use cases of musical features in the mappings of dimensions to sound. The findings of the study include Table 2, where a physical property is described utilizing sound features, ordered by use frequency. This suggests that some tendencies exist in the choice of audio, in terms of features, for relaying information.

Physical Property	**Sound Features (By usage frequency)**
Location	Spatialization, Pitch, Pitch Range, Instrumentation, Duration, Sequential Position
Distance	Loudness, Pitch, Duration
Motion	Pitch, Spatialization, Doppler Effect, Loudness, Duration
Density	Pitch, Duration, Loudness
Orientation	Pitch, Spatialization
Velocity	Pitch, Tempo, Brightness, Loudness
Size	Pitch, Duration
Spectral Signal Energy Distribution	Pitch, Spectral Power, Loudness
Pressure	Pitch
Energy	Loudness
Acceleration	Pitch, Loudness
Event Rate	Tempo
Temperature	Pitch
Signal Amplitude	Loudness
Signal Frequency	Pitch
Color Luminosity	Loudness
Mass	Pitch

Table 2 - Mapping of Physical Quantities to Sound Features by frequency of usage [1].

As an example, for representing Distance, the Loudness feature was used. Since the listener can infer from real-world experience that when an object is closer to the listener it is also louder, it makes a successful sonification. From Table 2 it is possible to see that the most recurred sound features for the description of quantification of physical properties are the Pitch, Loudness and Duration. Particularly, Pitch appears in 12 out of the 17 physical properties, making it the most used musical feature throughout the several projects analyzed. During this work, the features mentioned will be some of the most explored.

Classification of Emotions

One of the existing definitions for emotions describes them as the mental states of the person, causing “bodily disturbances” and then perceiving the latter as emotions [13]. However, the definition of what is an emotion has still not achieved a general agreement among the scientific community [14], [15]. The emotion definition varies, mostly from the determining factor on what impacts the recognition of an emotion such as the culture and experience.

Considering emotions as a type of information, these would have the possibility to be transmitted through sound. Some examples can be found in the cinematographic industry, having music purposely composed for a specific reason, or movie [16]. However, due to the subjectivity associated to the definition of this concept and the variety of ideas linked to it, the recognition of emotion needed for said purposes carries the same problem as the definition. Russel, in [7], attempted to bypass this challenge by introducing a way to classify emotions using a model composed of two axis, called the *Circumplex Model of Affect*.

Circumplex Model of Affect

Regarding the representation of emotions, the widely-accepted model is the Circumplex Model (see Figure 1). This model is not limited to a sound/music context, being used as a standard to emotion representation and comparison of different individuals and their corresponding states. Each emotion is represented by a pair of values, Valence on the X axis and Arousal on the Y axis [7]. Valence shows how positive or negative the emotion is, ranging from pleasant to unpleasant. Arousal describes the level of engagement that a person felt towards a certain stimulus, ranging from low activation to high activation. Using a pair of values of Valence and Arousal, it is possible to place an emotion on a 2D space composed by four quadrants. The four quadrants are the positive Valence and positive Arousal, negative Valence and positive Arousal, negative Valence and negative Arousal, and positive Valence and Negative Arousal.

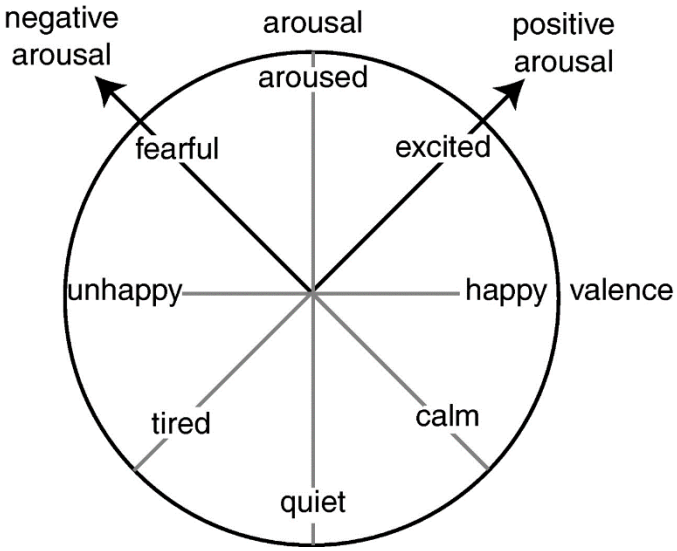

Figure 1 - Circumplex Model [17].

There has been some debate about other ways of representing emotions, particularly about using three-axis for representation and how it could improve the emotion labelling for music and sound. One of the studies that tries to observe the differences between such a model and the Circumplex Model replaces the Arousal axis for Activity and adds a third axis that according to the authors allow for a better representation of emotions such as Fear and Anger, that would otherwise be almost indistinguishable using the Circumplex Model [18]. However, these models are still not widely accepted due to the discussion about the introduction of the third axis, as well as what it should represent. An example of such model is presented in Figure 2, with the third axis representing Tension.

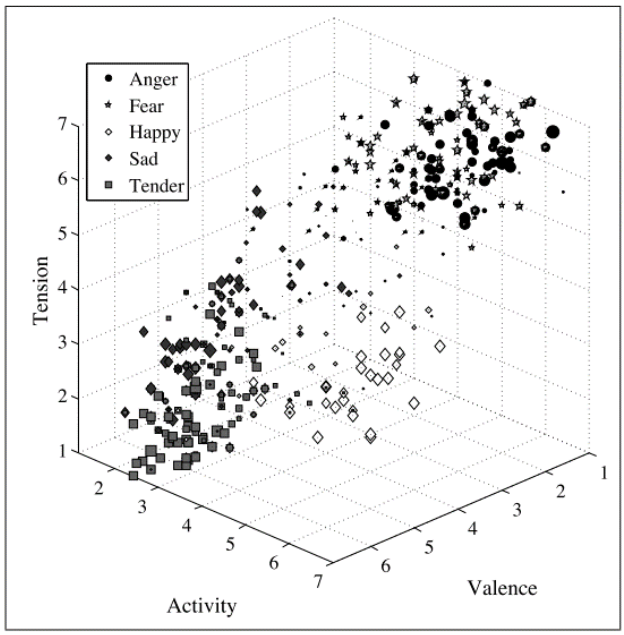

Figure 2 - Example of a three-axis representation of emotions [18].

Mapping Valence and Arousal

A popular way of attributing pairs of Valence and Arousal values is to make use of the Self-Assessment Manikin (SAM), visible on Figure 3 [19]. This technique consists on using scales with figures representing each of the Circumplex's dimensions, making it a fast to apply solution for emotion rating. A third scale is also available under the name *Dominance*.

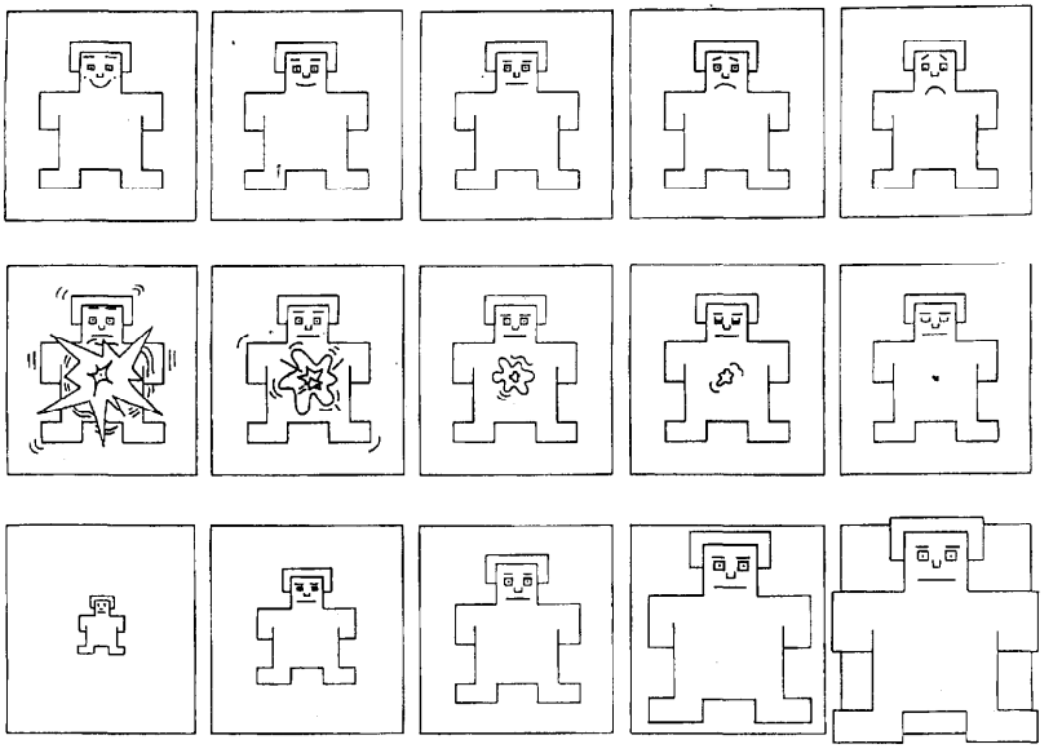

Figure 3 - SAM scales: Valence, Arousal and Dominance (Top to Bottom) [19].

In a research study conducted based on automatic playlist generation using affective computing, the authors attempted to create a playlist of music based on the emotion of the user using psychophysiological data. This was to be acquired from various physiological sensors, ranging from Photoplethysmograph (PPG) to Electrodermal Activity (EDA). This would result in psychophysiological indices such as heart rate and skin conductance to predict the current emotion of the listener [20]. Some studies also show that Heart Rate and EDA correlate to the Arousal levels of the user [21].

Sound Features and Emotions

There have been studies that tried to observe patterns in terms of features and musical characteristics on music gathered that had been previously labelled. One of such studies reports the findings of the most common characteristics found in Happiness, Sadness, Fear, Anger and Tenderness [22], [23].

Emotion	Musical Feature
Happiness	Fast tempo, small tempo variability, major mode, simple and consonant harmony, medium-high sound level, small sound level variability, high pitch, much pitch variability, wide pitch range, ascending pitch, perfect 4th and 5th intervals, rising micro intonation, raised singer's formant, staccato articulation, large articulation variability, smooth and fluent rhythm, bright timbre, fast attacks, small timing variability, sharp contrasts between 'long' and 'short' notes, medium-fast vibrato rate, medium vibrato extent, micro-structural regularity
Sadness	Slow tempo, minor mode, dissonance, low sound level, moderate sound level variability, low pitch, narrow pitch range, descending pitch, 'flat' (or falling) intonation, small intervals (e.g. minor 2nd), lowered singer's formant, legato articulation, small articulation variability, dull timbre, slow attacks, large timing variability (e.g. rubato), soft contrasts between 'long' and 'short' notes, accelerando, medium-fast vibrato rate, large vibrato extent, micro-structural irregularity
Anger	Fast tempo, small tempo variability, minor mode, atonality, dissonance, high sound level, small loudness variability, high pitch, small pitch variability, ascending pitch, major 7th and augmented 4th intervals, raised singer's formant, staccato articulation, moderate articulation variability, complex rhythm, sudden rhythmic changes (e.g. syncopations), sharp timbre, spectral noise, fast attacks/decays, small timing variability, accents on tonally unstable notes, sharp contrasts between 'long' and 'short' notes, accelerando, medium-fast vibrato rate, large vibrato extent, micro-structural irregularity
Fear	Fast tempo, large tempo variability, minor mode, dissonance, low sound level, large sound level variability, rapid changes in sound level, high pitch, ascending pitch, wide pitch range, large pitch contrasts, staccato articulation, large articulation variability, jerky rhythms, soft timbre, very large timing variability, pauses, soft attacks, fast vibrato rate, small vibrato extent, micro-structural irregularity
Tenderness	Slow tempo, major mode, consonance, medium-low sound level, small sound level variability, low pitch, fairly narrow pitch range, lowered singer's formant, legato articulation, small articulation variability, slow attacks, soft timbre, moderate timing variability, soft contrasts between long and short notes, accents on tonally stable notes, medium fast vibrato, small vibrato extent, micro-structural regularity

Table 3 - Observed features per emotion [22].

Sound Features and Valence-Arousal Models

In a study conducted by Gomez and Danuser in [24], 31 participants were asked to listen to a total of 32 excerpts of 30 seconds each, 16 excerpts being noise and the remaining 16 were taken from instrumental passages of Western music. SAM was used to classify each of the excerpts in both Valence and Arousal. For each of the participants the authors fitted a curve trying to predict the value of a musical feature based on a given value of Valance and Arousal. The curves are described by the equation:

$$Y = \alpha + \beta v\, V + \beta a\, A + \beta va\, V \times A + E \qquad (1)$$

where Y is the musical feature, the β are the coefficients for the respective feature for Valence and Arousal, V and A are the values of Valence and Arousal and E consists of the error. The results can be found on Table 4, presenting the uncovered coefficients for each dimension, organized by musical feature [24].

Musical Feature	Constant	Valence Estimate Coefficient	Arousal Estimate Coefficient	Valence * Arousal estimate Coefficient
Sound Intensity	58.40	0.50	2.14	-0.18
Tempo	1.20	0.10	0.43	-
Rhythm	2.23	0.16	0.35	-0.05
Accentuation	2.24	0.00	0.46	-
Rhythmic Articulation	7.92	-0.26	-0.33	-
Melodic Direction	2.69	0.12	0.00	-
Pitch level	4.54	0.05	-0.07	-
Pitch range	2.17	0.22	0.37	-0.05
Mode	6.40	-0.27	-0.71	0.11
Complexity	4.17	-0.19	0.07	-
Consonance	5.80	0.01	-0.37	0.04

Table 4 - Results of Curve Fitting [24].

Through a coefficient analysis it is possible to observe that Sound Intensity is the feature that possesses the highest impact on both Valence and Arousal, having the highest absolute values on each of the categories. It is followed by Mode (absolute value) in terms of Valence and Arousal, and, can be stated that together with Pitch Range and Rhythmic Articulation, these are the features that exert more influence while listening to distinct types of sounds. Furthermore, a study that analyzes the automatic recognition of emotion caused by audio events found that Arousal is highly correlated to Loudness, while the latter is negatively correlated to Valence. Valence correlates negatively to Spectral Flux and especially to Spectral Harmonicity. It is also mentioned that loud sounds are unpleasant [25].

Prediction of Valence and Arousal in Sound and Music

Researchers have been trying to predict not the features of a song, but the emotion perceived by the listeners through a certain audio event. Although there is a lot of uncertainty on the ongoing research, machine learning algorithms are the preferred approach and usually have as inputs some signal component values. Acquiring the ground-truth and a baseline to compare the outputs of such approaches can be hard to attain, hence some musical platforms ask the listeners to tag the songs according to the perceived emotion of the user [26].

Yang et al. tackled the problem by exploring regression algorithms, through Multiple Linear Regressions (MLR), Support Vector Regressions (SVR) and AdaBoost.RT (BoostR). Out of the three regression models trained SVR achieved the best results, having a R^2 value superior in both Valence and Arousal, with 28.1% and 58.3% respectively [27]. Chin et al. proposed in 2017 to use a probability density function (PDF) over the Valence-Arousal space to identify a possible location of an emotion rather than an accurate value for each dimension, trying to minimize the subjectivity of the task. This showed the challenge of handling a big quantity of data and how it impacted the results, comparing the

obtained values for two different datasets, mostly differing in the number of samples [28]. Aljanaki, Soleymani, and Yang proposed a dataset to use for the exploration of techniques to predict emotions, composed by three datasets, each one presenting values for several signal features and Valence and Arousal. In total, this dataset is composed of 1802 song excerpts, following an established protocol for the data collection. Other information is available online such as the free emotion label, provided by the participants, although contact with the authors is required [29], [30].

Chapter 3: Feature Extraction and Modelling

Much of the work presented on this thesis is about the prediction of emotions in sound and how to provide a way to use it in games. As a result, the first part of this chapter focuses on the prediction algorithms used and the second delves on the benchmarking of the models obtained using those methods.

As established before, the current methods for predicting Valence and Arousal are quite unsuccessful in their tasks, achieving very low prediction rates especially while trying to model the Valence dimension. This serves as an opportunity to improve on those results, by following another approach in terms of algorithms. By doing so, the goal of this chapter is to pinpoint the emotion that a sound file would produce, predicting both dimensions of the Circumplex Model, Valence and Arousal.

Approaches

Using the DEAM dataset, that contains continuous values for physical acoustic features and Valence and Arousal SAM values for several excerpts, the goal was to identify how to correctly replicate (for existing datasets) and produce new values for both Circumplex dimensions (new datasets). Since there was existing accessible data, the chosen algorithms were supervised learning algorithms. Furthermore, since the resulting models will have to predict continuous values, this consisted of a regression problem. A total of twenty-two models were trained using the tools available from MATLAB and compared for the end results of this chapter. These models could be grouped into four categories, namely Linear Regressions (2), Support Vector Machines (6), Gaussian Process Regressions (4) and Artificial Neural Networks (10).

Linear Regressions

Linear Regressions are a set of supervised learning algorithms, used for predicting datasets that follow a pattern, and, like all other regression algorithms, their objective is to describe a relationship between a set of input and output variables. The result consists of fitting a straight line through the data, following the trend of the set. These algorithms are usually evaluated using the R^2 value, ranging from zero to one, effectively giving an approximation of how much of the dataset the model can predict. For example, a R^2 value of 0.41 would mean that the model can predict 41% of the data it was exposed to. Another meaningful value is the RMSE, Root Means Square Error, a squared average of how much the predictions are distant from the original values. This indicator does not have a standard numerical value since it depends on the dataset and although a low value is considered adequate, it can also be a sign of the model suffering from overfit. One of the advantages of using these algorithms lies on the fast training

time for the models, making them very timely cost-efficient. The output of the linear models presented here follow the equation:

$$y = \sum_{i=1}^{m} \beta_i x_i \qquad (2)$$

where y represents the result for the desired dimension, β represents the coefficient value for each feature x, and m represents the total of features used [31].

Support Vector Machines

Support Vector Machines (SVMs) aim to divide the dataset by making use of data-points in the set, called Support Vectors and are identified by a kernel function. While usually used for classification purposes, these can be also applied for regression. Visually, SVMs applied for regressions can be seen passing a separating line on the data and are enveloped by the error function area, as seen in Figure 4.

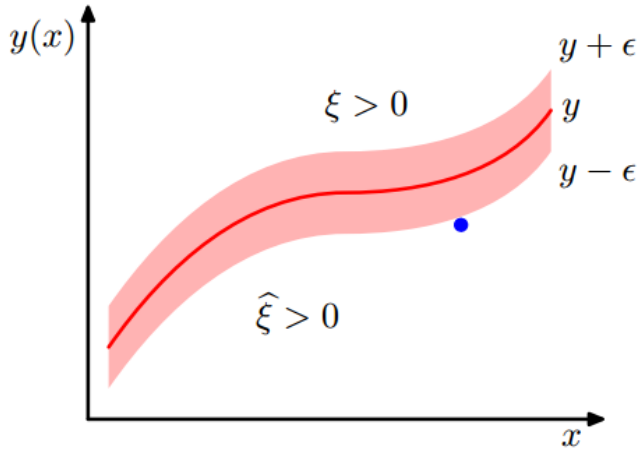

Figure 4 - Support Vector Machine Regression Example [32].

The equation presented by Bishop to perform predictions of new sets of inputs is given by:

$$y(\mathrm{x}) = \sum_{n=1}^{N} (a_n - \hat{a}_n) k(x, x_n) + b \qquad (3)$$

a_n and $\hat{a}_n$ are Lagrangian multipliers, a technique used in several optimization problems, and $k(x, x_n)$ represents the utilized kernel function. Some types of kernel functions are known as Linear and Gaussian functions. A kernel function is a way of representing data using other dimensions, in a way that it is possible to make clear distinctions groups of points, allowing a better separation and classification [32]. Moreover, the speed and efficiency of the SVMs depends on the kernel function chosen, as well as the size of the dataset and the noise that is present. In terms of evaluating the resulting models, the indicators are the same and follow the same logic as the ones presented on the Linear Regressions.

Gaussian Process Regressions

Gaussian Processes are a set of supervised learning algorithms that generate probabilistic models and attempt to accurately predict values, with an associated area over space that follows the Gaussian Distribution, with 95% of confidence. Rasmussen and Williams define Gaussian Processes as

a collection of random variables, any finite number of which have a joint Gaussian distribution [33]. When used for regression, these algorithms generate a curve with the associated area. An example can be observed in Figure 5, where three functions where drawn used Gaussian Processes and the grayed zones corresponds to the 95% confidence area.

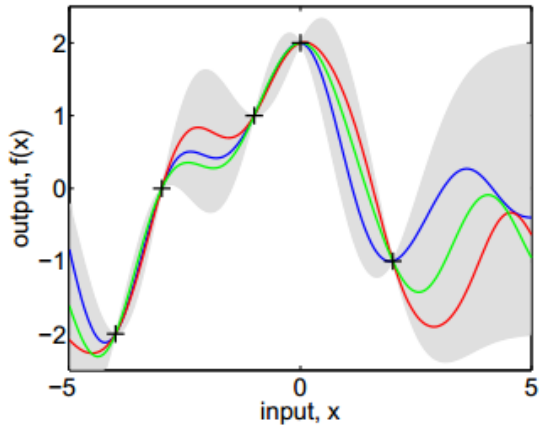

Figure 5 - Gaussian Process - Resulting Functions and Confidence Areas [33].

Gaussian Processes, similarly to Support Vector Machines, also make use of kernel functions and some of the widely known include the Squared Exponential and Rational Quadratic kernels. The calculated values for the resulting models also include R^2 and RMSE values. However, Gaussian Processes tend to be more affected by noisy data than Support Vector Machines, while the latter suffer from more complex kernel operations that are simpler when working with Gaussian Processes.

Artificial Neural Networks

Artificial Neural Networks (ANNs) were inspired on the biological brain and try to emulate its way of operating. The architecture of the ANNs used for this work is named Feedforward Artificial Networks and consist on a set of inputs, followed by a hidden layer where several operations are performed on the input data. The hidden layer is then proceeded by the output layer, adjusting the previous' layer results to produce outputs on a scale that makes sense for the current problem and finally generating the outputs. The architecture of this ANN can be seen in Figure 6. The equation for an ANN model can be written as:

$$y = \sum_{n=1}^{N} x_n w_n + b \qquad (4)$$

where the output value is given by a sum of the multiplication of the x_n input and the w_n weight with the added bias (b). For the presented ANNs, the Bayesian Regularization Backpropagation algorithm was used and has as goal the modification of the weights in order to minimize the error produced and better generalize the data. However, this algorithm is severely affected by the existence of local minimum points, meaning that the step and the starting point for the descent can produce different outcomes each time an ANN is trained.

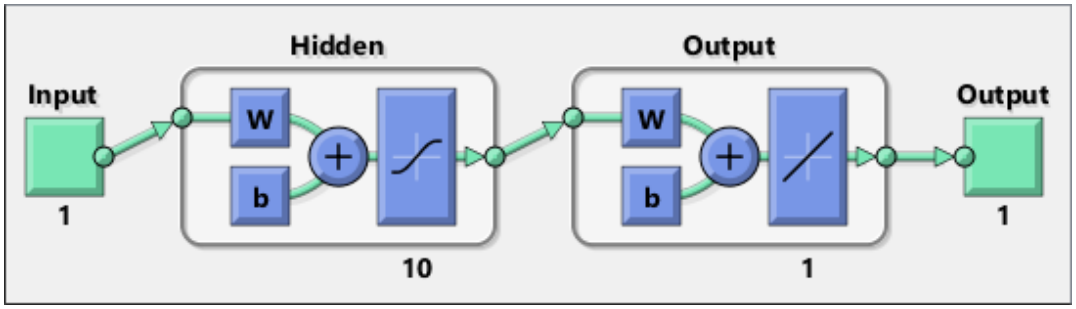

Figure 6 - Feedforward Neural Network Architecture [34].

The training speed for ANNs depends on the input data, the learning algorithm and the size of the hidden layer. The size of the hidden layer is associated with the number of neurons it contains and can be modified to produce better results. While increasing the number of neurons leads to a better learning curve, it can also increase the risk of overfit, resulting in a model that cannot accurately predict data not analyzed during the learning stage. The Bayesian Regularization Backpropagation is an established ANN training algorithm that minimizes the risk of overfit. This leads to a higher possible number of neurons present in the hidden layer for training.

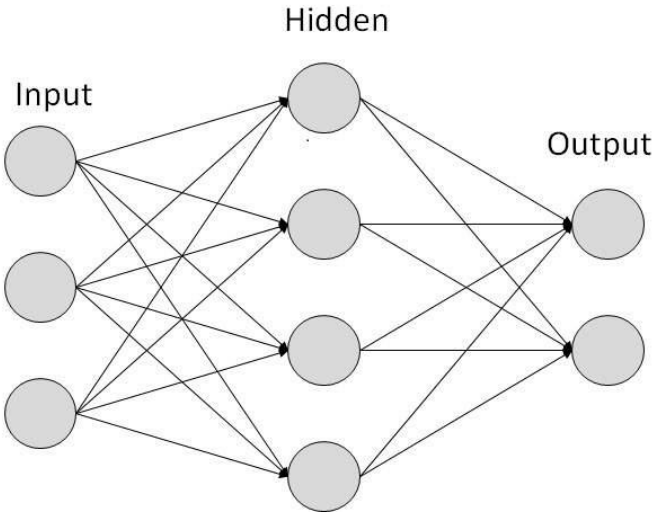

Figure 7 - Feedforward Neural Network [35].

In Figure 7 is possible to observe that three inputs are present, feeding into a hidden layer with four neurons. After all the functions and outputs are calculated, two output values are generated.

Predicting Arousal and Valence

Dataset Characteristics

Initially, two sub-sets of the DEAM dataset (MediaEval2013, 744 audio clips, and MediaEval2014, 1000 audio clips) were used to train several different models for the prediction of Valence and Arousal, separately, and one sub-set to test (MediaEval2015, 58 audio clips) [29], [30]. For the MediaEval2013 and MediaEval2014 datasets, each song was classified in Valence and Arousal by at least 10 participants. MediaEval2015 counted with five participants for each song, also being classified in Valence and Arousal. To classify each of the datasets, the authors made use of an online platform called MTurk where participants were recruited and compensated for their contribution. Each participant would rate the audio clip that was played and rate it by making use of the interface shown in Figure 8, identical for both Valence and Arousal. Dynamic values for Valence and Arousal (over the duration of the clip) were also available, as well as static value collected using the Self-Assessment Manikin.

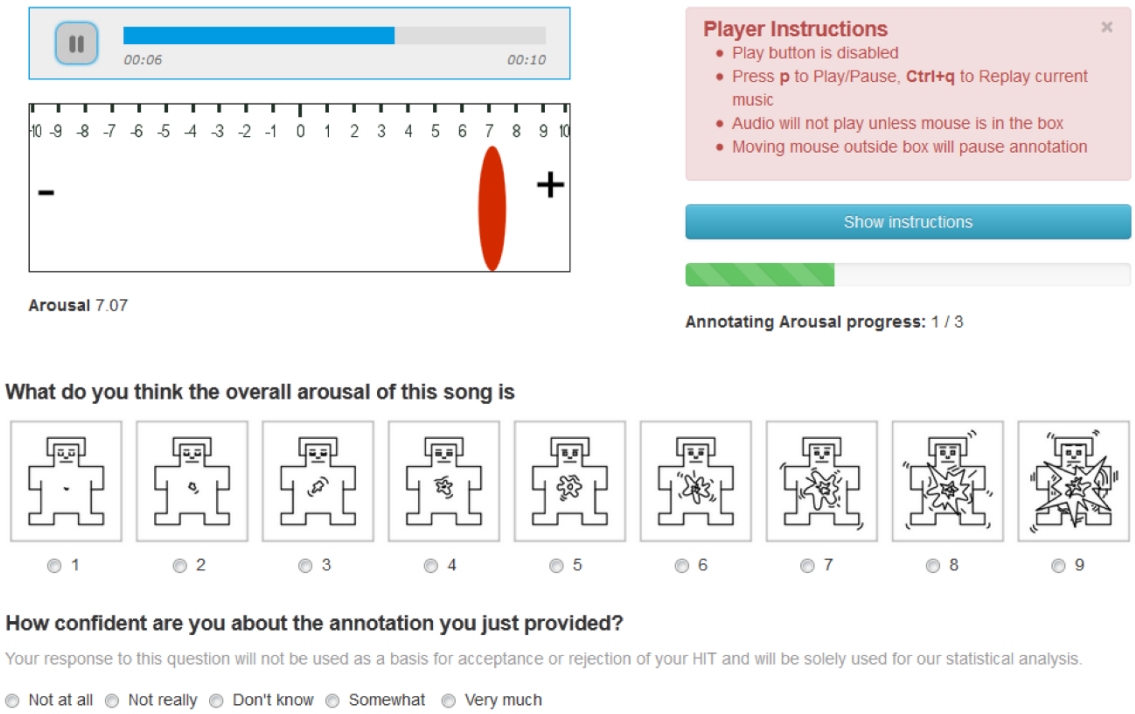

Figure 8 - Annotation interface for the DEAM dataset classification [29].

However, this lead to very poor results, leading to the integration of the MediaEval2015 sub-set into the remaining and resulted in the usage of the full DEAM dataset. The data also contained the labels for the musical genre of each of the musical excerpts analyzed, making for a possible genre classification task. The genres included are Rock, Pop, Soul, Blues, Electronic, Classical, Hip-hop, International, Experimental, Folk, Jazz, Country, Rap and Reggae. Moreover, several of the entries are followed by an excerpt of the same musical genre.

Each of the entries on this dataset possesses values for 260 acoustic features over their duration, extracted using openSMILE and at 2Hz, as well as annotation values for Valence and Arousal at the same frequency. For each of the excerpts is also provided a standard value for Valence and Arousal using the Self-Assessment Manikin (SAM), ranging between 1 and 9. For consistency, the only values for Valence and Arousal considered were the SAM ones. This lead to a mismatch in the dataset dimensions, having multiple lines of values for each feature column and only a single end value for Valence and Arousal. This was solved by calculating the mean of each feature column and the median of every participant for Valence and Arousal, for any given excerpt. As such, a single entry is comprised of a single line with 262 values, the first 260 being the features, the 261st being Valence and the 262nd being Arousal.

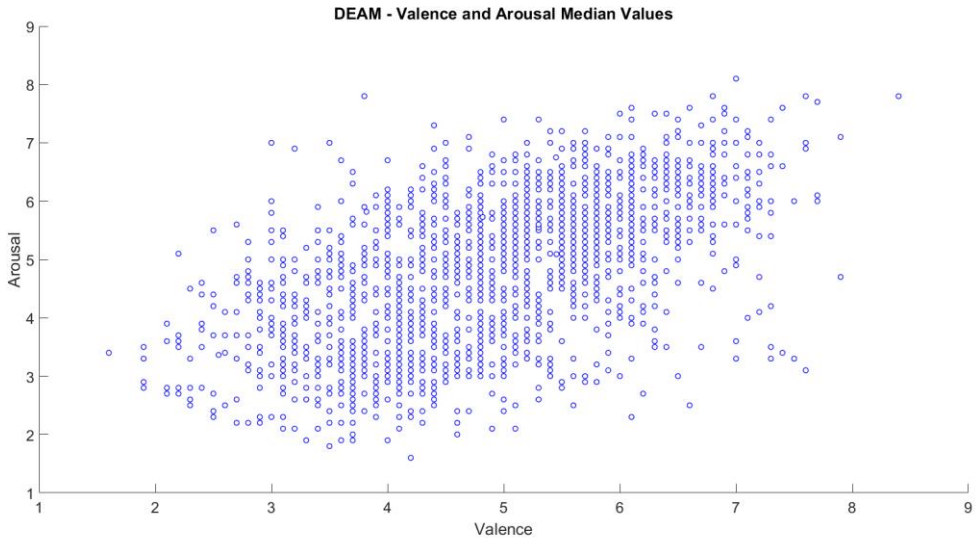

Figure 9 - Post-Processed Valence and Arousal Values.

Having finished the pre-processing of the full dataset, 15% randomly selected entries were selected and made up a Testing Set with a total of 271 entries. The remaining 85% were used then to build and train the models presented throughout this chapter. All the models were trained using 70% of the Training set and the remaining 30% were used for Validation and Testing purposes.

It is also worth mentioning that the data division for Training, Validation and Testing was done by sequential blocks to guarantee that every model was trained with the same dataset. For example, in the case of the training data being comprised of a single musical genre. Afterwards, the trained models were exposed to the previously established Testing set (15% of the DEAM dataset) and compared.

Model Specifics

As previously described, the four groups of trained models are the Linear Regressions, Support Vector Machines, Gaussian Process Regressions and Artificial Neural Networks. Table 5 contains the exact type of model trained as well as the categories it belongs to.

Linear Regressions	Linear Regression Robust Linear Regression
Support Vector Machines	Linear SVM Quadratic SVM Cubic SVM Fine Gaussian SVM Medium Gaussian SVM Coarse Gaussian SVM
Gaussian Process Regressions	Rational Quadratic GPR Squared Exponential GPR Matern 5/2 GPR Exponential GPR
Artificial Neural Networks	Two-layer Feedforward Neural Network

Table 5 - Algorithms used for predicting Valence and Arousal ordered by group.

It is also worth noting that for both Valence and Arousal, ten Artificial Neural Networks were trained varying only the Hidden Layer size, starting with only one neuron and incrementing the amount until ten.

Regarding Linear Regressions, Support Vector Machines and Gaussian Process Regressions, the output values are the resulting models' RMSE, Adjusted R^2, MSE and MAE. The Linear Regressions presented are the standard Linear Regression and the Robust Linear Regression. The latter ignores some of the farthest points on the dataset, the outliers, reducing the impact of noisy data. This allows for a better fit of the higher spread of data.

Considering Support Vector Machines, the models trained make use of four main different kernels, the Linear, Quadratic, Cubic and Gaussian. The differences between the Fine Gaussian, Medium Gaussian and Coarse Gaussian are presented in MATLAB's documentation. The parameters were left on the default settings.

The Gaussian Process kernels utilized for this work, also presented in MATLAB's documentation, are evolutions of the Squared Exponential kernel, also known as

Radial-basis Function (RBF) kernel. The parameters were left on the default settings.

For Artificial Neural Networks, the calculated values are the Training and Testing R. Training R represents the Correlation Coefficient between the calculated values of the model and the original values of the training dataset, effectively describing how well the model can predict already analyzed values. Testing R represents the Correlation Coefficient between know data not analyzed and the output calculated by the model.

Another value presented utilized is the Training Time, in seconds, took by the software to build and train the respective model.

Model Results

To assess the goodness of the Linear Regressions, Support Vector Machines and Gaussian Process Regression models, the Adjusted R^2 (primary) and the Training Time (secondary) were chosen as metrics for evaluation. While a value for R^2 cannot be negative, the presented Adjusted R^2 can, effectively describing a worse fit than a horizontal line passed through the dataset.

Regression Based Models - Valence

Taking from the literature, Valence is the dimension that is harder to predict due to the subjectivity of what an individual considers positive or not. Table 6 shows the resulting values of the model training for Valence.

Model	**RMSE**	**Adjusted R^2**	**MSE**	**MAE**	**Training Time (sec)**
Linear Regression	0.93	0.4	0.86	0.72	3.2837
Robust Linear Regression	0.93	0.39	0.87	0.73	2.9825
Linear SVM	0.87	0.46	0.76	0.68	2.9856
Quadratic SVM	1.03	0.25	1.06	0.76	2.5201
Cubic SVM	1.71	-1.06	2.93	0.93	2.448
Fine Gaussian SVM	1.21	-0.02	1.45	0.98	0.68312
Medium Gaussian SVM	0.86	0.47	0.75	0.69	0.8373
Coarse Gaussian SVM	0.88	0.45	0.78	0.7	0.65381
Rational Quadratic GPR	0.86	0.48	0.74	0.67	28.132
Squared Exponential GPR	1.19	0	1.42	0.98	42.946
Matern 5/2 GPR	0.86	0.48	0.73	0.67	20.792
Exponential GPR	0.86	0.48	0.73	0.67	19.961

Table 6 - Valence Models - Linear Regressions, Support Vector Machines and Gaussian Process Regressions.

From the results presented in Table 6, it is possible to see that the Cubic SVM and Fine Gaussian SVM do not fit the dataset at all. The best results were achieved by the Rational Quadratic, Matern 5/2 and Exponential GPRs. However, the Training Time was much higher when compared to other model results. For a better comparison of the model performance, the Adjusted R^2 is plotted for each model in Figure 10.

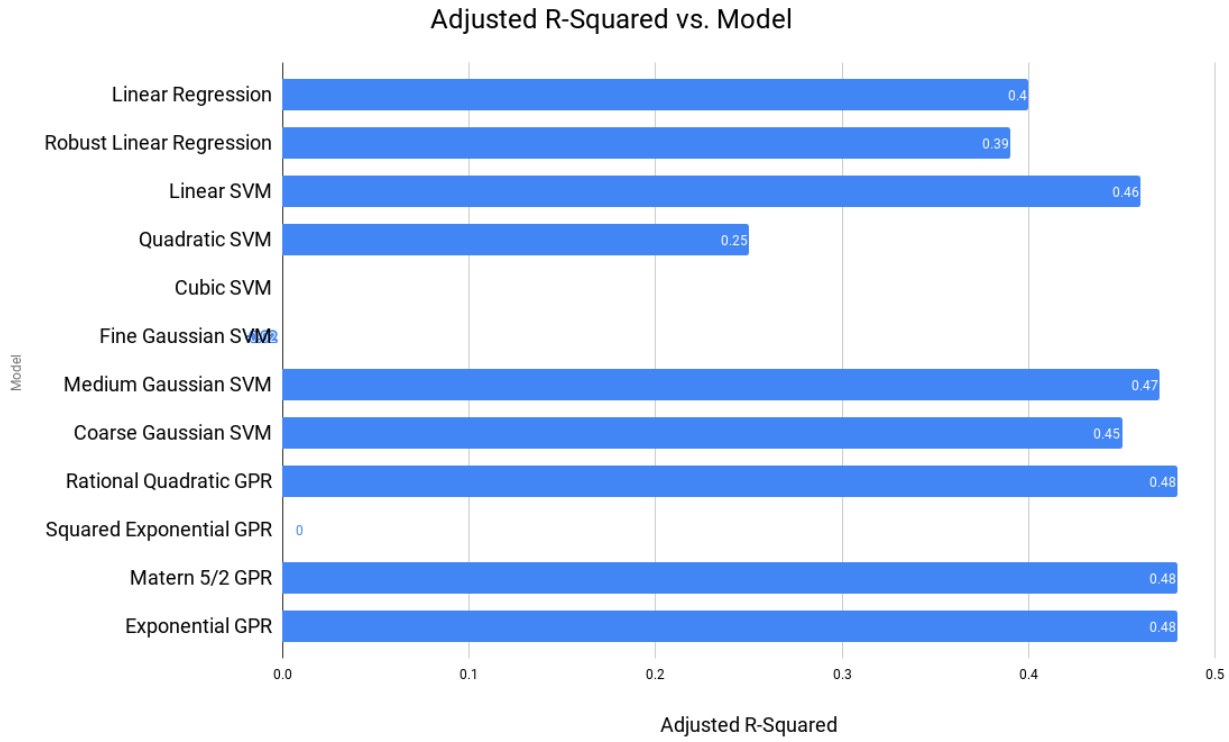

Figure 10 Valence Adjusted R-Squared Values per Model.

Artificial Neural Networks - Valence

Regarding Artificial Neural Networks, the metric used to assess the models was the Testing R, representing how well the model fared with data that it was never exposed to. Table 7 presents the results for all the ANNs trained to predict Valence.

Layer Size	Training R	Testing R	Training Time (sec)
1	0.76455	0.71127	4.146
2	0.76708	0.75616	83.452
3	0.8117	0.71926	215.919
4	0.82583	0.72005	470.343
5	0.83405	0.65947	382.383
6	0.86202	0.66153	436.584

7	0.89333	0.60519	771.228
8	0.82467	0.69374	1152.84
9	0.93694	0.46116	1336
10	0.92187	0.55447	1598.852

Table 7 - Valence Models - Artificial Neural Networks.

Table 7 shows that while increasing the number of neurons in the hidden layer results in a better prediction rate for the training data, the prediction for the testing data does not improve in the same way. In fact, the tendency for the testing values prediction is negative, always dropping when the number of neurons is greater than two. The only exceptions are when the number of neuron equals to eight and ten, possibly explained by the finding of another local minimum on the training algorithm. Figure 11 plots both the Training R and Testing R against the Layer Size for a better visualization of the performance.

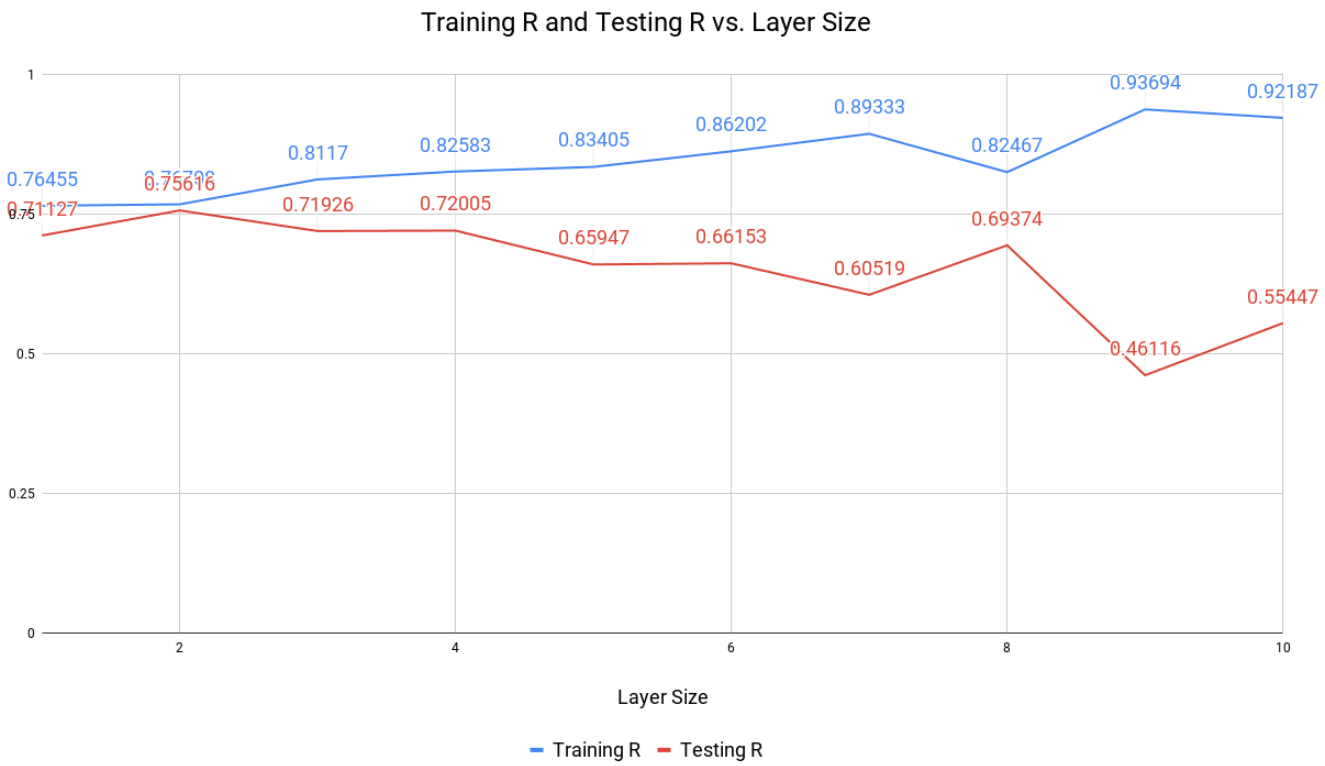

Figure 11 Results for Valence using Artificial Neural Networks.

It is possible to see that the most successful network had a hidden layer size of two (with a Testing R of 0.75616). This was followed by networks with a hidden layer size of four and three.

Regression Based Models - Arousal

The same metrics were used for the evaluation of the models predicting Arousal. This includes the Adjusted R^2 and Training Time as the metrics for the Linear Regressions, Support Vector Machines and Gaussian Process models. Expected Arousal values for the Adjusted R^2 values are around 0.50 according to the literature.

Model	RMSE	Adjusted R^2	MSE	MAE	Training Time (sec)
Linear Regression	1.1	0.31	1.21	0.87	3.2677
Robust Linear Regression	1.11	0.29	1.24	0.88	2.166
Linear SVM	1.01	0.42	1.01	0.8	2.5042
Quadratic SVM	1.4	-0.12	1.97	0.91	1.5445
Cubic SVM	15.95	-143.61	254.49	1.76	1.4488
Fine Gaussian SVM	1.31	0.02	1.73	1.09	0.42306
Medium Gaussian SVM	0.94	0.5	0.88	0.75	0.46666
Coarse Gaussian SVM	1.02	0.41	1.04	0.83	0.38189
Rational Quadratic GPR	0.92	0.52	0.85	0.74	18.322
Squared Exponential GPR	1.33	0	1.76	1.11	24.534
Matern 5/2 GPR	1.33	0	1.76	1.11	13.605
Exponential GPR	0.93	0.51	0.87	0.75	10.324

Table 8 - Arousal Models - Linear Regressions, Support Vector Machines and Gaussian Process Regressions.

Following the same trend as with the Valence models, the best results belong to the Gaussian Process Regressions, with the Rational Quadratic GPR model presenting the best Adjusted R^2 value, followed by the Exponential GPR one. While the Training Times from the Gaussian Processes are very steep comparing to the rest of the model groups, the difference between them is not as high as in Valence. Figure 12 presents the plotted Adjusted R^2 per model for an easier comparison.

The best results were achieved by Rational Quadratic GPR, Exponential GPR ad Medium Gaussian SVM. These three models also produced some of the highest results along the Valence dimension presented above.

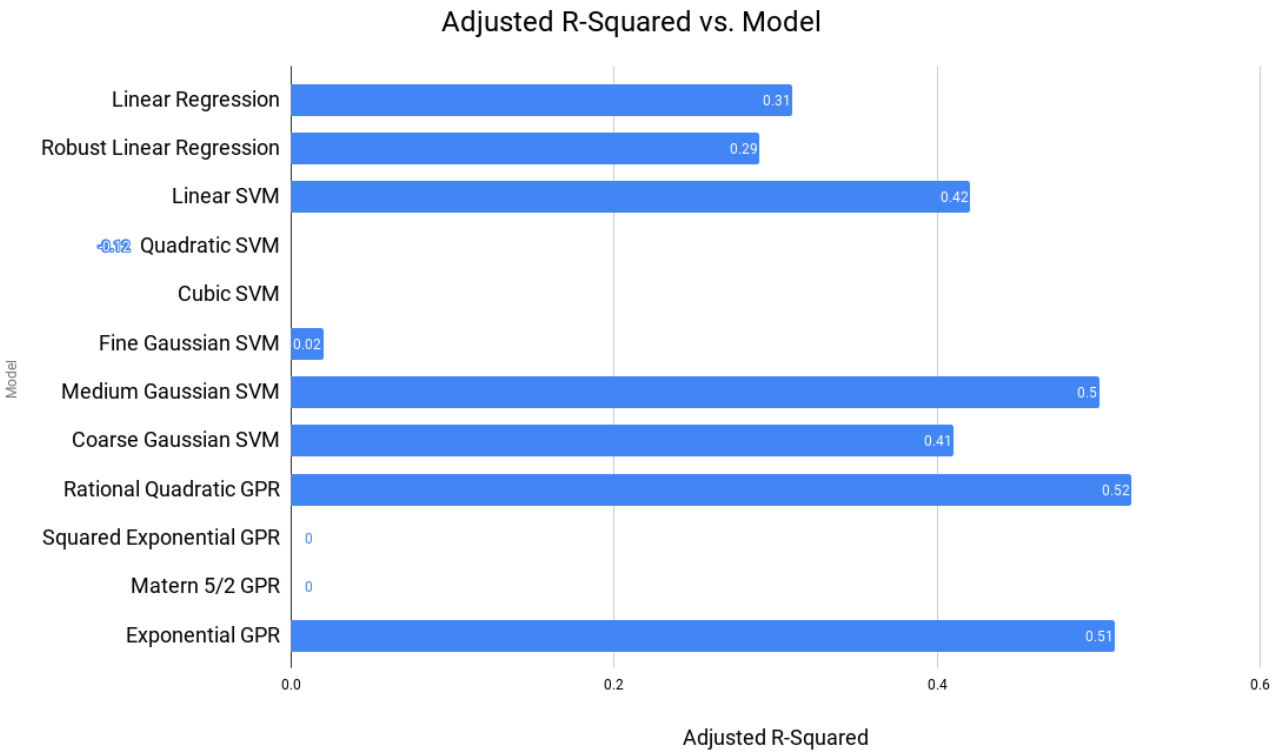

Figure 12 - Arousal Adjusted R-Squared Values per Model.

Artificial Neural Networks - Arousal

The same metrics were used for evaluating Artificial Neural Networks trained to predict Valence were also used for the Arousal dimension (Testing R). Table 9 presents the Training R, Testing R and Training Time for each ANN trained to predict Arousal.

Layer Size	Training R	Testing R	Training Time (sec)
1	0.76122	0.74589	25.972
2	0.81572	0.68425	60.742
3	0.85161	0.71352	189.435
4	0.86986	0.66473	165.764
5	0.88537	0.72067	411.887
6	0.9179	0.64151	345.083
7	0.93897	0.62759	772.312
8	0.93876	0.60181	1281.947
9	0.94762	0.62806	1528.596
10	0.94529	0.67519	1628.19

Table 9 - Arousal Models - Artificial Neural Networks

Once again, the tendency on the networks is to increase the Training R at the expense of the Testing R, for each neuron added. With two exceptions, every time a neuron is added, the training time increases by a significant margin.

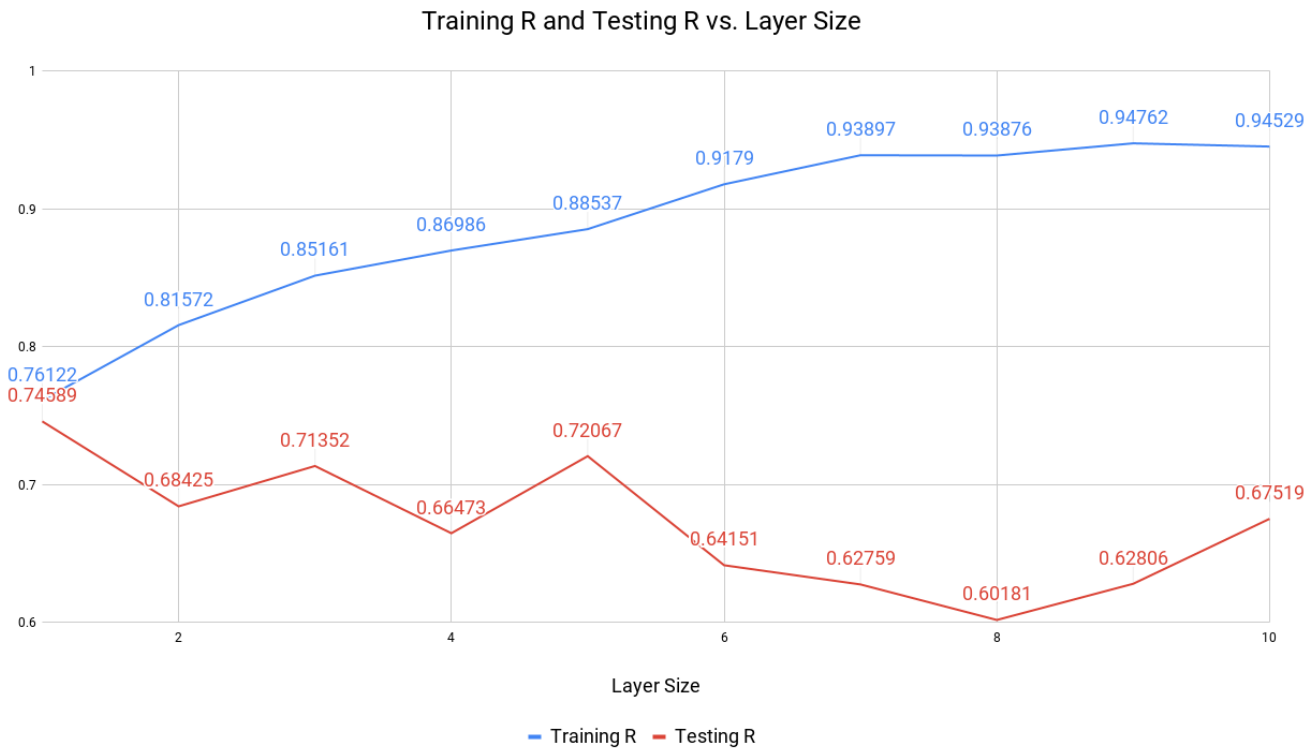

Figure 13 Results for Arousal using Artificial Neural Networks.

Model Comparisons

However, after training all the models, a method of comparison was lacking. It was not possible to directly compare Adjusted R^2 from the Linear Regressions, Support Vector Machines and Gaussian Process Regression to Testing R from Artificial Neural Networks. As such, the prediction output of each model was compared to the original values of the Testing set, with the objective of calculating a Correlation Coefficient. This would serve as the metric of comparison between all the model groups and the final models to utilize would be selected based of this value.

Correlation Coefficients - Valence

From the previously presented results, the top Valence models were the Rational Quadratic GPR, Matern 5/2 GPR, Exponential GPR and Medium Gaussian SVM. This should hold true for the correlation coefficients. Regarding ANNs, presented results suggest that a lower hidden layer size produces better results. Table 10

presents the obtained Correlation Coefficients, MSE and Training Time of each model for Valence.

Model	Correlation Coefficient	MSE	Training Time (sec)
ANN - 1 Neuron	0.6993	0.727	4.146
ANN - 2 Neurons	0.7087	0.7085	83.452
ANN - 3 Neurons	**0.7288**	**0.6766**	**215.919**
ANN - 4 Neurons	0.7129	0.7152	470.343
ANN - 5 Neurons	0.6970	0.7667	382.383
ANN - 6 Neurons	0.6961	0.7922	436.584
ANN - 7 Neurons	0.6632	0.9588	771.228
ANN - 8 Neurons	**0.7221**	**0.6947**	**1152.84**
ANN - 9 Neurons	0.5508	1.7321	1336
ANN - 10 Neurons	0.6304	0.9872	1598.852
Linear Regression	0.6838	0.8006	3.2837
Robust Linear Regression	0.6886	0.7949	2.9825
Linear SVM	0.6960	0.7431	2.9856
Quadratic SVM	0.3567	3.0613	2.5201
Cubic SVM	0.2347	20.2417	2.448
Fine Gaussian SVM	0.1410	1.3843	0.68312
Medium Gaussian SVM	0.7089	0.6998	0.8373
Coarse Gaussian SVM	0.6875	0.7484	0.65381
Rational Quadratic GPR	**0.7224**	**0.6756**	**28.132**
Squared Exponential GPR	-0.0146	1.4117	42.946
Matern 5/2 GPR	-0.0146	1.4117	20.792
Exponential GPR	**0.7340**	**0.6536**	**19.961**

Table 10 - Valence results Bold text indicates the four best models.

The top 20% of the trained models for predicting Valence, by Correlation Coefficient order, were the Exponential GPR, ANN – 3 Neurons, Rational Quadratic GPR and ANN – 8 Neurons. This correlates to the previously presented Adjusted R^2 values of the Exponential GPR and Rational Quadratic GPR models, in which the two were showed the best results. Regarding the Artificial Neural Networks, the 3 Neurons ANN presented the best Testing R for Valence, as well as the 8 Neurons ANN being one of the top results for this group of models. After uncovering the top 20% models, the **Exponential GPR** model was chosen for having the lowest training time from the top 20%.

Correlation Coefficients - Arousal

Previous regression results suggest that Rational Quadratic GPR, Exponential GPR, Medium Gaussian SVM and Linear SVM are the most accurate in predicting Arousal. Regarding ANNs, the three best models resulted from hidden layer sizes of one, three and five neurons.

Model	Correlation Coefficient	MSE	Training Time (sec)
ANN - 1 Neuron	0.7463	0.7515	25.972
ANN - 2 Neurons	0.7484	0.7451	60.742
ANN - 3 Neurons	**0.7641**	**0.7142**	**189.435**
ANN - 4 Neurons	**0.7751**	**0.6945**	**165.764**
ANN - 5 Neurons	0.7564	0.7395	411.887
ANN - 6 Neurons	0.7190	0.8836	345.083
ANN - 7 Neurons	0.6462	1.2402	772.312
ANN - 8 Neurons	0.7079	0.9608	1281.947
ANN - 9 Neurons	0.7180	0.9204	1528.596
ANN - 10 Neurons	0.6741	1.0981	1628.19
Linear Regression	0.7033	0.896	3.2677
Robust Linear Regression	0.7050	0.8961	2.166
Linear SVM	0.7148	0.8201	2.5042
Quadratic SVM	0.6253	1.3374	1.5445
Cubic SVM	0.1770	24.0384	1.4488
Fine Gaussian SVM	0.2902	1.5159	0.42306
Medium Gaussian SVM	**0.7571**	**0.7067**	**0.46666**
Coarse Gaussian SVM	0.6837	0.8778	0.38189
Rational Quadratic GPR	**0.7644**	**0.6991**	**18.322**
Squared Exponential GPR	0.0319	1.639	24.534
Matern 5/2 GPR	0.0321	1.639	13.605
Exponential GPR	0.7526	0.7288	10.324

Table 11 - Arousal results Bold text indicates the four best models.

Following the same pipeline for Arousal, the best 20% of the trained models by Correlation Coefficient were the ANN – 4 Neurons, Rational Quadratic GPR, ANN – 3 Neurons and Medium Gaussian SVM. Again, the top non-neural Network models also showed some of the best results regarding the Adjusted R^2 values, with the best result belonging to Rational Quadratic GPR and the third to Medium Gaussian SVM. Regarding the Artificial Neural Networks, the 3 and 4 Neuron ANNs presented some of the best relationships between Training and Testing R.

After uncovering the top models for the Arousal dimension, the model with the lowest Training Time was chosen, the **Medium Gaussian SVM**.

Chapter 4: Implementation

Throughout this chapter the implementation process will be discussed. This includes the decision process and details of the implemented functionalities, description of the resulting solution and limitations associated.

Goal

The first established goal for this work was to provide a way for any game developer/programmer to choose the music and/or sound for their game, based on the emotional content of the piece. As previously mentioned, the Circumplex Model serves as an abstract two-dimensional representation of emotions in the form of Valence in the horizontal axis, and Arousal in the vertical axis. Taking this approach into consideration, several models were built to attribute and represent pairs of values to new music and sounds. Furthermore, we assessed from the literature review that musical parameters such as Pitch and Tempo can affect the perceived and interpreted emotion on the listener. As such, another objective was to allow the modification of those features in real time, following the guidelines found during the research preceding the implementation of this work.

Requirements

To achieve the previously defined goal, the developed tool will need to meet certain requirements, both Functional and Non-Functional. The Functional requirements focus on what the tool should be able to perform, more specifically audio attribution and modification in real-time. These are shown in Table 13.

Requirement Type	**Requirement**
Functional	Represent Emotion Points (Circumplex) in Real-Time
	Assign music/sound to a GameObject
	Definition of audio source parameters in Real-time (3D Sound, Spatialization, Attenuation, Trigger Type,…)
	Change of sound parameters in Real-time (Pitch, Tempo, Volume)
	Allow user imported files
	Save/Load Capability
Non-Functional	Easy to use
	Light
	Extendible Library of Sounds and Music

Table 12 - Functional and Non-Functional Requirements for Implementation.

Software Used

MATLAB was used for processing and building models, and to build a standalone executable file to process CSV files that contain information on a WAV file and output the correspondent Valence and Arousal values. *Unity3D* was the engine

chosen for the development of the proposed tool. It was chosen for being a standard with fellow researchers, easy to port to other platform, and its increasing popularity. Some of the most recent popular games developed in Unity3D include *Pokémon GO* for mobile platforms, *Hearthstone* for mobile and desktop platforms, and *Rick and Morty: Virtual Rick-ality* for Virtual-Reality. The version utilized to develop this tool was the *Unity 2017.1.0.p4* and the programming language used was C#.

As part of the requirements listed in Table 12, the change of musical parameters in real-time is an important part of the end goal. For this, *FMOD* [36] was chosen to complement *Unity3D*'s sound handling capabilities. The *FMOD* plug-in requires an existing *FMODStudio* project to be indicated. However, this tool does not require a project, resulting in the creating and usage of an empty *FMODStudio* project. A more detailed Table containing the functionalities of both is provided by Dani Kogan. The *FMOD* version used was the *FMODStudio 1.10* for *Unity3D* [37].

MATLAB Standalone Executable

A standalone executable was built using MATLAB. To build it, the wizard for "Application Compiler" was used and compiled for C language. The runtime component necessary for the execution is included in the resulting package. This executable prompts the user to select a CSV file and, assuming the corresponding WAV file is on the same directory and has the same name, renames the WAV file to make it ready to import into the *Unity3D* project. The resulting filename is composed of three identifying fields, Valence and its value, Arousal and its value, and original filename. These three fields are separated by underscores ("_"), and the values for Valence and Arousal have their decimal part separated from the integer part by a dollar sign ("$") to facilitate the parsing process in *Unity3D*, as in:

VALvalue_**AR**value_Original Filename.wav

An example of the process can be seen in Figure 13, with the respective input and output.

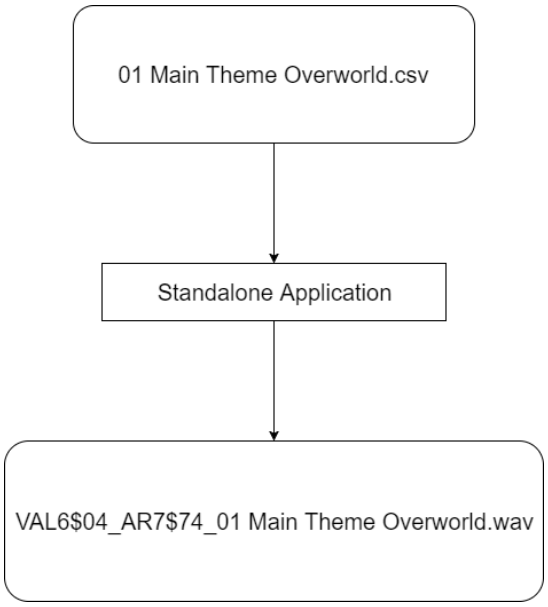

Figure 14 - Naming Flowchart.

Unity3D Implementation

The output for this part of the implementation process was idealized as a plug-in for *Unity3D*. It should be capable of being imported into scenes with ease and attribute sound based on its emotional content to objects presented inside the game, both while designing the game and after its deployment. This would effectively allow an individualized sonification for each player. While the focus lies in choosing according to the emotional content, it should also support all the audio operation that are provided with *Unity3D,* in a way that sonification can be performed under any circumstance.

The three milestones in the following sections are the representation of emotion utilizing the Circumplex Model, the change of audio parameters in real-time according to results found in literature, and the integration of the resulting tool in a generic scene.

Emotion Representation

As this work focuses on the selection of music and sound based on emotional content, a way to provide that information to the user was necessary. As such, a Circumplex Model representation was included in the *Graphical User Interface* and is accessible for the audio selection, as can be seen in Figure 15. This allows the user to choose an audio clip based on a target emotion, allowing Sonification based on the emotional content intended.

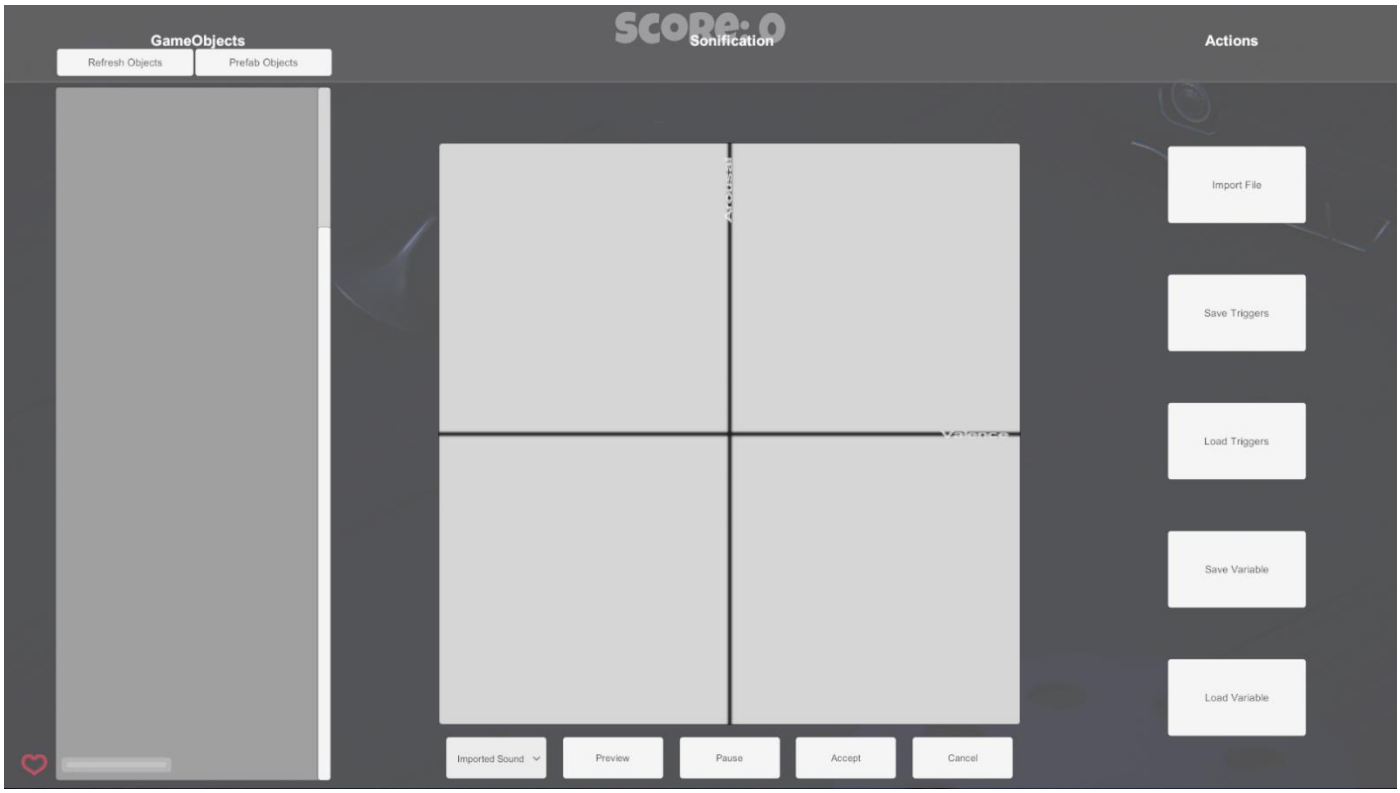

Figure 15 Representation of the Circumplex Model.

Music Parameter Change

This tool allows for a change in the musical parameters of the chosen music. The alteration of said parameters was also extended to include sound changes. This is solely done by manipulating the parameters using *FMOD*.

Based on the previous work by Gomez et al. in [24], it is suggested that sound features can be changed to achieve a certain location on the Circumplex Model. *FMOD* provides audio tools that enable the change of several features, making use of the proposed equations for the features. As presented on Chapter 2 (see Table 3), features were distinguished and noted according to the emotion the listeners reported. However, these results do not differentiate the pure musical features resulting from the composition of a piece itself, interpretation features (such as expression of the interpreter) and physical sound features. Moreover, it is the interest of this thesis to find the possible parameters to change for a dynamic Sonification. To serve this purpose, the findings were separated into Compositional Features, Interpretation Features and Changeable Features. Compositional Features are the characteristics that are decided upon the creation of the musical piece, such as the Mode and progression. Interpretation features are the ones that are associated with the way the piece is interpreted, affected by the attack on the notes and note articulation. Changeable Features are the ones that can be modified without directly modifying the audio file or specifics parts of it.

Compositional Features	Interpretation Features	Changeable Features
Major Mode	Rising micro intonation	Fast Tempo
Simple and consonant harmony	Raised singer's formant	Small Tempo Variability
Perfect 4th and 5th intervals	Staccato articulation	Medium-high sound level
Smooth and fluent rhythm	Large articulation variability	Small sound level variability
Micro-structural regularity	Fast attacks	High Pitch
	Small timing variability	Much Pitch Variability
	Sharp contrasts between 'long' and 'short' notes	Wide Pitch Range
	Medium-fast vibrato rate	Ascending Pitch
	Medium vibrato extent	Bright timbre

Table 13 - Separation of the musical features of Happiness into the four columns proposed.

Looking at the Changeable Features column in Table 13, it is possible to group the values into four high-level feature groups: Tempo, Sound Intensity, Pitch and Timbre. Out of these, the chosen features were Tempo, Sound Intensity and Pitch for their objectiveness and the operation associated with it being standard. As previously mentioned, these can be altered in real-time, do not depend on the audio clip and are universal.

$$Sound\ Intensity = 58{,}40 + 0{,}5 \times valence + 2{,}14 \times arousal - 0{,}18 \times valence \times arousal \quad (5)$$

$$Tempo = 1{,}2 + 0{,}1 \times valence + 0{,}43 \times arousal \quad (6)$$

$$Pitch = 4{,}54 + 0{,}05 \times valence - 0{,}07 \times arousal \quad (7)$$

However, these equations result in unclearly defined values, missing a proper scale and a way to compare and differentiate similar music. Tempo classification ranged from Slow to Fast, and Pitch from Low to High. The exception was the Sound Intensity, being labelled as decibels. Since the values of Valence and Arousal are already know from the work presented in Chapter 3, the use for these equations changed from a deterministic one to a

if the feature values matched perfectly. This is done by calculating how much a feature value would be in the initial point, with its original Valence and Arousal, and how much a feature value would be in the final position, the position clicked by the user during the selection process. Afterwards, a simple division of the end values by the initial values yields a coefficient, a percentage of change. This change can be applied to the original value of the feature, since every feature has 1 as a start value. If the change is less than 3% in any of the features, the change for that feature is ignored. The process is illustrated in Figure 16.

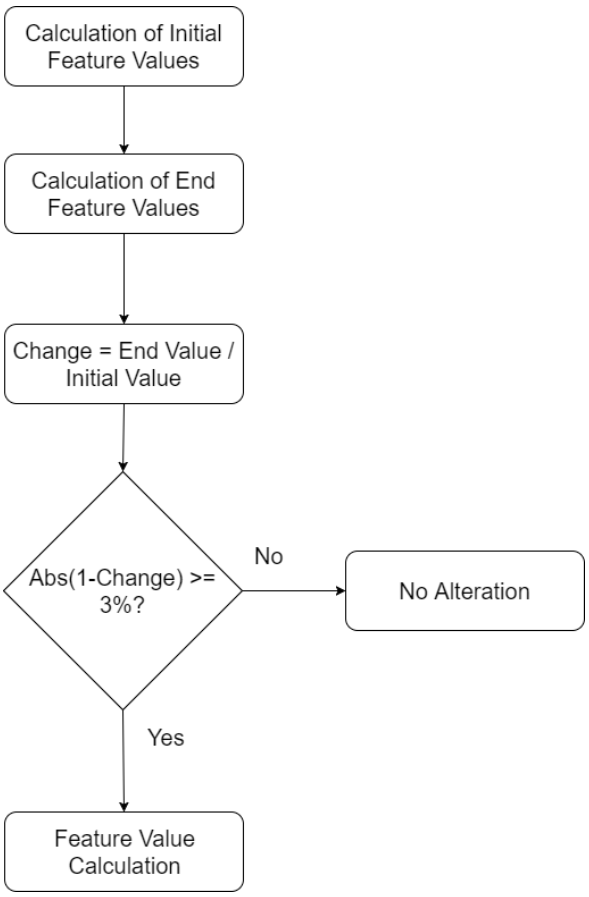

Figure 16 - Feature Change Flowchart.

However, only Sound Intensity and Tempo are represented by linear scales. Pitch is defined by an exponential scale where a decrease in octave results in halving the current frequency, while an increase results in doubling the frequency. For this purpose, it is important to understand how many half-step intervals compose an octave and how these are calculated. The calculation of the coefficient for the determination relative to the change is given by the expression bellow.

$$coeff = 2^{\frac{Half-steps\ of\ Tempo\ Change}{12}} \quad (8)$$

Should the frequency be higher, the coefficient is multiplied by the base value; if lower the base value is divided by it. Applying this information to the standard A at 440Hz, Table 14 is obtained, representing the frequencies of the intervals both and octave up and down.

Note	Interval (Half-steps)	Frequency (Hz)
A	-12	220
A# (Bb)	-11	233,0818808
B	-10	246,9416506
C	-9	261,6255653
C# (Db)	-8	277,182631
D	-7	293,6647679
D# (Eb)	-6	311,1269837
E	-5	329,6275569
F	-4	349,2282314
F# (Gb)	-3	369,9944227
G	-2	391,995436
G# (Ab)	-1	415,3046976
A	0	440
A# (Bb)	1	466,1637615
B	2	493,8833013
C	3	523,2511306
C# (Db)	4	554,365262
D	5	587,3295358
D# (Eb)	6	622,2539674
E	7	659,2551138
F	8	698,4564629
F# (Gb)	9	739,9888454
G	10	783,990872
G# (Ab)	11	830,6093952
A	12	880

Table 14 - Note Frequency Distribution and relative Half-Steps.

A Tempo change in *FMOD* also results in a frequency change, speeding up or slowing down the played file by the factor established. However, the change in the Tempo parameter needed cannot affect the Pitch. As such, a Pitch Shift effect is applied in real-time, having as objective the compensation of the altered Pitch resulted from the Tempo change. The value for this effect is calculated in the same way as for the Pitch. For example, an increase of 20% starting from the A (440Hz) would lead to a frequency of 528Hz. To compensate this, the Pitch Shift would need to be at 83.3% of the base value. Inversely, if the Tempo would slow down 20%, the Pitch Shift would need to be at 125% of the base value. Figure 17 illustrates the process described.

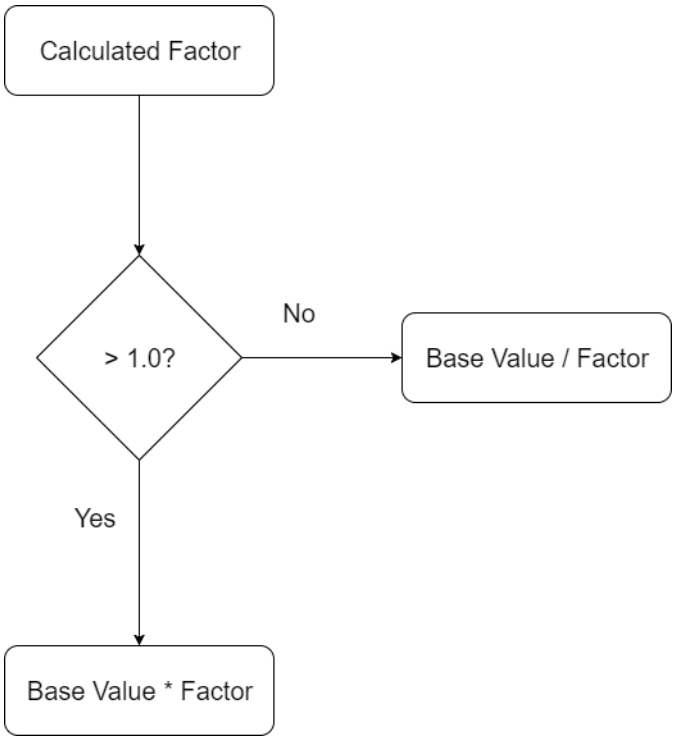

Figure 17 - Pitch Compensation Flowchart

Game Integration

The tool developed requires only the import of the package containing two *Unity3D* prefabs with the *GameObjects*, the scripts associated and ready to use. By simply dragging these to the *Scene Hierarchy,* the game will be ready to perform sonification.

The *GUI* (Figure 18) presents the user with a menu containing sonification options, action Buttons, and a list area to select the desired objects from and attribute sound or music. This list is filled when the button “Refresh Objects” is pressed, displaying *GameObjects* that possess a *Render* Component (can be made visible in the scene), are part of any *UI*, and do not possess a *Particle System* Component. The latter is due to the conflict with rendering meshes in *Unity3D*, with both not being supported at the same time. The main button also displays the current sonification settings chosen by the user. If none was set, default values are presented. This list can also be used to perform sonification with pre-build game objects in the project (Prefabs), by pressing the button “Prefab Objects”.

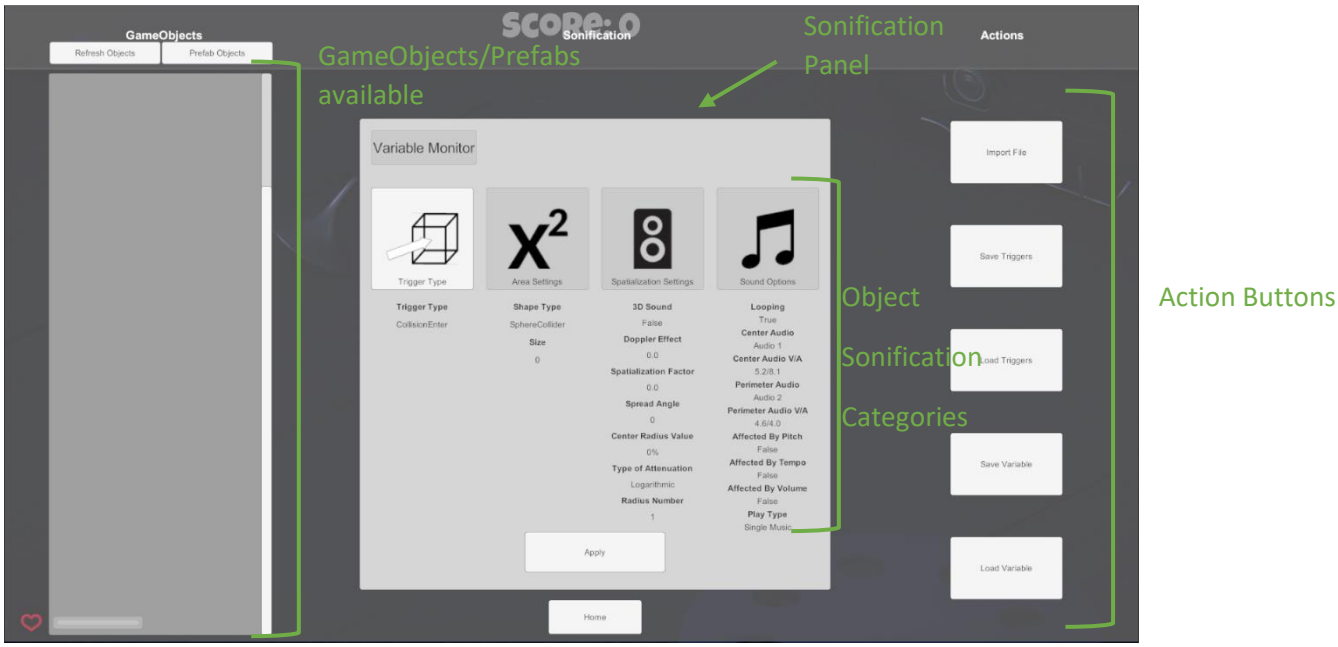

Figure 18 - Graphical User Interface in Unity3D

The scene hierarchy is visible in Figure 19, with the two objects highlighted being the required ones for the functioning of the tool. The *MusicPanel* object contains all the display settings and *GUI*, while the *PlaybackController* object possesses the models for feature altering and the calls for the *FMOD* plug-in.

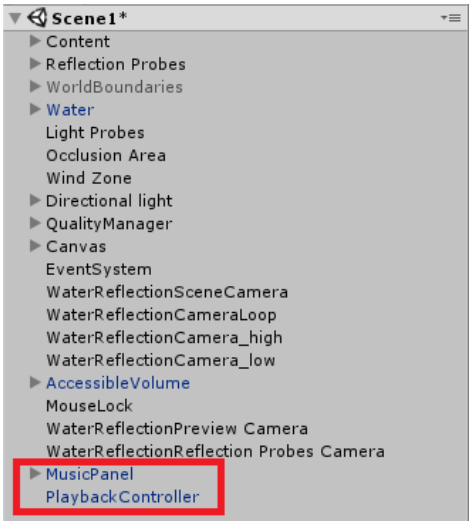

Figure 19 - Unity3D Scene Hierarchy (red box highlights the developed prefabs).

Object Sonification

After the selection of one or more objects to attribute audio to, the user is prompted to customize several options related to the attribution of sound. The objects affected by this Sonification option can be either objects already present

in the scene, or previously built objects (Prefabs) that are instantiated during runtime. The Object Sonification options are separated into four categories and count with the choosing of the playing trigger, the area associated and several other options. The summary of the options is available in Table 15, with the options presented in each category, and with the values that can be inputted or selected. These are further explained on the respecting menu section. Each menu section contains a screenshot a brief description of the menu, a table indicating the options available and a brief explanation.

Category	**Option**		**Option Values**
Trigger Type	Trigger Type		Collision Enter
			Collision Exit
			Distance
			Probability
Area Settings	Collider Type		BoxCollider
			SphereCollider
			CapsuleCollider
	Collider Size		Depends on the Collider
Spatialization Settings	3D Sound		True/False
	Doppler Effect		Decimal Number (0 - 5)
	Spatialization Factor		Decimal Number (0 - 1)
	Spread Angle		Decimal Number (0 - 360)
	Center Radius Value		Decimal Number (0 – 100)
	Type of Attenuation		Logarithmic
			Linear
			None
	Radius Number		Integer (1 – 2)
Sound Options	Looping		Boolean
	Center Audio	Audio Select	WAV File
		Valence	Decimal Number (1 - 9)
		Arousal	Decimal Number (1 - 9)
	Perimeter Audio	Audio Select	WAV File
		Valence	Decimal Number (1 - 9)
		Arousal	Decimal Number (1 - 9)
	Feature Changes	Pitch Affected	Boolean
		Tempo Affected	Boolean
		Volume Affected	Boolean
	Play Type		Single Music
			Single Adapted Music
			Multiple Music
			Multiple Adapted Music

Table 15 - Option Menu parameters.

Trigger Types

By selecting the Trigger Type options on the main menu, the user is prompted with four buttons, each representing a different trigger type. The four trigger types available are the CollisionEnter, CollisionExit, Distance and Probability. Table 16 contains a brief description for each trigger type.

Trigger Type	Effect
CollisionEnter (1)	The selected audio is triggered whenever the player enters in contact with the added collider on the target object(s).
CollisionExit (2)	The selected audio is triggered whenever the player exits the area defined by the collider.
Distance (3)	The selected audio will play while inside the area defined by the collider size and the inner radius associated.
Probability (4)	Acts as a CollisionEnter trigger, with a set chance of activating.

Table 16 - Trigger Type Effect Descriptions.

The triggers presented give freedom to the users in the sense of allowing them to choose how any given piece of audio will start playing. For example, upon entering a room the user wants to hear a phone ringing. This is made possible using the trigger type CollisionEnter, starting the audio reproduction as soon as the player enters the defined area for said room. Another scenario would be hearing wind effects when a player leaves an area, prompting the user to select a trigger type of CollisionExit. Distance triggers are useful for a continuous sound, that plays while the player is inside a defined area. The menu for this selection is presented in Figure 20.

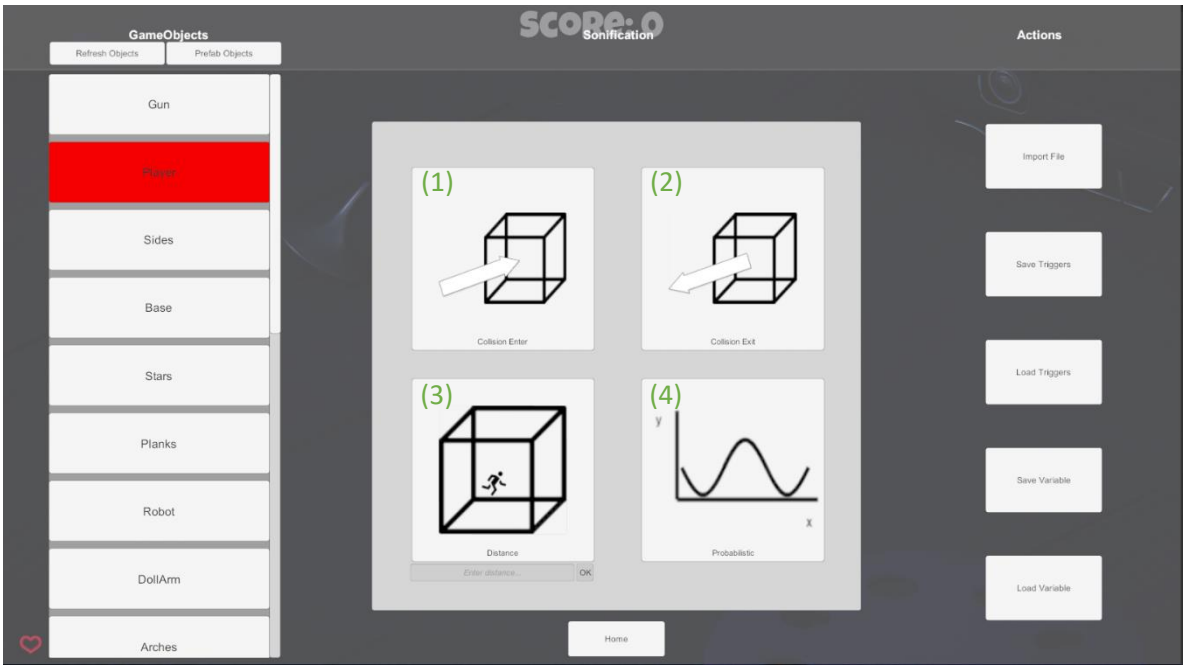

Figure 20 - Trigger Type Menu. (1) and (2) define entry and exit points for triggering the audio; (3) defines the radius of audio playing; and (4) allows specifying the probability of activation of the audio. See Table 16 for further information.

Area Settings

On this menu the user is presented with three buttons, each representing a collider shape that can be associated to the object(s) selected. Each collider represents a different area to be associated to an object. After the user presses one of the buttons, input fields for the dimensions of the selected collider will

appear. Depending on the collider choice, the inputs options vary as per Table 17.

Collider Type	Dimensions	Effect
SphereCollider (5)	Radius	Defines the radius of the SphereCollider
BoxCollider (6)	Size X	Size along the X axis
	Size Y	Size along the Y axis
	Size Z	Size along the Z axis
CapsuleCollider (7)	Direction	Alignment along the axis (X, Y or Z)
	Height	Defines the height of the CapsuleCollider
	Radius	Defines the radius of the CapsuleCollider

Table 17 - Area Settings Description.

Figure 21 shows the Area Settings Menu, with the SphereCollider chosen and the input field for its radius size on the lower right corner of the sonification panel. Once the user has entered a radius value, pressing the *Apply* button will apply both the shape and size of the area.

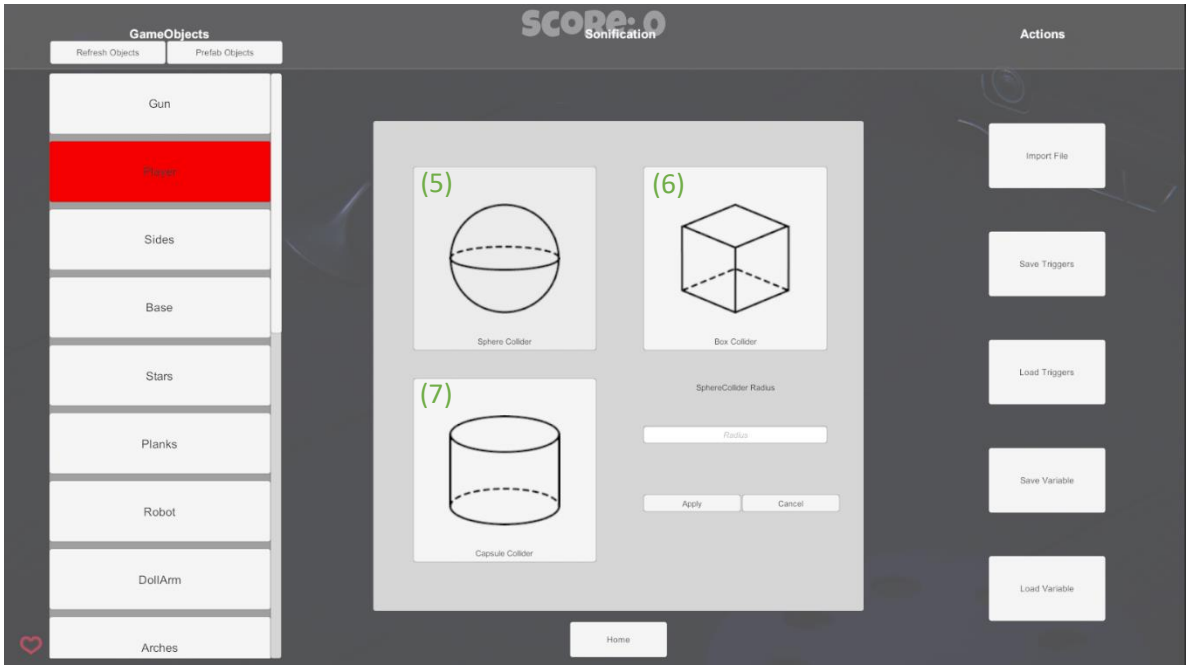

Figure 21 - Area Settings Menu. (5) defines the sonification area as a sphere, while (6) and (7) define the area as a box and a capsule, respectively. See Table 17 for further information.

Spatialization Settings

In this menu the user can set the values for 3D sound and other audio options related to distance and diffusion of sound. Enabling 3D settings will route the sound through *Unity3D*'s 3D sound engine, panning and volume attenuation according to the player's position according to the target game object(s). The Spatialization Factor dictates how much the audio is affected by the 3D sound

engine effects mentioned above. Spread Angle sets in which direction the clip is played. Table 18 presents more information on the effects of each setting.

Setting	Value	Effect
3D (8)	True/False	Sets the value for the 3D sound engine
Doppler Effect (9)	Decimal Number (0 – 5)	Doppler Effect Level
Spatialization Factor (10)	Decimal Number (0 – 1)	Affects the attenuation and panning on a 3D sound. 0 makes the sound fully 2D and 1 makes the sound fully 3D
Spread Angle (11)	Decimal Number (0 – 360)	Degrees of the spread of the audio in a 3D setting from the source
Center Radius Value (12)	Decimal Number (0 – 100)	Percentage of the size of the collider to define the center radius. Only available if the radius number equals to 2
Type of Attenuation (13)	Logarithmic	Attenuation curve for Volume over Distance
	Linear	
	None	
Radius Number (14)	Integer (1 – 2)	Number of radius considered for the object

Table 18 - Spatialization Settings Option Description.

By pressing the button that represents two radiuses, the user can then input a value for the inner radius. The value inputted corresponds to a percentage of the size selected on the Area Settings menu for the collider. For example, with a center radius value of 50 and a SphereCollider with size equal 10, the center radius would have a value of 5. Pressing the button corresponding to the Radius Number of one will setting the center radius equal to 1, the minimum distance required by *Unity3D* to manipulate 3D sounds.

Should the Type of Attenuation be set to *None*, the volume will not decay over distance, but the panning will still shift according to the player's position relative to the sound source.

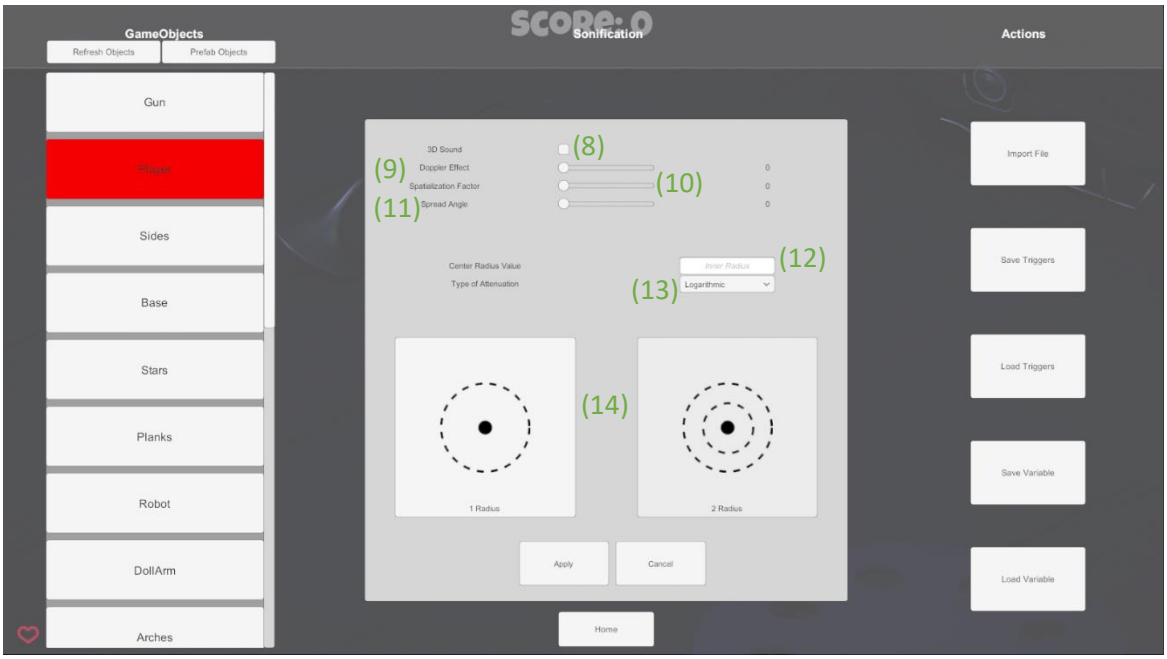

Figure 22 - Spatialization Settings Menu. (8) through (12) represent general sound option for 3D environments; (12) defines the center radius size based on the collider size; (13) presents the type of Attenuation for volume over distance; (14) represents the radius number buttons. For further information see Table 18.

Sound Options

Here the user can choose the audio files to associate to the defined center and perimeter, set the looping flag and adjust Valence and Arousal levels for an audio clip. It is also possible, depending on previous choices, to customize the playing behavior.

Setting	**Value**	**Effect**
Looping? (15)	True/False	Sets if the audio will repeat once it ends
Audio File (16)	Chosen from the Circumplex	Audio clip to play
Audio Valence (17)	Decimal Number (0 – 9)	Valence of the chosen audio clip. Can be modified
Audio Arousal (18)	Decimal Number (0 – 9)	Arousal of the chosen audio clip. Can be modified
Pitch Affected (19)	True/False	If true, the audio clip(s) will be affected by the Pitch models presented
Tempo Affected (20)	True/False	If true, the audio clip(s) will be affected by the Tempo models presented
Volume Affected (21)	True/False	If true, the audio clip(s) will be affected by the Volume models presented
Single Music (22)	True/False	Controller will only play the selected perimeter audio
Single Adapted Music (23)	True/False	Controller will play the selected perimeter audio with the choices from the affected flags

Multiple Music (24)	True/False	Controller will play multiple audio files between the center and perimeter audio, according to the player position
Multiple Adapted Music (25)	True/False	Controller will play multiple audio files between the center and perimeter audio, according to the player position. The played files are modified depending on the affected flags

Table 19 - Sound Settings Description.

Due to limitation on *FMOD* and *Unity3D*, the change of parameters with 3D sound was not implemented. As such, the options for the Pitch, Tempo and Volume affected are disabled when the 3D Sound flag is set to true on the Spatialization Menu. The same happens with the options Single Adapted Music and Multiple Adapted Music.

Regarding the type of play, the options are mutually exclusive, only allowing for one at any given time. As already mentioned, the 3D setting impacts the interaction, as well as the radius number. When the radius number is set to 1, the Multiple Music and Multiple Adapted Music are disabled.

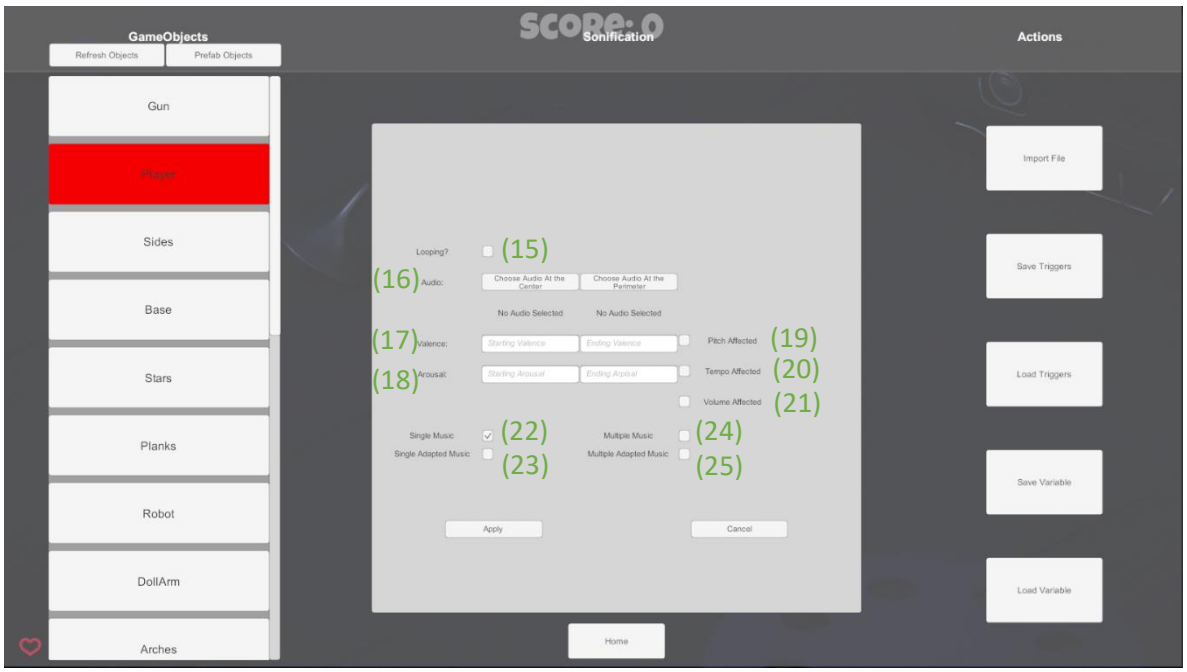

Figure 23 - Sound Options Menu. (15) stands for the looping option. (16) signals the buttons for choosing the audio clips, with the results being shown in (17) and (18) for Valence and Arousal, respectively. (19) through (21) consist of the feature alteration options. (22) through (25) define the behavior of audio playing. For further information see Table 19.

When the user presses any of the Audio buttons, a Circumplex representation is displayed with the available audio clips. These are organized by their respective Valence and Arousal values.

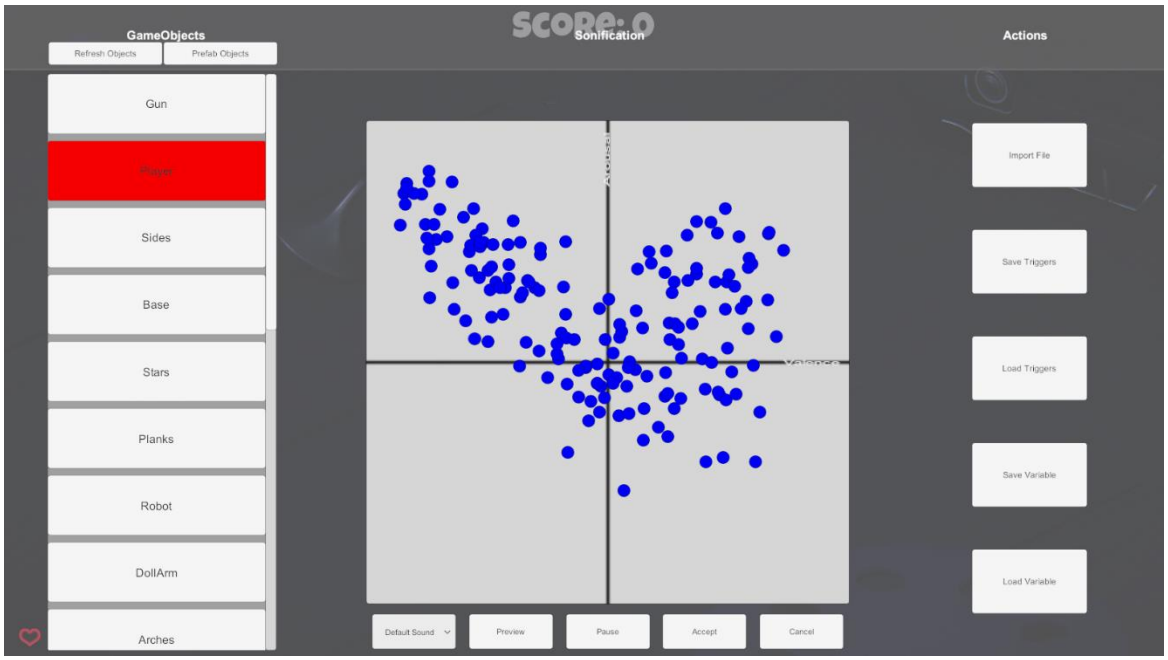

Figure 24 - Representation on the Circumplex Model.

While on the Circumplex representation, it is possible to choose the category of audio to play by using the dropdown menu on-screen. The available categories are *Default Music*, *Default Sound*, *Imported Music* and *Imported Sound*. Some audio clips are already provided for the *Default Music* and *Default Sound* categories, while the remaining are left blank for the user to import into. In particular, the clips provided for the previous are part of the IADS-2 dataset [38].

To choose an audio clip the user should press on any circle object presented, turning the chosen object to a yellow color. Whenever the user hovers the pointer on an object, it will display the respective name, Valence and Arousal. Furthermore, it is possible to preview the currently selected audio (*Preview* button) before assigning it (*Accept* button). The user can also choose to return to the Sound Options menu by pressing the *Cancel* button.

After the user is satisfied with the customization, the *Apply* button should be pressed. This action handles the instantiation of the collider on the object(s) selected and applies all the parameters. Once this is done, the user can press the key “F2” to enable the minimum and maximum distances as wireframes, as in Figure 25. While this feature is active, the *Renderer* component of the selected object(s) will also change colour to green for easy identification.

Figure 25 - Maximum and Minimum Wireframe Distances.

It is possible to save the progress at any time, creating a JSON file. This can be shared with other users, that simply must load the specified file. However, the last configuration will overwrite the previous settings.

Audio Changes

By assigning two different audio clips, for the inner and outer radius, the tool will establish a pathway from the outer clip to the inner clip. This is done by relating the distance in-game to the object by the listener and the Euclidean distance in the Circumplex Model of the two clips.

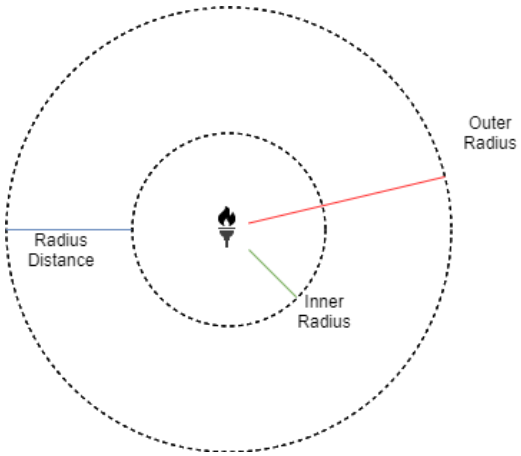

Figure 26 - Example of Inner Radius, Outer Radius and Radius Distance of a Torch GameObject.

When the player is very close to the Outer Radius, the audio played will be the one chosen on the Menu. However, when the audio reaches 90% of the total time and depending on the current distance to the object, a new audio clip will be played depending on the valence and arousal of the position. For example, when a clip reaches 90% and the listener is at 40% of the radius distance counting from the outer radius, a new Valence and Arousal pair of values is calculated according to a straight line in the circumplex model. Figures 27 and 28 illustrate this process.

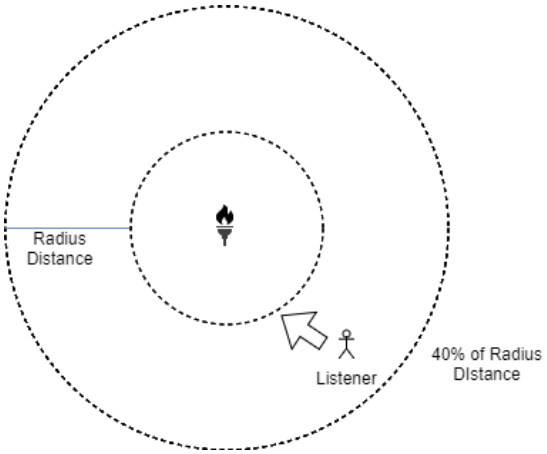

Figure 27 - Position of the Listener according to Radius Distance.

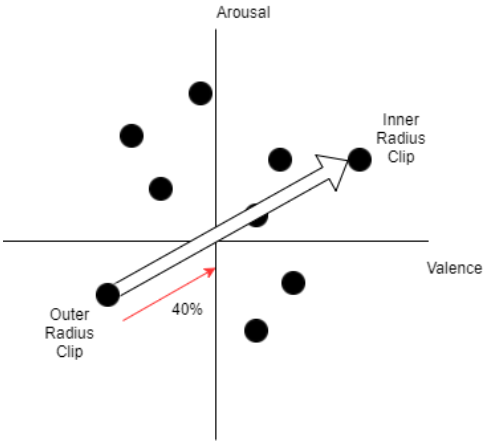

Figure 28 - Calculating new Valence and Arousal values according to the distance.

The new Valence and Arousal values are calculated by retrieving the inclination of the straight line going from the Outer Radius to the Inner radius and the Euclidean distance between these points. The Valence and Arousal Euclidean distance (γ) is given by the Pythagorean theorem.

$$\gamma = \sqrt{(Inner\ Valence\ -\ Outer\ Valence)^2 + (Inner\ Arousal\ -\ Outer\ Arousal)^2} \quad (9)$$

The inclination factor is always considered as the absolute value, since the relevant values are the angle and the distance, not the direction.

$$Inclination\ = \left|\frac{Inner\ Arousal - Outer\ Arousal}{Inner\ Valence - Outer\ Valence}\right| \quad (10)$$

However, the precise angle is needed for the calculation of how much both dimensions changed. Since the *Inclination* is always calculated as the absolute value, it will always represent a straight line with a positive slope. Then, all that is needed is to perform the reverse tangent and convert the value to radians.

$$\theta\ =\ \tan^{-1}(Inclination)\ \times \frac{\pi}{180} \quad (11)$$

Having the angle (θ) in radians, the exact percentage of the travelled length in relation to the inner and outer radius is needed. This also requires the calculation of the position according to the audio source.

$$Distance\ to\ Source =\ Inner\ Radius + \propto \times\ (Outer\ Radius\ -\ Inner\ Radius) \quad (12)$$

where $\propto$ represents the remaining distance to travel to reach the inner radius. Taking this into account, the coefficient that represents the travelled distance between both radius (β) is given by:

$$\beta\ =\ 1 - \propto \quad (13)$$

The last step consists on calculating the new Valence and Arousal values. These can be calculated by finding the variation on both dimensions, as in:

$$Valence\ Variation\ =\ \cos\theta\ \times\ \beta\ \times\ \gamma \quad (14)$$

$$Arousal\ Variation\ =\ \sin\theta\ \times\ \beta\ \times\ \gamma \quad (15)$$

Depending if the Valence and Arousal outer values are higher or lower than the inner values, these will be added or subtracted to the outer ones in each dimension. Afterwards, the closest audio clip to the point with the calculated

Valence and Arousal values will be chosen by Euclidean distance and played. This process is repeated until the player reaches the inner radius or leaves the area (outer radius).

Variable Sonification

Should the user want to perform sonification based on a target value instead of a specific game object, it is possible to do so from the Sonification Panel. By having exclusively one game object selected from the objects in the scene, the button “Variable Monitor” becomes available. The later enables the user to choose a variable from a component present in the selected game object (including custom classes built by a developer). The target variable must be of a public protection level, as well as of the type “float”. Any component that possesses a variable that respects the two mentioned conditions is listed on the Component dropdown option.

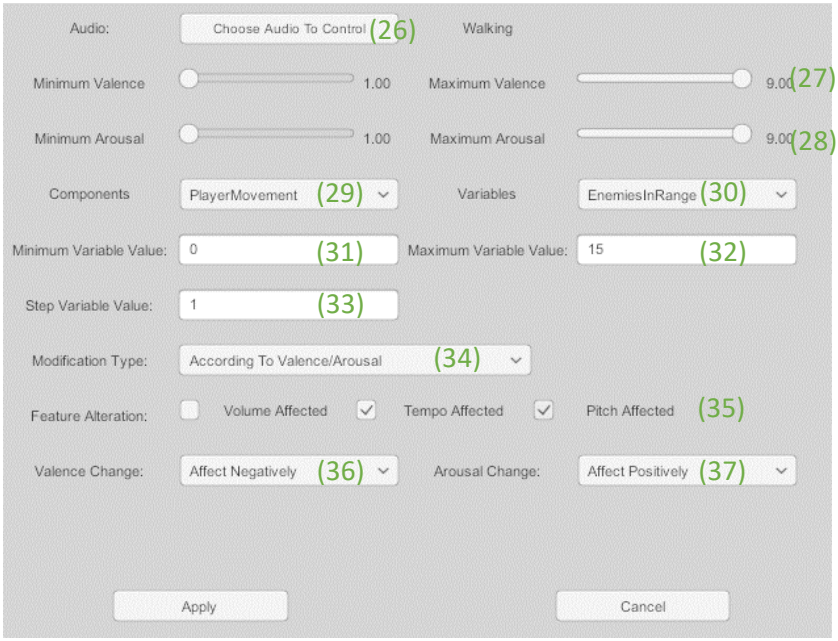

Figure 29 - Variable Monitor Menu. (26) reflects the choice of audio. (27) and (28) represent the ranges for the Valence and Arousal dimensions. (29) is the Component dropdown menu. (30) represents the Variable dropdown menu. (31) and (32) signal the minimum and maximum values for the variable. (33) represents step value for the variable. (34) is the Modification type dropdown. (35) consists of the audio features to change. (36) and (37) allow specification of the audio change in the Valence and Arousal dimensions. For further information see Table 20.

While on the screen presented in Figure 29, the user can select an audio file in the same fashion as presented with Object Sonification by pressing the button (26). Once an audio file is chosen, the user can specify the range of values the music will vary on each of the Circumplex dimensions by using (27) and (28). Once a variable from a component is chosen from the Variable dropdown menu, the user must indicate what are the values in which the audio changes will be applied. The parameters needed are the minimum and maximum values of the

variable, as well as its step. The later indicates how much the variable would need to vary for the application of a single audio change. Additionally, depending on the user's intentions, it is possible to vary the Volume, Tempo and Pitch according to the selected variable, or add an effect and change its impact in a linear fashion. The effects available in the second mode are Pitch Shift, Distortion, Tremolo, Tempo and Volume. The impact of each effect can be change by the respective slider.

Figure 30 - Variable Monitor Menu with custom effect change. (38) consists of the available effects. (39) displays the effect change value for each effect. For further information see Table 20.

Figure 30 shows the Variable Monitor Menu while the free effect changes are enabled. All the effects can be enabled at the same time without any restriction. This functionality was specifically developed for scenarios in which the user is already in possession of an audio file, and the Valence and Arousal of the later can be ignored. This functionality may be used to trigger effects for example in varying speed of a player, number of enemies in range, size, score, or other common variables present in games.

As with the Object Sonification options, it is also possible to save the configuration for a target variable, for loading at later stage. The resulting file is also a JSON file, however, due to the different file structure, it is saved separately from the Object Sonification settings. Another similarity lies on the last modification performed overwriting the previous.

Setting	Value	Effect
Audio File (26)	Chosen from the Circumplex	Audio clip to play
Allowed Valence Range (27)	1 – 9 (Both sliders)	Specifies the intervals in which the Valence of the song can vary
Allowed Arousal Range (28)	1 – 9 (Both sliders)	Specifies the intervals in which the Arousal of the song can vary
Component Dropdown (29)	Component from Selected Object	Selects the component from which to choose a variable
Variable Dropdown (30)	Variable from Selected Component	Selects a public “float” variable from chosen component
Minimum Variable Value (31)	Decimal Number	Sets the minimum value for the variable
Maximum Variable Value (32)	Decimal Number	Sets the maximum value for the variable
Variable Step Value (33)	Decimal Number	Variable difference required for one audio change
Modification Type Dropdown (34)	According to Valence/Arousal	Chooses whether the audio changes will be according to Valence and Arousal models or just incremental changes
	Just effects	
Feature Modifications Toggles (35)	True/False (for each)	Feature(s) to be affected by the changes
Valence Change Type Dropdown (36)	No Change	Affects how a change in the variable will affect the dimension
	Affect Positively	
	Affect Negatively	
Arousal Change Type Dropdown (37)	No Change	Affects how a change in the variable will affect the dimension
	Affect Positively	
	Affect Negatively	
Feature Modification Toggles 2 (38)	True/False (for each)	Feature(s) to be affected by the changes
Feature Modification Sliders (39)	According to individual effect	Step values for feature changes

Table 20 - Variable Monitor Settings Description.

Limitations

Size

Regarding the size of both implemented solutions, while both are light on their own, the added files needed make them relatively heavy. In case of the created MATLAB standalone, the inclusion of the run-time executable component pushes its size to about one Gigabyte. In the *Unity3D* implementation, it is dependent on the audio files it requires, especially costly since the format WAV itself is heavy.

Audio Formats

Unity3D can only load new clips as WAV files, limiting the format choice, also impacting the total size required. Although openSMILE and the produced MATLAB standalone work with WAV files, the end-user will need to make sure that the correct format is utilized, as well as the naming process referred in this chapter.

External Standalone Application

Having to use a standalone application for the prediction of Valence and Arousal might be disruptive, with users not willing to install additional components for its execution. This can have a negative impact in the sense that users might wrongly classify an audio clip.

Unity3D Parameter Change

While *Unity3D* allows for a variety of sound operations, it does not support the change of the playback speed without it affecting the pitch (only changing the frequency rate). This was the main reason for utilizing *FMOD*, since it allows the programmer to place several different effects in the audio clips during runtime.

FMOD 3D Sounds

Although *FMOD* is a more specialized plug-in, its lack of documentation and examples for the used C# language can be a discouraging reason for its usage. For 3D sounds, *Unity3D*'s native support turned to be more beneficial, due to its ease of use and handling of events such as updating the listener's position relative to the audio source automatic. This limitation, together with the previous, made it so that the modification of audio parameters on 3D sound instances are not possible.

(Left in blank intentionally)

Chapter 5: Case Scenarios

To demonstrate the functionality of the Unity3D tool developed during this work, two existing games developed in Unity were chosen. If the game had already audio, it was removed. The games were chosen from two different game categories: Third-Person Shooter and Arcade. After choosing the games, some objectives were defined as to what the developed tool should add to each game.

Install and Requirements

To use the proposed tool, the developer must import the *FMOD Unity* Integration package to the project. This package requires the path to an *FMOD* Studio project, but since this tool does not require so, it can be set to an empty *FMOD* Studio project. To do so, the developer must create a new project using *FMOD* Studio and use the Build option, as in Figure 31.

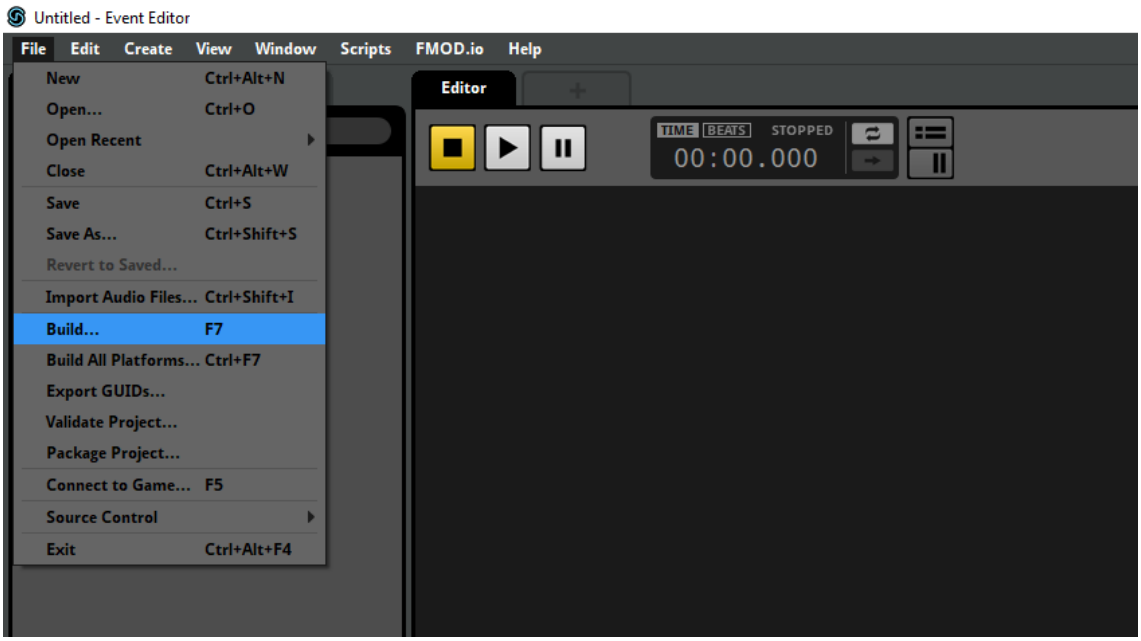

Figure 31 - Building FMOD Studio project.

Afterwards, the developer must import the package containing the proposed tool, containing the developed scripts, Prefabs and audio clips that compose it. Once this has been done, the developer needs only to drag the Prefabs *MusicPanel* and *PlaybackController* to the scene hierarchy and set the reference for main camera for the class “Music Chooser”, as well as the reference for the *PlaybackController* object. The location of the two instances of this class can be seen in Figure 32. Additionally, the developer should add an EventSystem to the scene hierarchy if one is not present.

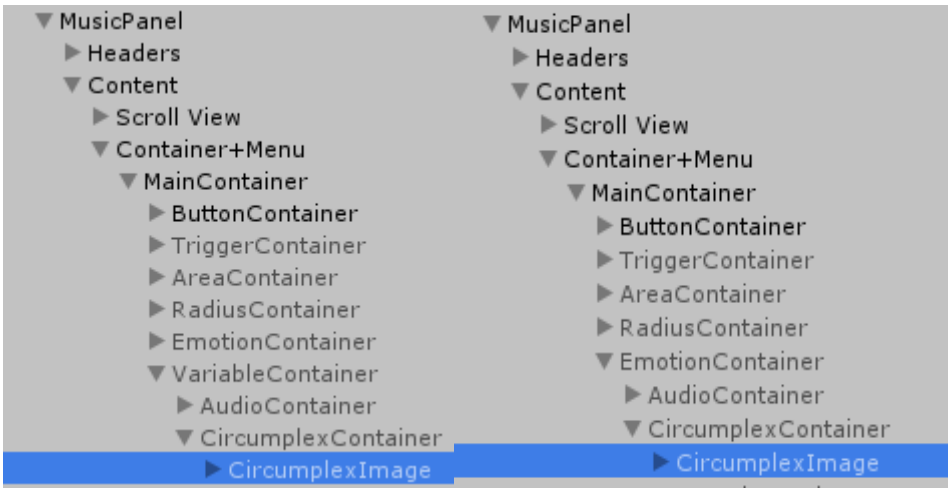

Figure 32 - Location of MusicChooser class instances.

The folders Default Music, Default Sound, Imported Music and Imported Sound are already included in the package. It is important to note that none of the steps described above need to be followed by the persons playing the game. Should all conditions be met, the tool is fully integrated into the game and ready to create custom Sonification.

Third-Person Shooter - Nightmare

For the Third-Person Shooter game, a game called *Nightmare* was selected. This game is a survival-type game where the player takes control of an avatar and faces enemies generated according to a time interval. A screenshot of the game can be seen in Figure 33.

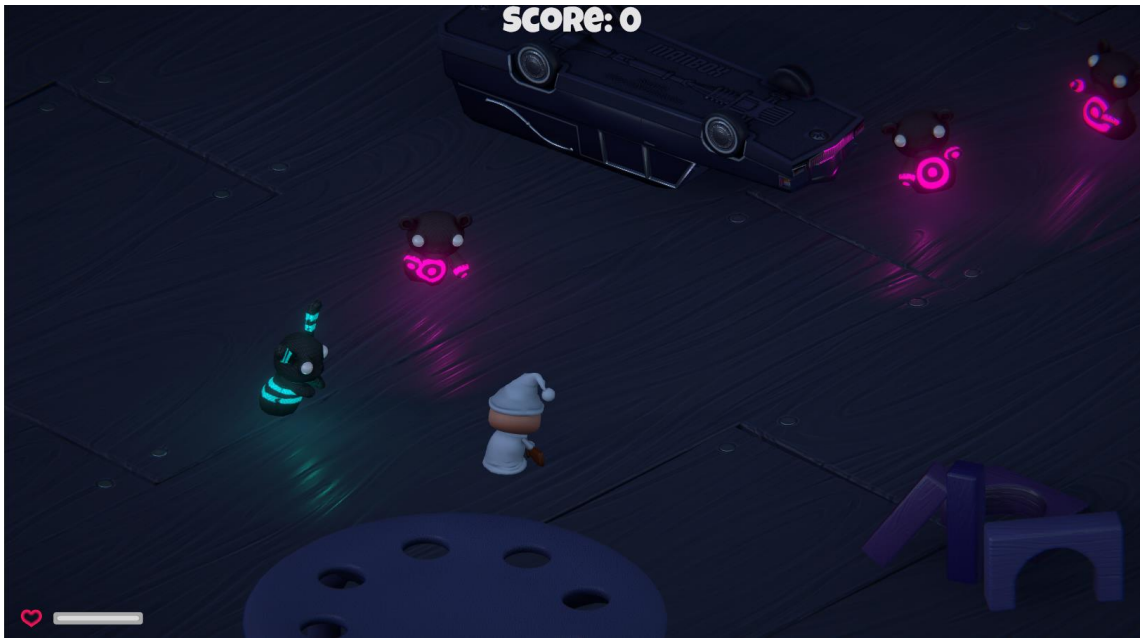

Figure 33 - Nightmare game screenshot.

Goal: The objectives established for this game consisted on (1) adding a 3D sound of footsteps with the enemies as a source of the audio, (2) increasing the Arousal of the background music according to the number of enemies in a certain range of the player, and (3) adding a warning sound when the player hits an object.

To accomplish these goals, the user must enable the Sonification Menu, by using the F1 key. Once the menu is presented, the user must press the button *Prefab Objects* to bring up the Prefab objects in the current project. The Prefab objects that represent the enemies in this game are called “ZomBunny”, “ZomBear” and “Hellephant”. Figure 34 depicts the main Sonification menu with the enemy Prefabs selected from the object list.

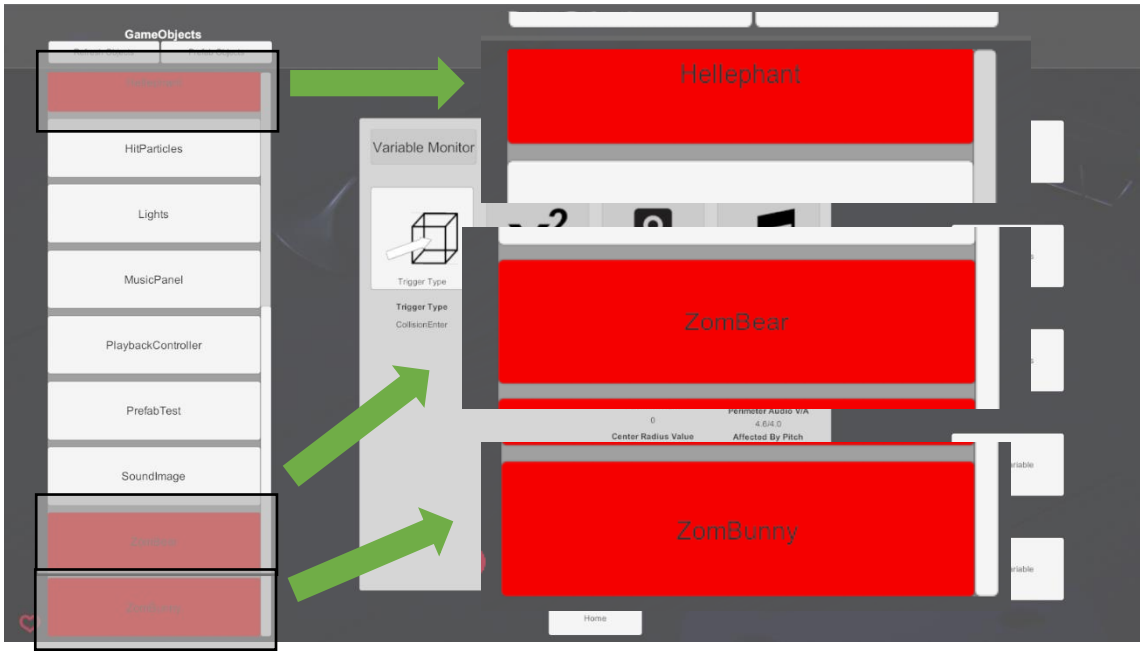

Figure 34 - Selection of enemy Prefab Objects.

Once all enemy Prefabs are selected, the user must choose the desired trigger for the audio. For this example, the chosen trigger was the Distance Trigger, with a value of five units. As such, it is only needed to enter the *Trigger Type* menu and press the Distance Trigger option to allow the specification of the desired distance value, as illustrated in Figure 35. This results in the activation of audio associated to the selected prefabs to trigger whenever the prefabs are in a radius less or equal to the chosen value.

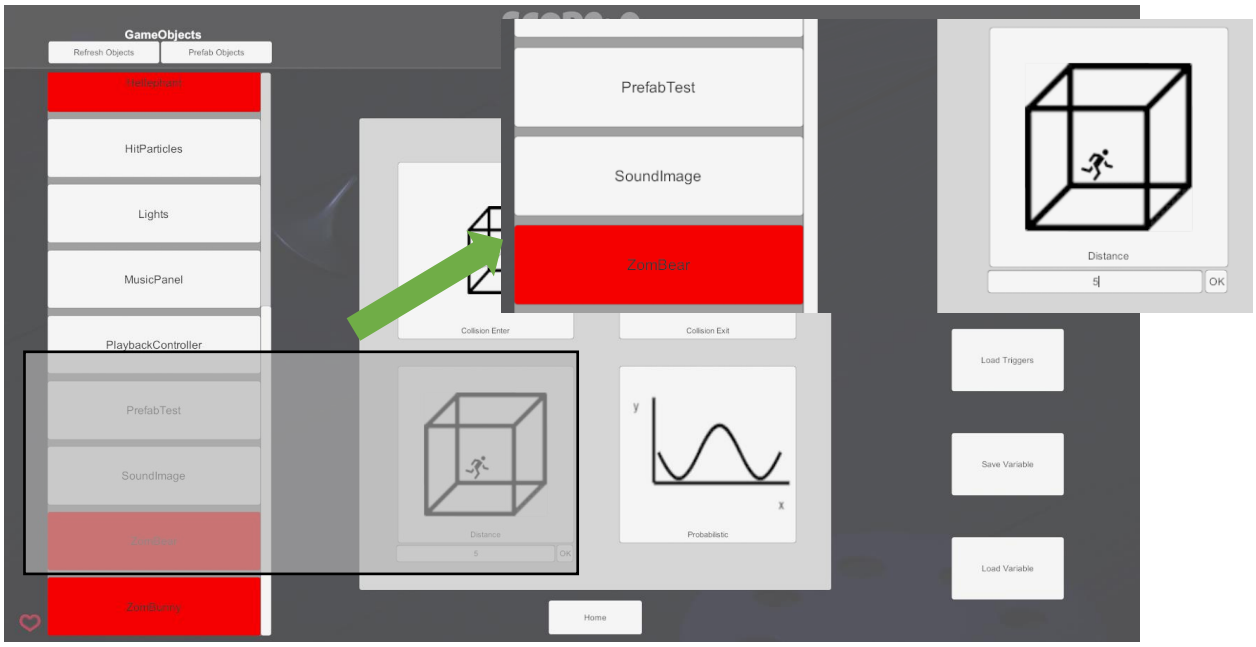

Figure 35 - Distance Trigger selection.

After the selection of the trigger and the distance, the user will be returned to the main Sonification menu, with the *Area Settings* option available. This menu will enable the user to choose the type of collider to add to the selected game objects. For this example, an appropriate collider was a Sphere Collider, with a radius of one unit. To do so, the user just needs to press the Sphere Collider button and fill the input field with a value of one. Figure 36 illustrates the menu before applying the chosen parameters.

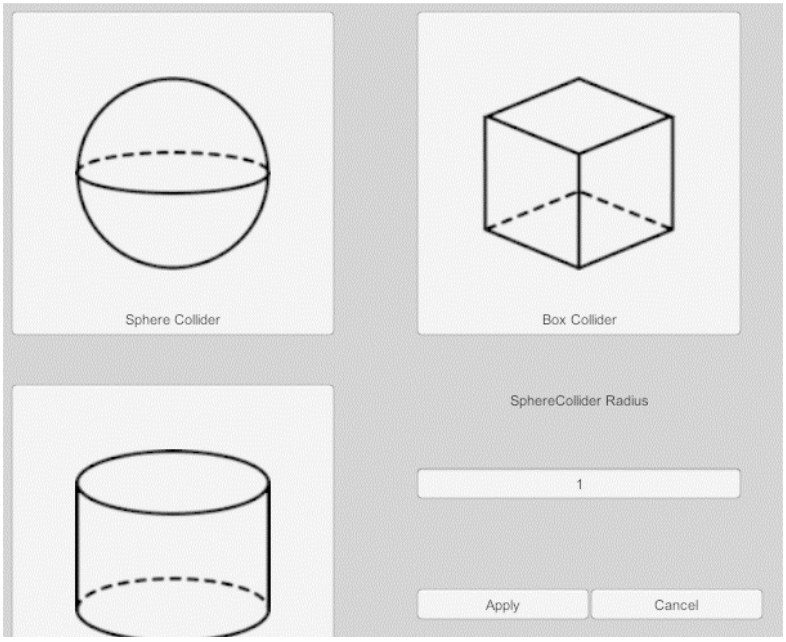

Figure 36 - Area Settings Menu.

Should the “Apply” button be pressed, the user will once again be returned to the main Sonification menu, with the *Spatialization Settings* and *Sound Options* button enabled. In the *Spatialization Settings* menu, the user should check the option of the 3D sound, as well as settings the appropriate values for the Doppler Effect, Spatialization Factor, Spread Angle, number of radius and type of Attenuation. Regarding the Spatialization Factor, it should be set to one, to make it a full 3D sound. For the Spread Angle, the value should be 360 to make it hearable from every direction. During testing the number of radius was kept at a value of one, meaning the sound would originate from the middle point of the game object and decay, according to the chosen decay type, until the specified radius in the *Area Settings* menu.

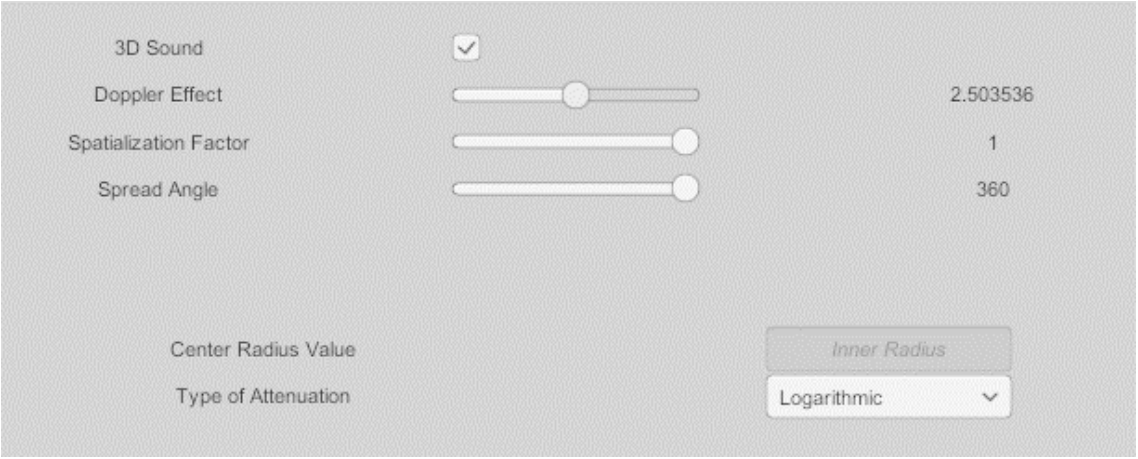

Figure 37 - Spatialization Settings setup.

Finally, the user must choose the audio to attribute to the selected game objects. To do so, the user should enter the *Sound Options* menu, where the audio-related options are presented. The footsteps should be audible whenever the enemies are in range, meaning the Loop options should be enable. Since the number of radius was set to one, only the perimeter audio option is available. Should the user press this button, the Circumplex Model will appear and display the available audio clips. The audio clip used for testing was under the Default Sound, named “Walking”, with the value for Valence and Arousal of 4.83 and 4.97, respectively. This audio clip is part of the IADS-2 dataset and has therefore already established values for both referenced dimensions. The dimensions of Valence and Arousal are not relevant for this demonstration and as such can be ignored. Regarding the type of music, since the 3D Sound option was enabled and only one radius was selected, Single Music is the only option available. Figure 38 shows the *Sound Options* menu configuration and Figure 39 shows the location of the used sound in the Circumplex Model.

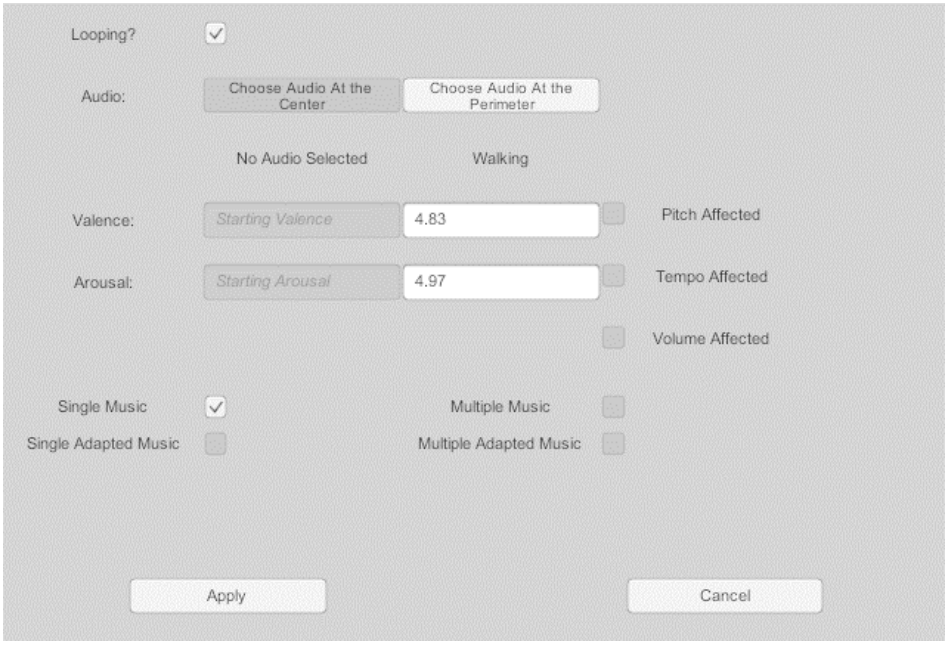

Figure 38 - Sound Options configuration.

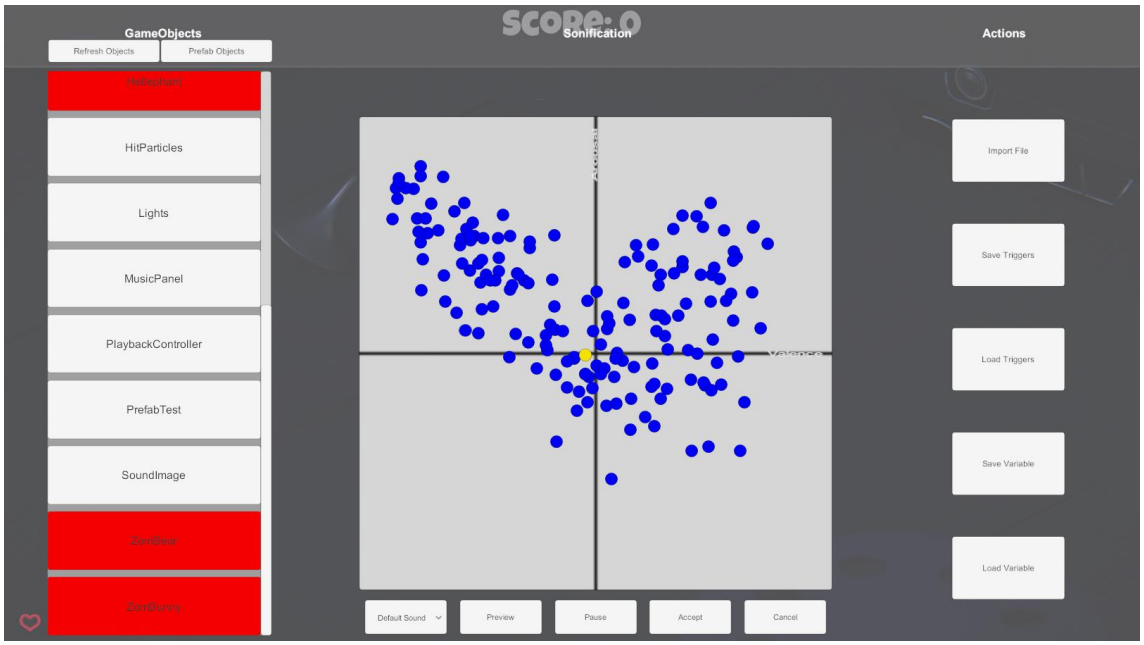

Figure 39 - Location of used sound in the Circumplex Model

Once all the mentioned configuration is completed, the user must only press the Apply button on the main Sonification menu to apply the setup (Figure 40).

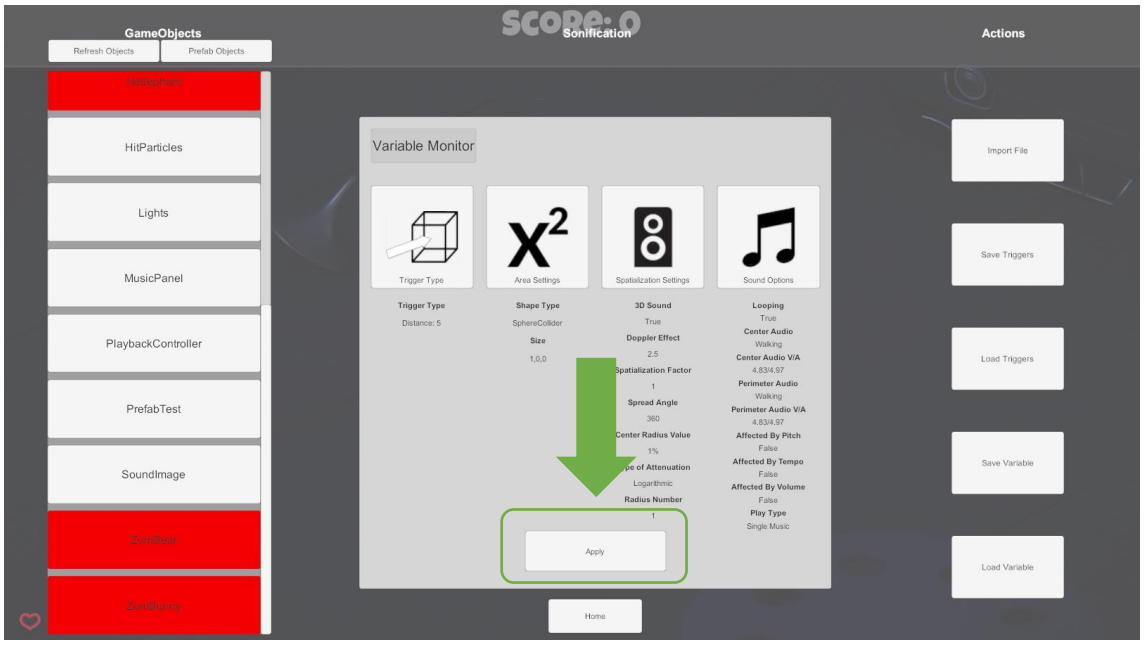

Figure 40 - Application of settings.

Having accomplished the first goal, the second goal was defined as adding background audio to be influenced in the Arousal dimension according to the number of enemies in a certain range of the player. To do so, the user must select the "Player" object from the list of game objects in the scene and press the "Variable Monitor" button. Once done, the user will be faced with the Variable Monitor menu, as in Figure 30. There the user should set the desired audio and the Valence and Arousal values in case a custom value is needed.

Figure 41 - Valence and Arousal adjustments on the Variable Monitor menu.

Afterwards, the user should choose the component which contains the target variable that will control the background music. In this case, a function was created on the component "PlayerMovement" that returns how many enemies are in the range of the player. The resulting variable that contains this information is called "EnemiesInRange". This setup can be seen in Figure 42, containing the chosen component, the variable and how the variable will affect the background music.

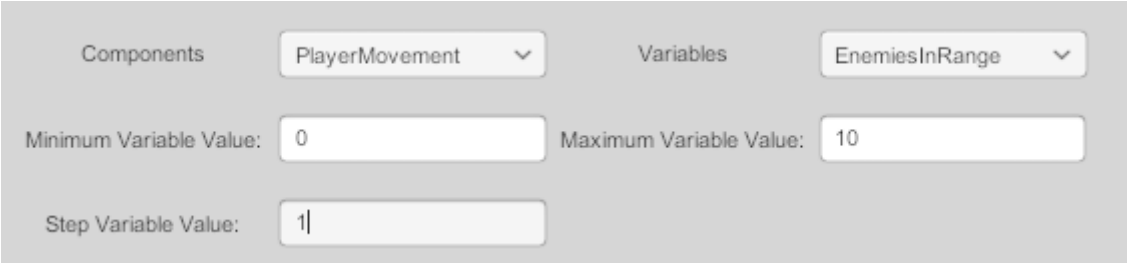

Figure 42 - Selection of Component and Variable on the Variable Monitor menu.

As can be seen in Figure 42, the chosen values for minimum and maximum variable value are zero and ten. This means that if the variable “EnemiesInRange” has a value that is below zero, no changes in the background music will be made. The same will happen if the monitored variable exceeds the value ten. Step Variable Value has a value of one, meaning that each time one enemy disappears or appears in range of the player, a change will be applied to the background music. Once this configuration is complete, the user will need to specify how the changes in the variable affect the audio. Figure 43 shows the enabling of changes in the audio in terms of Volume, Tempo and Pitch, targeting the Arousal dimension.

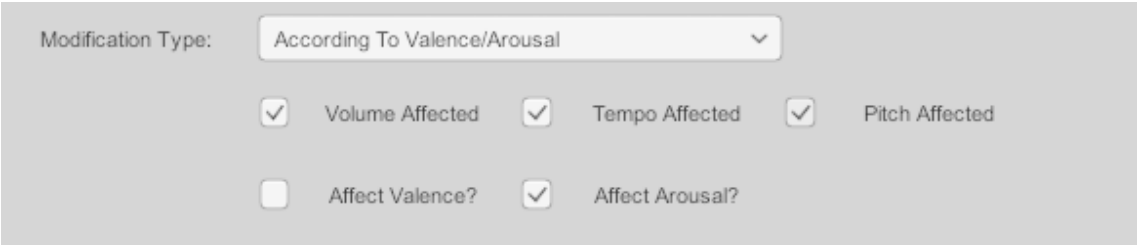

Figure 43 - Selection of Feature Changes on a selected Dimension.

Regarding the third objective, adding a warning sound whenever the player collides with an object, the steps are the same described for the first objective with the difference being the Trigger Type. To fulfil this objective, the user would need to choose the type Collision Enter, as it triggers when an object encounters the selected one (Player).

Arcade – Tetrimo

Tetrimo is a Tetris clone, a game where the user must line different geometrical blocks that are falling and complete lines to earn points. Each completed line yields points to the player and functions as the deciding factor for level advancement. For each level progressed, the speed of the falling blocks is increased, increasing the challenge. This game was developed by

DraphonyGames and the full project can be found on GitHub. A screenshot of this game is available in Figure 44.

Goal: The first objective defined for this game was to use the current level as a variable to control audio changes targeting the Valence dimension. The second objective was to use the block falling speed as a variable to control Arousal in the audio.

Due to the types of variables that contained this information, simple changes in the code were required. Values for Score, Level, Falling Speed and Completed Lines were already available and these were updated whenever a block reached its final position, resulting in the addition of four lines of code to the existing game logic. The created class, under the name *Holder*, was attached to an existing game object called “HUD: Gamefield”.

Figure 44 - Tetrimo game screenshot.

To reach the defined objectives, the process is the same as the described for the *Variable Monitor* on the previous Nightmare game. To do so, the user must press the F1 key to bring up the main Sonification menu and get the objects on the scene. From the resulting list, the object that contains this information is the mentioned “HUD: Gamefield”.

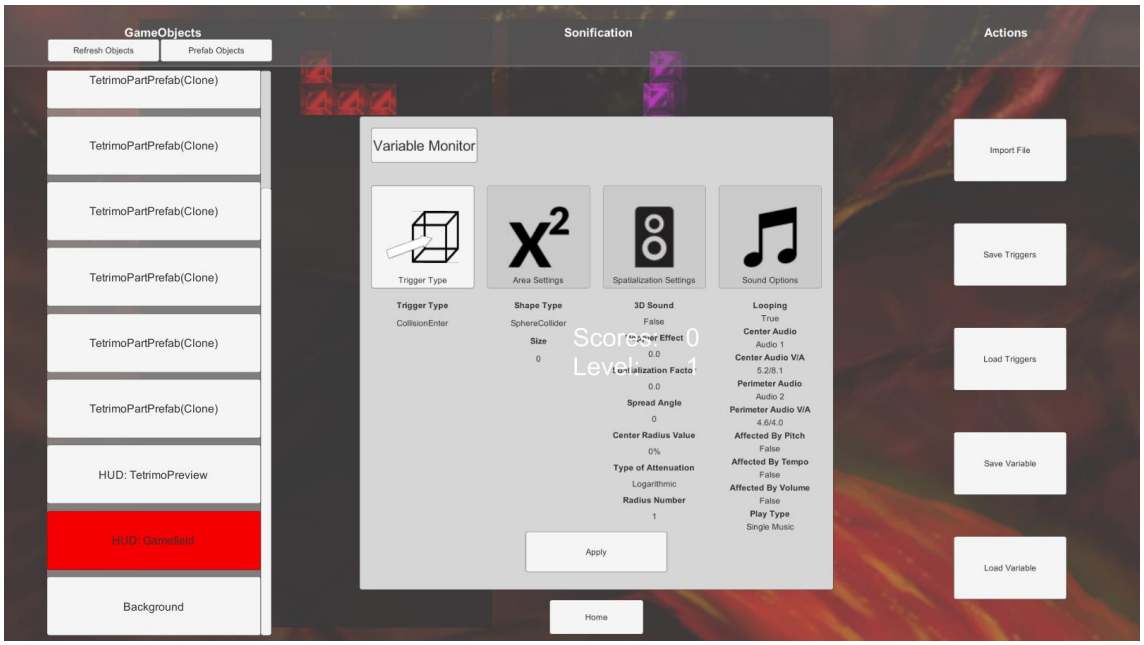

Figure 45 - Selection of object that contains the target variables

Once the object has been selected, the user should enter the *Variable Monitor* menu and select the audio clip to play, Valence and Arousal ranges, the Variable to monitor and its range and step, as well as the types of audio modifications to occur. Figures 41 through 43 show these steps. Figure 46 illustrates the component that holds the variables relevant to this game's sonification.

Figure 46 - Component that holds the relevant variables.

To use the falling speed of the blocked, the only alteration required is the selection of the variable from the Variable Dropdown, represented in Figure 46.

In these examples, the tool showed its capabilities and applications on different real and practical scenarios and contexts by making use of both Object Sonification and Variable Sonification. This chapter also presented the install process required by the developer for the integration of the developed *Unity3D* tool.

Chapter 6: Conclusions

This work had as its expected result the development of a tool that allows game designers a quick and easy way to perform sonification in games, to choose the music or sound more adequate to a situation in the game. The developed work implements the research objectives, resulting in a tool that empowers the end users to dynamically perform sonification on their games. This sonification includes (1) the analysis and categorization of the emotional content of music, (2) a graphical interface for the sonification of games in Unity, (3) stochastic sonification, through game events or game variables, and (4) the real-time manipulation of music following either a model-based adaptation of its Valence and Arousal or in a free-form. Since the tool retains its editing functionalities after a build has been compiled, users can also audio-design the levels of a game without programming requirements. The tool developed in *Unity3D* benefits from being easy to import, only requiring the import of the respective package and importing the *FMOD* plug-in. Similarly, it is also easy to import on existing projects by simply adding the two provided prefabs.

Literature points that while an agreement by the scientific community has not been achieved on what are emotions, it is still possible to classify them. Such classification is employed through this work, making use of the Circumplex Model and its Valence and Arousal dimensions. The Self-Assessment Manikin and some methods using measurements such as Heart Rate to infer Valence and Arousal from users have been explored in previous studies, with the Self-Assessment Manikin being widely accepted as a measurement system. Research also suggests that music can indeed provoke changes on the listener, with some musical features having a greater impact than others. Models that attempt to relate feature values to Valence and Arousal dimensions were explored and adapted for use, in a way that is transparent to the player. Further studies are needed to assert the impact of changes in musical features and what these induce on the listener. Having the possibility to modify an audio clip to express different values of Valence and Arousal can also be valuable, as it lessens the time and cost to produce new music in games.

The most important contribution of this work lies on the trained models for the prediction of Valence and Arousal of audio clips for the duration of this project. Several techniques were used, namely Linear Regressions, Support Vector Machines, Gaussian Processes and Artificial Neural Networks. The contribution is especially noted in the Valence dimension where previous research found a R^2 value of 0.281 in [27], against a value of 0.48 obtained during this work, representing an improvement of 71%. In the Arousal dimension previous results differ from the obtained ones, varying only 0.06 in R^2 and representing a decrease of 15%. Although it remains a great challenge, the existence of new paradigms in the Machine Learning field might in time provide a better solution than the one proposed here. For the time being, the findings of this project can pose an advance in emotion extraction from music, providing ways of better classifying

and perhaps understanding the underlying mechanisms of what is affects an emotion, how it is perceived and how it can be induced.

The developed tool for *Unity3D* takes the player position inside the game, information of objects located in the scene and classified audio clips in terms of Valence and Arousal to give game designers and players the choice of the audio intended, decided upon an emotional target. It is also possible to save the defined sonification and share it with other users by sharing the resulting JSONs. Users are also provided with default content to start, having the folders and always accessible. The alteration of audio features based on in-game variables expands upon this concept and presents the user with even more design choices for how to build custom music for their games, considering Valence and Arousal. By giving more power to target end users to decide what they want to hear and how they want to hear it, audio game design might be thought of differently.

Limitations

Due to limitations imposed by *Unity3D* and *FMOD* regarding the 3D sound engines, this work was not able to make use of the best of both worlds simultaneously. Another limitation consists on the restrictions on the file format, accepting only WAV files. This impacts the applications of the work developed, although it might be possible to overcome this issue in future versions of the tools used.

The work developed could not evaluated by any external game designer or player due to time limitations, only making use of three environments as *sandbox* for the testing while developing. The very controlled setting of development could result in limitation in terms of the expected functionalities of the tool, from a user's point-of-view.

Future Work

This work can be further improved by providing end-users a place to exchange sonification profiles, where users could upload the saved JSON files for a certain game. Should the specified *Unity3D* or *FMOD* limitations be removed, this would allow the implementation of the 3D sound affected by parameter change, resulting in an even greater level of customization for the user. The sonification performed can be based on the distance of the player relative to other objects, a concept that can be expanded to other applications such as the relative speed or weight and volume by using in-game variables to describe such dimensions. Documenting the used custom audio and the attributed Valence and Arousal could also prove to be beneficial. The latter would allow the making of a dataset of emotional audio and greatly help the train of new models for more accurate predictions. Studies with users will be required to determine the usability and functionality of the developed tool, most likely requiring *GUI* and functionality

changes. Finally, removing the need of a standalone MATLAB application would greatly improve the flow of the process, ideally performing all the required steps within the game, on the main menu presented.

The resulting *Unity3D* tool is to be used in a study with Alzheimer patients involving musical distortion, both as an audio stimulus and as a way of checking whether patients can distinguish altered audio parameters based on their memory of a music.

(Left in blank intentionally)

Bibliography

[1] G. Dubus and R. Bresin, “A systematic review of mapping strategies for the sonification of physical quantities,” *PLoS One*, vol. 8, no. 12, 2013.

[2] T. Hermann, A. Hunt, and J. G. Neuhoff, *The Sonification Handbook*, 1st ed. Berlin: Logos Publishing House, 2011.

[3] I. Ekman, “Psychologically Motivated Techniques for Emotional Sound in Computer Games,” in *Proc. AudioMostly 2008*, 2008, pp. 20–26.

[4] K. Collins, *Game Sound: An Introduction to the History, Theory, and Practice of Video Game Music and Sound Design*, 1st ed. Cambridge, MA: MIT Press, 2008.

[5] “Extra Credits: Video Game Music - YouTube.” [Online]. Available: https://www.youtube.com/watch?v=CKgHrz_Wv6o. [Accessed: 02-Mar-2017].

[6] P. Lopes, A. Liapis, and G. N. Yannakakis, “Sonancia: Sonification of Procedurally Generated Game Levels,” in *Proceedings of the 1st Computational Creativity and Games Workshop.*, 2015.

[7] J. A. Russell, “A circumplex model of affect,” *J. Pers. Soc. Psychol.*, vol. 39, no. 6, pp. 1161–1178, 1980.

[8] “Red Dead Redemption - Making of Music ViDoc | HD - YouTube.” [Online]. Available: https://www.youtube.com/watch?v=IJAIFmOdB00. [Accessed: 01-Mar-2017].

[9] T. Hermann, “Taxonomy and Definitions for Sonification and Auditory Display,” in *Proceedings of the 14th International Conference on Auditory Display*, 2008, pp. 1–8.

[10] “What Is Pitch in Music? - Definition & Concept - Video & Lesson Transcript Study.com.” [Online]. Available: http://study.com/academy/lesson/what-is-pitch-in-music-definition-lesson-quiz.html. [Accessed: 02-Mar-2017].

[11] “Timbre and envelope: From Physclips.” [Online]. Available: http://www.animations.physics.unsw.edu.au/jw/timbre-envelope.htm. [Accessed: 02-Mar-2017].

[12] “Spectrum, harmonics and timbre: From Physclips.” [Online]. Available: http://www.animations.physics.unsw.edu.au/jw/timbre-spectrum.htm. [Accessed: 02-Mar-2017].

[13] J. Williams, “What is an Emotion?,” *Mind*, vol. 9, no. 34, pp. 188–205, 1984.

[14] L. Smith-Lovin, M. Lewis, and J. M. Haviland, “Review: Handbook of Emotions,” *Contemp. Sociol.*, vol. 24, no. 3, p. 298, May 1995.

[15] K. R. Scherer, “What are emotions? And how can they be measured?,” *Soc. Sci. Inf.*, vol. 44, no. 4, pp. 695–729, 2005.

[16] J. K. Vuoskoski and T. Eerola, “Emotions Represented and Induced by

Music - The Role of Individual Differences," University of Jyväskylä, 2012.

[17] B. Knutson and S. M. Greer, "Anticipatory affect: neural correlates and consequences for choice," *Philos. Trans. R. Soc. B Biol. Sci.*, vol. 363, no. 1511, pp. 3771–3786, Dec. 2008.

[18] T. Eerola, O. Lartillot, and P. Toiviainen, "Prediction of multidimensional emotional ratings in music from audio using multivariate regression models," in *ISMIR*, 2009, pp. 621–626.

[19] M. Bradley and P. J. Lang, "Measuring Emotion: The Self-Assessment Manikin and the Semantic Differential," *J. Behav. Ther. Exp. Psychiatry*, vol. 25, no. I, pp. 49–59, 1994.

[20] D. Griffiths, S. Cunningham, and J. Weinel, "Automatic Music Playlist Generation Using Affective Computing Technologies," in *Fifth Int. Conf. on Internet Tech. and Applications (ITA13)*, 2013, pp. 177–183.

[21] A. Drachen, L. E. Nacke, G. Yannakakis, and A. L. Pedersen, "Correlation between heart rate, electrodermal activity and player experience in first-person shooter games," in *Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games -* , 2010, pp. 49–54.

[22] P. N. Juslin and P. Laukka, "Expression, Perception, and Induction of Musical Emotions: A Review and a Questionnaire Study of Everyday Listening," *J. New Music Res.*, vol. 33, no. 3, pp. 217–238, 2004.

[23] P. N. JUSLIN and E. LINDSTRÖM, "Musical Expression of Emotions : Modelling Listeners' Judgements of Composed and Performed Features," *Music Anal.*, vol. 29, no. 1–3, pp. 334–364, Mar. 2010.

[24] P. Gomez and B. Danuser, "Relationships between musical structure and psychophysiological measures of emotion," *Emotion*, vol. 7, no. 2, pp. 377–387, 2007.

[25] B. Schuller, S. Hantke, F. Weninger, W. Han, Z. Zhang, and S. Narayanan, "Automatic recognition of emotion evoked by general sound events," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 341–344.

[26] Y. E. Kim *et al.*, "Music Emotion Recognition : a State of the Art Review," in *Proc. ISMIR*, 2010, pp. 255–266.

[27] Y.-H. Yang, Y.-C. Lin, Y.-F. Su, and H. H. Chen, "A Regression Approach to Music Emotion Recognition," *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 16, no. 2, pp. 448–457, Feb. 2008.

[28] Y.-H. Chin, J.-C. WANG, J.-C. Wang, and Y.-H. Yang, "Predicting the Probability Density Function of Music Emotion using Emotion Space Mapping," *IEEE Trans. Affect. Comput.*, pp. 1–1, 2017.

[29] A. Aljanaki, Y. H. Yang, and M. Soleymani, "Developing a benchmark for emotional analysis of music," *PLoS One*, vol. 12, no. 3, 2017.

[30] M. Soleymani, A. Aljanaki, and Y.-H. Yang, "DEAM: MediaEval Database for Emotional Analysis in Music," pp. 3–5, 2016.

[31] J. A. Nelder and R. W. M. Wedderburn, “Generalized Linear Models,” *J. R. Stat. Soc. A.*, vol. 135, no. 3, pp. 370–384, 1972.

[32] C. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2006.

[33] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning.*, vol. 14, no. 2. Cambridge, MA: MIT Press, 2005.

[34] “Feedforward neural network - MATLAB feedforwardnet.” [Online]. Available: https://www.mathworks.com/help/nnet/ref/feedforwardnet.html. [Accessed: 20-Jan-2018].

[35] “Artificial Intelligence Neural Networks.” [Online]. Available: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm. [Accessed: 20-Jan-2018].

[36] “FMOD.” [Online]. Available: https://www.fmod.com/. [Accessed: 12-Feb-2018].

[37] D. Kogan, “Game Audio Table - Audio Middleware Implementation Table.” [Online]. Available: http://danikog.github.io/GameAudioTable/. [Accessed: 01-Mar-2017].

[38] A. P. Soares, A. P. Pinheiro, A. Costa, C. S. Frade, M. Comesaña, and R. Pureza, “Affective auditory stimuli: Adaptation of the International Affective Digitized Sounds (IADS-2) for European Portuguese,” *Behav. Res. Methods*, vol. 45, no. 4, pp. 1168–1181, 2013.