UtopicSense: a tool to support the use of synesthesia as an assistive resource in 2D games

Adriel Augusto Germano Silva
Departamento de Computação
Centro Federal de Educação Tecnológica de Minas Gerais
Belo Horizonte, Brazil
adrielags@gmail.com

Flávio Roberto dos Santos Coutinho
Departamento de Computação
Centro Federal de Educação Tecnológica de Minas Gerais
Belo Horizonte, Brazil
fegemo@cefetmg.br

***Abstract*—The concept of synesthesia in games comprises providing stimuli in more than one sense, such as displaying visual effects concurrently with sound effects on a screen. Algorithmic synesthesia is a concept that describes multimedia processes in which sounds, images, and in some cases, haptic sensations have as generators the same computational process or data source. Game developers can leverage such concept as an ancillary resource for people with hearing loss or deafness. Despite its advantages to the gaming experience, synesthesia is rarely contemplated in gaming design with the focus on accessibility. This situation usually happens because developers are unaware of its benefits or want to avoid additional costs and more implementation work on the project's schedule. This paper presents UtopicSense, a tool that automates implementing synesthesia in 2D games inside Unity. To create it, we identified positive and negative accessibility aspects perceived through the analysis of 2D games. We then developed the tool which allows users to configure aspects of the generated visual effects through a Unity custom editor panel. We evaluated the tool in terms of usability with potential users (Unity game developers). They provided feedback and evaluated the tool in terms of ease of use, flexibility, and usefulness. Participants expressed positive reactions towards UtopicSense and indicated that it fulfilled its goal of integrating algorithmic synesthesia into 2D games in an easier way.**

***Index Terms*—algorithmic synesthesia, game accessibility, hearing impairment, deafness, unity**

I. Introduction

Digital games are significant leisure means, and as such, they should be a viable option to everyone, regardless of physical or mental conditions [1]. For that to be possible, it is necessary to take accessibility into account. Accessibility in games is the possibility of having a good gaming experience even under functional limitations or disabilities [2]. A technique that can improve user accessibility and immersion associated with sensory stimuli is the use of synesthesia. This concept refers to a neural condition in which people perceive sensations of a stimulus through different senses (for example, a sound stimulus linked to a visual effect). In the case of digital games, simultaneously conveying information through multiple media is also understood as synesthesia. And besides the benefits to player immersion [3], in the case of players with hearing impairment, the use of synesthesia associating sounds with visual effects (and other media) is an alternative to making games more accessible and exciting [4].

Despite the benefits, implementing synesthesia requires time and is not always contemplated in projects' schedules [5] and, when it is, it is usually not included as an assistive resource. Part of the process of generating different stimuli from the same source (eg, create a visual effect based on an audio file) can be automated in a process called algorithmic synesthesia [4].

In this context, this paper proposes, UtopicSense, a tool to apply algorithmic synesthesia in 2D games. It targets game developers who seek to implement synesthesia (associating sound and visual stimuli) in their games to improve accessibility for the deaf and hard-of-hearing players, but also to build on the game's immersion [3], [5].

We implemented the tool within the Unity engine, which is a popular game development software [6], [7]. Utilizing Unity, people can create 2D and 3D games, and many developers used it to create very successful and popular games [8].

For the development of this work, we gathered requirements based on positive and negative aspects of accessibility we perceived through the analysis of some 2D games. We implemented a game prototype with basic synesthesia functionalities as a model for the final tool from the defined features list. We also conducted a preliminary evaluation with Unity game developers assessing the tool's quality of use.

As a result, we have a tool that allows developers to use Unity to associate sound with visual stimuli, flexible enough for different genres of 2D games. We developed the application to respect the game art direction freedom and subserve the synesthesia's implementation process in a more automated way. In a preliminary evaluation, from the tasks users had to execute, they completed 85.3% without mistake, 14.7% with errors, and there were no tasks that users could not complete.

We expect the tool to facilitate the creation of 2D games that use synesthesia. Therefore, it can positively impact the user experience, especially the players with hearing loss or deafness. Another contribution stemming from the tool is the possibility to evaluate the use of synesthesia as an assistive resource thoroughly.

II. Synesthesia

The word synesthesia has a Greek origin (*sin* + *aisthesis*), and it means the meeting of multiple sensations. It is a concept approached in several knowledge fields such as Biology and Arts [9].

The Neuroscience field considers that synesthesia is a neurological condition that causes the stimulus of one sense to cause reactions in another, creating a sensory mixture between sight, smell, hearing, taste, and touch. In Arts, synesthesia contemplates metaphors and comparisons to evoke multiple sensations across multiple works.

However, synesthesia exceeds those areas, reaching other spheres such as technology [10]. In this field, a concept introduced by Dean *et al.* [11] is algorithmic synesthesia. It describes multimedia processes in which sounds, images, and in some cases, haptic sensations have as generators the same computational process or data source. An example would be an algorithmic process that generates visual and sound information simultaneously as its output.

This concept brings synesthesia to the world of interaction and makes it a lively resource for interactive media such as games [12]. Many games implement this concept, especially as they have become well-established and viral immersive media [5]. One way to increase this immersion is through the implementation of synesthesia. Associating an event interaction with several sensations can engage different types of players [13]. There are highly successful commercial games like Journey [14] that use the synesthesia concept applied to their artistic vision (Fig. 1).

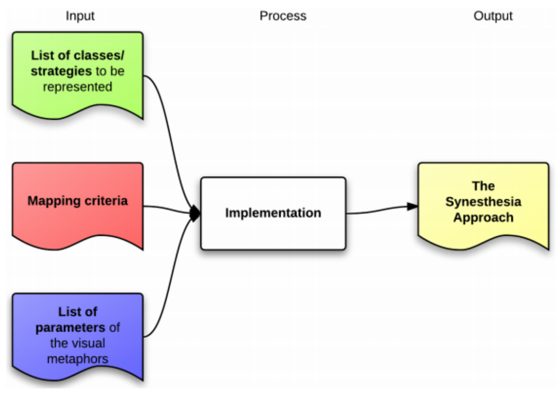
Fig. 1. Example of the use of synesthesia in the game Journey.

III. Related Works

Different works show that synesthesia can improve the player experience (a) by enabling higher levels of immersion and also (b) as an assistive tool. In the first research line (a), although the definition of immersion being a source of debate, Bastos *et al.* [3] identified seven characteristics in games that contribute to making the player feel like part of the experience, and one of them was audiovisual synchronicity. They created a game prototype with five of those features. After evaluation, they observed that the synesthesia resulting from the audio and video cohesion was the most immersive factor in the game.

Liang [5] presented different ways game designers can use synesthesia to create a more immersive atmosphere. The study advocates drawing inspiration from the findings from other fields (such as Arts, Neuroscience) regarding how synesthetes (people who have the neural condition of mixing senses) map properties from one stimulus into another (e.g., sound frequencies to colors).

In the second research line (b), Coutinho [4] introduced a theoretical framework that allows designers to reflect, propose and implement synesthesia in their games. It encompasses a process to define the synesthesia model (shown in Fig. 2) and another to specifically determine how to map characteristics from one medium into the other (Fig. 3).


Input
Process
Output
List of classes/ strategies to be represented
Mapping criteria
List of parameters of the visual metaphors
Implementation
The Synesthesia Approach
Fig. 2. Process of the synesthesia model implementation.

The overall process (Fig. 2) involves determining which elements from a source medium (e.g., auditory) convey information that should also be transmitted through a target media (e.g., visual). Designers also define which characteristics of the target media can be used to convey information (e.g., visual effects with different colors, shapes, sizes). Finally, it requires consistent mapping criteria for features in the source to the target medium such that in an audio to visual approach, the visual metaphors used to represent sounds can somehow visually depict the information conveyed through sounds.

In turn, the process to determine the mapping criteria (Fig. 3), besides generating a consistent association, should be harmonic with the game art style. The implementation could be manually performed by game designers or be partly or entirely automated (algorithmic synesthesia).

In a similar line of research, Silva, Callado, and Jucá [15] investigated using synesthesia from the audio to visual medium with a manual implementation. They used particle systems with different emission shapes, movements, and colors and tried to map different configurations of such parameters to feelings (e.g., anger, love, fear). After an experiment with 40 users, the study found mappings with some consistency to six emotions and made a Unity plugin with pre-configured visual effects available.

Our work builds on the concept of algorithmic synesthesia, also on the audio to visual mediums, by providing a Unity tool that automates the procedural generation of visual effects from audio files, specifically for 2D games. In addition, such

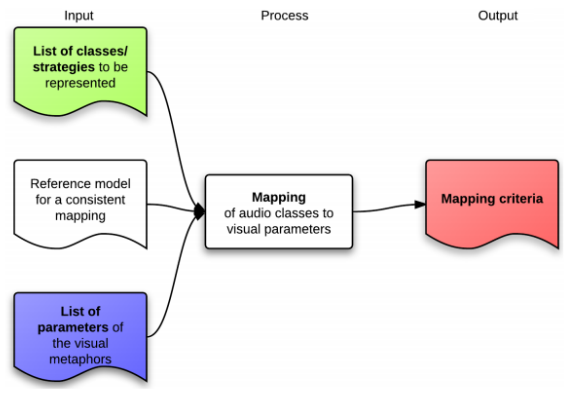

Input
Process
Output
List of classes/ strategies to be represented
Reference model for a consistent mapping
List of parameters of the visual metaphors
Mapping of audio classes to visual parameters
Mapping criteria

Fig. 3. Process of defining mapping criteria between media.

effects can be configured, allowing designers to keep control over the art style of the visual metaphors.

IV. Methodology

This work followed a cascade software development process split into 4 phases. The first one (1) consisted of defining requirements for the UtopicSense tool. We surveyed the literature on synesthesia and algorithmic synesthesia. Then we systematically analyzed some 2D games to raise potential problems and positive aspects related to accessibility for deaf and hard-of-hearing players. The selected games comprised different genres with action-based 2D mechanics. We analyzed this genre because of its popularity and the relevance of its audio in communicating with the player [16].

After that, in the second step (2), we built an algorithmic synesthesia prototype comprising a simple scenario with the essential functions for mapping sounds into visual effects. The prototype allowed us to focus first on the vital aspects for implementing synesthesia in games, as defined in [4].

In this paper, we create visual representations for sound stimuli, as if the player could see the game sounds. Hence, we selected audio characteristics (e.g., frequency, volume) to use as parameters to map into visual effects properties (e.g., colors, opacity, speed). After mapping those characteristics, we implemented the synesthesia approach to produce the visual effects based on the input sounds.

Afterward, in phase (3) we implemented a user interface for developers to configure the synesthesia model in their projects according to their game design concept.

In the last step (4), we assessed the tool regarding (a) the flexibility to apply its functions into different 2D games genres and also (b) to gather some preliminary feedback about the tool's user interface from Unity game developers.

V. Utopic Sense

This section presents the process of creating UtopicSense, detailing the steps described in the methodology and some technical decisions.

A. Requirements Gathering

As an initial step, we needed to define the requirements for a tool that supported the implementation of synesthesia. Then, we analyzed how sounds have been used in 2D games to convey information [16]. In doing so, we identified both good and bad communication strategies through audio and then proposed a set of features the tool should have. The selected games were:

- The Legend of Zelda: Link to the past (1991) [17]
- Limbo (2010) [18]
- Mark of the Ninja (2012) [19]
- Ori and the Blind Forest (2015) [20]
- Rayman Origins (2011) [21]
- Rayman Legends (2013) [22]
- Shadow Dancer (1989) [23]
- Sonic Mania (2017) [24]
- Sonic the Hedgehog (1991) [25]
- Sonic the Hedgehog 2 (1992) [26]
- Streets of Rage (1991) [27]

We analyzed the games by experiencing them with and without audio to compare positive and negative aspects to the player experience in both situations. For this evaluation, we differentiated primary (must be perceivable by the player to play the game) and secondary stimuli (being able to play a game does not depend upon being able to perceive these stimuli) proposed by Yuan, Folmer, and Harris [28]. We applied this concept together with the differentiation of diegetic sounds (sounds that communicate something that belongs to the game world) and non-diegetic sounds (sounds that communicate something through the interface but do not belong to the game world) to make the complete analysis.

In 2D platform games and others with a side-scrolling camera (e.g., Sonic the Hedgehog [25], [26], Streets of Rage [27]), there are several situations in which the game uses audio to communicate that a relevant event will occur in a few seconds or is already happening. Often, this kind of sound represents events in the game that will require some quick reaction from the player.

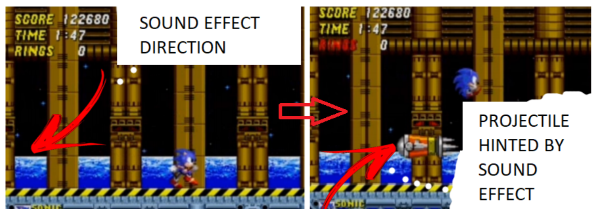


Fig. 4. Reacting to a sound effect emitted from outside the view range.

A problematic situation arises when game objects located outside the field of view emit primary auditory stimuli. In this case, important events may happen unnoticed or only get noticed after frustrating situations for those who do not receive this valuable information. Fig. 4 exemplifies a problem where the player may be frustrated by the fact that the game communicates the imminence of an out of sight event exclusively through a sound effect.

In other games, such as The Legend of Zelda: Link to the past [17] (illustrated by Fig. 5), there are also restrictions on what the player sees due to a simulation of darkness, fog, etc. In this situation, there are also accessibility problems when emitting sounds from regions outside the visual field. With the proper receipt of the sound information, the player has a higher propensity to have a better experience and get more resources (valuable game items for the character) hidden in dark or foggy places within the game. Thus, the more resources the character has throughout the game often entail better difficulty balance in gameplay flow and consequently better player experience.

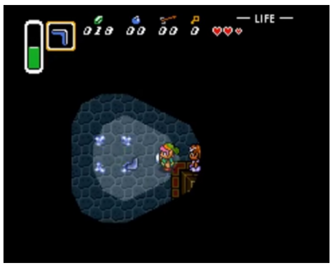


Fig. 5. Sounds sometimes come from dark areas in games.

We identified similar situations were frequent in older games. On the other hand, in more recent ones we could notice considerable improvements regarding these aspects. Despite this, there are still games that contain problems in their accessibility for the deaf and hard-of-hearing players.

Some games still exclusively use sound effects to provide clues of certain events, which are indispensable for progressing in the game (unlocking new levels and other bonuses). An example of this occurs in Rayman Origins [21]. This game rewards the player for locating and solving tasks in hidden areas. Finding such regions is facilitated by the fact that a sound signal guides their location. Not receiving the audio clue might lead the player to ignore these locations. Fig. 6 illustrates this event.

The games that do not have this type of problem use more than one stimulus to communicate an event or call attention to a particular location. In these cases, the valuable sound information often comes together with simultaneous haptic or visual effects.

Limbo [18], for example, uses the vibration provided in the player's control as an alternative to raising awareness of events in the game. A game that employs the use of visual effects linked to sounds is Mark of the Ninja [19], a 2D stealth game illustrated in Fig. 7. It associates the footsteps sounds of enemies (which are not always visible) and sounds of events that occur off the screen with visual effects that players perceive within the field of view in the game interface. The game communicates events through stimuli in more than one sense. Although creators may not have designed this mechanic with a focus on accessibility, it improves the game experience for a larger audience.

Table I
REQUIREMENTS FOR THE TOOL

	Requirements
R1	Generate 2D visual effects from sounds
R2	Support different game genres
R3	Allow developers to relate sounds of their choice to visual effects
R4	Support appliance of visual effects from different artistic visions
R5	Provide visual information for elements out of the game range view
R6	Provide the option to activate or deactivate visual effects in-game

From the analysis of 2D games from different eras, we verified that diegetic sound effects played an important role in most positive and negative communication strategies through audio. After this survey, we raised a list of requirements for the synesthesia tool, which we present in Table I.

In parallel with the requirements gathering, we also needed to define which sound characteristics we would map to visual effects properties. Based on the classes of auditory signs proposed by Nogueira *et al.* [29], we selected the following sound features:

- **Sound volume** maps to the opacity of the visual effect, that is, the lower the volume, the dimmer the effect, making it more subtle.
- The **frequency** of the sound maps to the visual effect color. Users can relate colors and particle effects to pre-established frequency ranges according to their artistic view.
- The **range inside which the player receives a sound** maps to the area of the visual effect, with sounds which can be heard from very far having extensive associated visual effects.

After defining this sound-to-image mapping model and the gathered requirements, we implemented the UtopicSense prototype, which is detailed next.

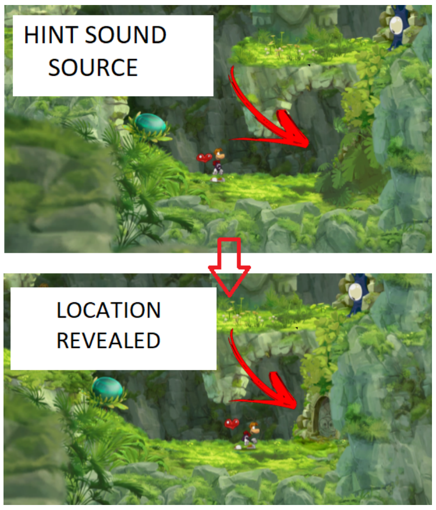


Fig. 6. Hidden area with sound hint in Rayman Origins.

B. UtopicSense Prototype

The UtopicSense tool prototype consisted of creating a 2D platform game scenario in Unity with the implementation of the algorithmic synesthesia mapping to produce the visual stimuli from the auditory. In this phase, we wanted to create a working scene with the effects implemented but leaving for a later stage the creation of an interface that allowed the configuration of the synesthesia. In the created game location, there were various objects emitting sounds in different situations.

We used a particle system as a visual representation of sound effects in the generated scenario. It was built based on the appearance of the physical ripple effect (cascade effect) phenomenon in water (the impact of mechanical waves gradually expanding through the water when it is disturbed). We used a texture with a circular form as the particle material so that the visual aspect resembled the desired design. Fig. 8 shows three stages of the lifetime of a particle.

The particle effect creates particles from the exact position of the sound source, and they grow and lose opacity over some time. The scene developed for the prototype had a controllable character, a jukebox object, and a cannon shooting projectiles. This scene (Fig. 9) contains 3 sound sources: the first associated with the cannon object (which has a sound emitted when it shoots projectiles), another for the jukebox object which plays music, and the controllable character (which emits a sound when jumping and colliding with the floor).

Using Unity's scripting system, we created a function to emit particles when a sound plays. This script dynamically creates particle sources that will react according to the sound source. For this, the script detects the `AudioSource` component (which is a Unity component responsible for reproducing sounds). For the algorithm, it is possible to catch more than one `AudioSource` component in a single object (an object can emit more than one type of sound and therefore have more than one `AudioSource` component). The synesthesia function detects a sound playing and, under this condition, starts the particle emission, which is instantiated gradually until the sound transmission stops. An example of a visual effect created for the cannon is shown in Fig. 10.

In the visual effects implementation function, we also added the functionality of associating the range of the sound in Unity to the scope of the visual effect (maximum distance of the source from the game character from which it still hears sound effects). Thus, the player can only see the visual effect within the range on which the game character should listen to the audio. Fig. 11 illustrates this functionality.

The same function maps the sound volume into visual effect transparency, making lower volume sounds produce more subtle visual effects. The `AudioSource` volume has a range of 0 to 1, and the software multiplies this value to the opacity of the visual effect (Fig. 12).

Based on the need for each sound to convey a unique visual aspect to allow their differentiation (e.g., the effect for the cannon sound being different from the one from the jukebox), we mapped the dominant frequency bands of the audio in different colors of the visual effect. The challenge of this process is the fact that most sound effects are composed of a combination of frequencies [30]. To process the audio files and determine its dominant frequency, we used a Unity function called `GetSpectrumData`[1]. It transforms the sound wave from the temporal to the frequency domain and, thus, allows the sound analysis, receiving as a parameter a Fast Fourier Transform (FFT) window.

Unity provides different options for calculating the FFT window (used to improve the approximation when converting the audio signal from continuous to discrete), and we used Blackman-Harris [31]. Although this is not the most efficient FFT window, as it involves sophisticated calculations compared to other options, it has the best accuracy for calculations that require greater precision. Computations such as those for determining the dominant frequency and tone (e.g., low,

[1]GetSpectrumData Unity documentation available in: https://docs.unity3d.com/ScriptReference/AudioSource.GetSpectrumData

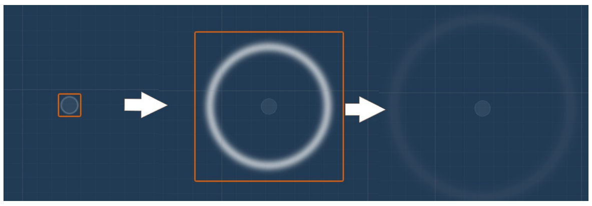


Fig. 8. Particle effect used to represent sounds visually in the prototype.

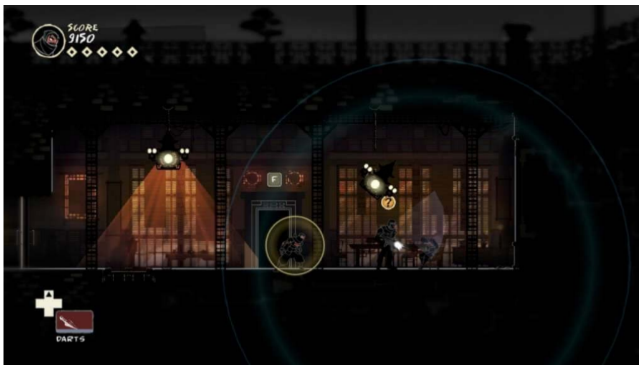


Fig. 7. Game Mark of the Ninja depicting sounds visually.

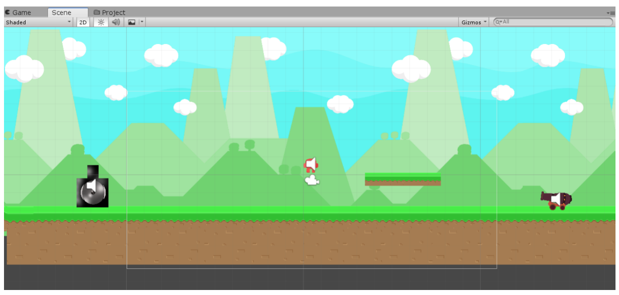


Fig. 9. Scene for implementing UtopicSense basic functionalities.

high, etc.) of the sound signal fall into such a situation [32], [33]. Precision is a required characteristic for UtopicSense, so the software can map the pitch values to the expected corresponding color. This FFT window also did not affect the 2D game's performance (including more complex projects using the tool's features).

Longer-lasting sounds, especially music, vary their dominant frequency over time, so it has to be detected continuously. Sampling is the transformation of a continuous signal into a discrete form in signal processing. We sampled the sound wave at regular intervals to be able to calculate the dominant frequency. The sound is discretized into 512 samples for frequency analysis. We chose the value through empirical tests, which consisted of the gradual application of powers of 2 in the sampling number 128 to 2048 (i.e., 128, 256, 512, 1024, and 2048). We applied the value of 512 samples as it had good results without negative performance impacts on the algorithm.

To map dominant frequencies in color, we split them into frequency bands, described by Table II. The bands include values between 20Hz and 20,000Hz as they are audible bands for the human ear [34]. Thus, the scopes with lower frequency values were arranged in smaller value intervals to enable more significant color variability in the particle system. These intervals were divided with non-uniform bands, as we observed a greater incidence in detecting lower dominant frequencies in sounds used in games. We performed an empirical analysis with 30 songs and sound effects to sustain this observation.

We associated each frequency range with a color pattern (color gradient), which updates colors as new dominant frequencies are recognized (i.e., a color change occurs as the prevalent frequency change occurs). The particle effect used in the project emits colors from a color gradient over time. That is, there is a two-color gradient related to a frequency band. As a more concrete example, for a sound effect in the frequency of 500Hz to 1000Hz, the particles appear red and disappear as orange.

We picked the standard colors sampling from the visible light spectrum. Therefore, lower frequency bands are associated with lower frequency colors in the light spectrum and higher color frequencies representing higher frequencies in that spectrum.

With the script to generate the visual effects implemented into the prototype, the next step was to incorporate these features into a graphical interface to allow developers to configure all of the settings. It contains the fundamental functions of the prototype specified in this section, with further adjustments to the features that make the application of effects more flexible to the artistic vision of game developers.

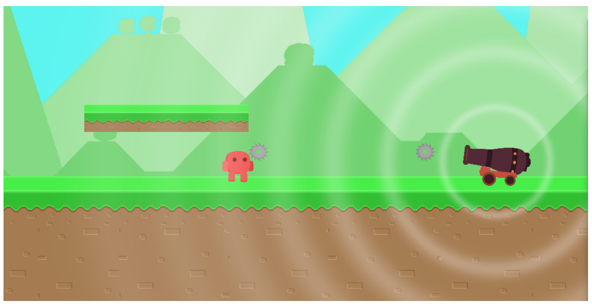


Fig. 10. Particle effects in the moment the cannon emits sound.

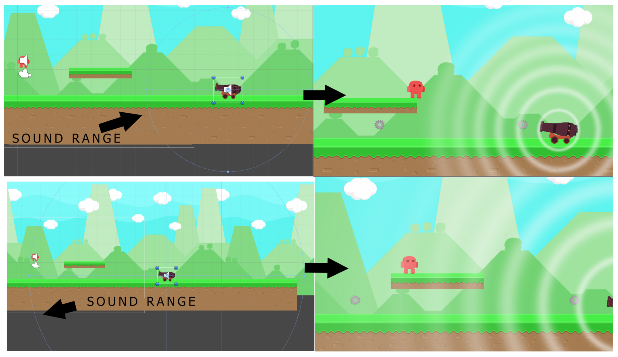


Fig. 11. Distance from the sound mapped in the scene.

C. *UtopicSense*

We implemented UtopicSense by extending the Unity editor and creating a custom `EditorWindow`[2]. With that, it was possible to assemble a graphical interface from pre-existing widgets and incorporate the window in the game creation environment (Fig. 13).

The tool detects configuration changes made by the user through its interface and automatically saves them in a file generated by UtopicSense within the project folder. It excludes the need to rely on a save button and centralizes all the

[2]Reference: https://docs.unity3d.com/ScriptReference/EditorWindow.html

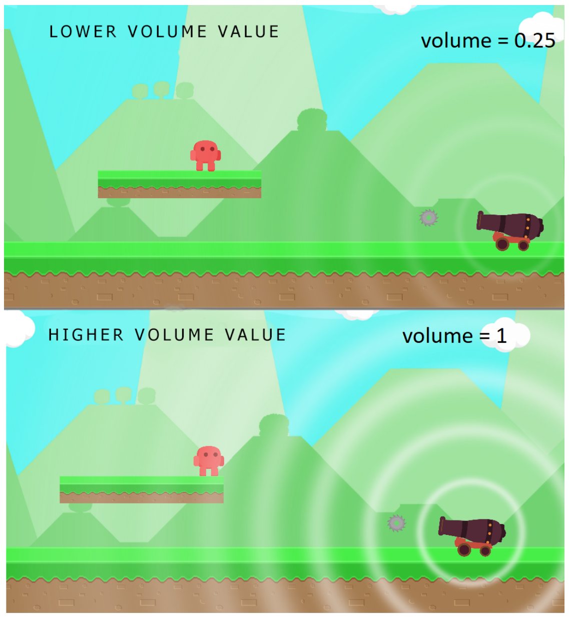


Fig. 12. Sound volume mapped to effect opacity.

Table II
UTOPICSENSE FREQUENCY RANGES AND DEFAULT COLORS MAPPED

Frequency Ranges	Gradient Colors	Hexadecimal	Color Swatch
20Hz to 100Hz	initial color	FC00FF	🟪
	final color	FF0094	🟥
100Hz to 500Hz	initial color	FF004C	🟥
	final color	FF1F00	🟥
500Hz to 1000Hz	initial color	FF5400	🟧
	final color	FF8C04	🟧
1000Hz to 2000Hz	initial color	FFE204	🟨
	final color	C7FF00	🟩
2.000Hz to 5.000Hz	initial color	1DFF00	🟩
	final color	00FF9B	🟩
5.000Hz to 9.000Hz	initial color	004EF8	🟦
	final color	0000FF	🟦
9.000Hz to 20.000Hz	initial color	4E00FF	🟪
	final color	8300FF	🟪

synesthesia effects settings in a single file. To provide user support, buttons with the "?" symbol in the interface open additional windows with more detailed explanations about the program's functionalities. We present each of the software's features in the following subsections.

1) Standard and simple modes: In the upper right side of the program interface, there is the mode selection, which corresponds to UtopicSense being in standard or simplified mode. The application uses the dominant sound frequency in the standard application mode to calculate the color mapping on the image. UtopicSense maps all sounds in a single color gradient in the simplified mode and does not consider the frequency in this process. Beneath this area selection, there is the `UtopicSenseActive` field, where the user can activate or deactivate the synesthesia effects in the project.

2) Select particle textures: this region provides means for the user to select and customize their effects according to their artistic vision. UtopicSense starts with five standard textures, but each one can be set to any texture within the project, as Fig. 14 illustrates.

3) Shader selection: the user can select different shaders to change the visual effect appearance and adapt the effect to different artistic concepts. Fig. 15 shows different shaders applied to the visual effect in the scene.

4) Color selection: The standard colors are defined and presented here as discussed in subsection IV-B. The user can modify the colors assigned to frequency bands according to need. After clicking on a color swatch, and RGB color selector, a window will open. Besides the color box, a dropper tool allows selecting any color on the computer screen. If the user wants to return to the default colors of the software, one can use the button "Reset default colors."

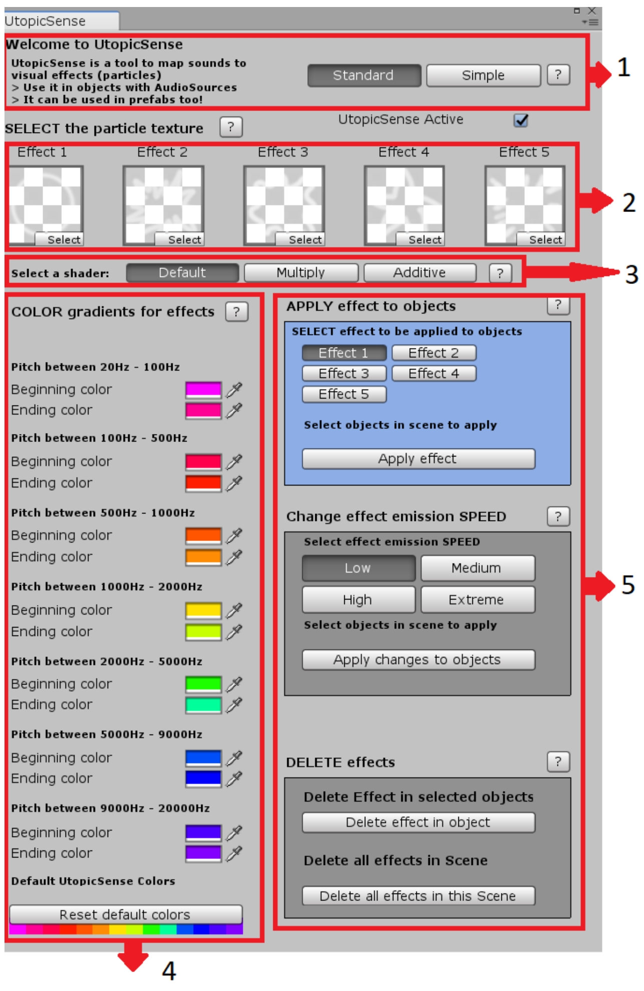
5) Apply or remove effect in selected objects: it is up to the game developer to choose and identify which scene objects will use the synesthesia effect. They can attach the particle system by selecting one or more game objects or prefabs[3] and then clicking "Apply effect". The user must select which effect to apply to those objects. Additionally, he can set the effect emission speed and change it in the selected items with attached effects (Fig. 16). In the same region of the interface, the user can also delete the effect in a single object or delete all effects in the scene.

After implementing the tool, we carried out some tests, which we describe in the next section.

D. Tool testing in 2D game projects

We tested the tool applying its functionalities to different 2D game projects of different genres. The initial test consisted of selecting five Unity 2D projects with different genres,

[3]*Prefabs* are objects saved in the project directory that serve as a model for replicating objects in other games scenes.


UtopicSense
Welcome to UtopicSense
UtopicSense is a tool to map sounds to visual effects (particles)
> Use it in objects with AudioSources
> It can be used in prefabs too!
Standard
Simple
?
1
SELECT the particle texture
UtopicSense Active
Effect 1
Effect 2
Effect 3
Effect 4
Effect 5
Select
2
Select a shader:
Default
Multiply
Additive
3
COLOR gradients for effects
Pitch between 20Hz - 100Hz
Beginning color
Ending color
Pitch between 100Hz - 500Hz
Pitch between 500Hz - 1000Hz
Pitch between 1000Hz - 2000Hz
Pitch between 2000Hz - 5000Hz
Pitch between 5000Hz - 9000Hz
Pitch between 9000Hz - 20000Hz
Default UtopicSense Colors
Reset default colors
4
APPLY effect to objects
SELECT effect to be applied to objects
Select objects in scene to apply
Apply effect
Change effect emission SPEED
Select effect emission SPEED
Low
Medium
High
Extreme
Apply changes to objects
DELETE effects
Delete Effect in selected objects
Delete effect in object
Delete all effects in Scene
Delete all effects in this Scene
5
Fig. 13. UtopicSense interface inside Unity.

artistic styles, and complexities to apply all UtopicSense functionalities. Among the projects, we obtained four from Unity Asset Store and a more complex project created by us.

In each project, we imported UtopicSense and its effects and functionalities applied to various objects and generators of pre-fabricated objects (prefabs). Then, we executed each game project and observed whether the effects were functional. We tested the following UtopicSense's features: (1) application of the standard and simple modes, (2) change of effect textures, (3) application of different shaders, (4) selection of new colors, (5) the application of many different effects in the same scene, (6) change in intensity in the visual effect, and the (7) deletion of effects on objects.

The tests enabled us to perceive the flexibility of applying the tool's features to other 2D game genres. An example is the application of effects to a project with a top-down camera (Fig. 17) and also in more complex 2D game projects (Fig. 18).

We also observed some limitations of the tool throughout the tests. One of the test projects contained a centralized entity responsible for reproducing and operating sounds in the game (a sound manager object). This entity, in general, produces a sound when an event occurs but does not associate it with a specific object (emission source). In this situation, UtopicSense is not practical since the object's location is essential for the tool to emit particles from a sound source. Another limitation found during the tests is that the program can have problems if the emission speed is low in short sounds (about 0.2 seconds long or shorter).

After performing the testing process, we evaluated UtopicSense with potential users to assess the tool in terms of usability.

VI. Evaluation

We conducted a preliminary evaluation of UtopicSense based on the user evaluation methodology in a controlled environment [35]. We used this method to characterize and evaluate the tool's usability from the perspective of game programmers using Unity.

Four users participated in the evaluation (U1, U2, U3, and U4). They had at least one project already completed at Unity and nice English reading skills. Among the participants, two were students in the Computing field, one was graduated and worked with information technology, and the other was a graphics designer. The analysis we conducted provided primarily qualitative data, reflected by impressions and reactions of the participants towards the tool. We also collected quantitative data based on the tasks performed using the software (completed without error, with error, and not completed). Each participant during the test had to perform tasks shown in Table III.

Users executed tasks in three different projects. Each participant had time to read the assignment and ask questions about them. After that, they had to announce when they started and ended each task through verbal expression. The maximum duration of the evaluation was 40 minutes. At the end of the assessment, each participant answered a post-test questionnaire and expressed their opinions and suggestions for improving UtopicSense.

While conducting the assessment, each of the four users performed 17 tasks. The evaluation includes a total of 68 tasks executed by users. Fig. 19 illustrates the proportion of tests completed without mistakes, with mistakes, and not completed. We can observe that users could succeed in many tasks without errors (85.3%, corresponding to 58 tests) and concluded a lower percentage of tasks with mistakes (14.7%, corresponding to 10 tests). There were no tasks without

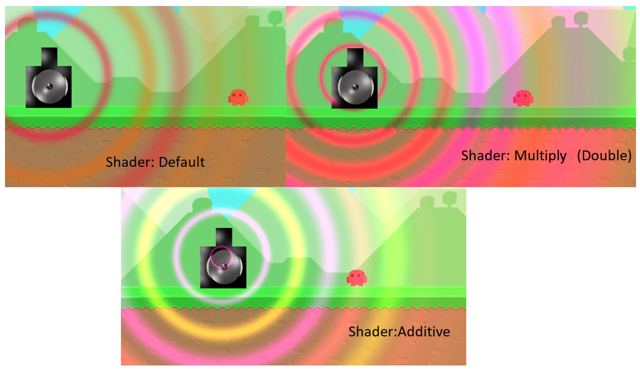


Fig. 15. UtopicSense shaders usage.

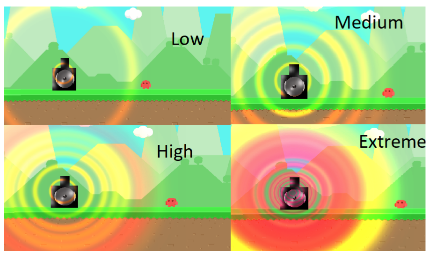


Fig. 16. Particle emission speed.

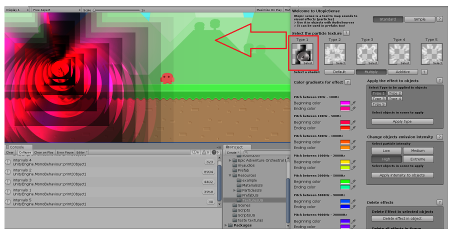


Fig. 14. Texture applied to sound effect.

Fig. 17. UtopicSense applied to a 2D topdown game.

Table III
TASKS USERS EXECUTED IN UTOPICSENSE DURING ITS EVALUATION

T1 - Open the UtopicSense tool
T2 – Apply effect type 1 in an object
T3 – Make the effect range bigger
T4 – Change the effect speed in an object
T5 – Apply the effect in other two different objects
T6 – Change the effect type
T7 – Change the texture related to an effect type
T8 – Change effect colors
T9 – Change the shader applied to the effect
T10 – Delete the effect in only two objects in the scene
T11 – Change the colors back to the default ones
T12 – Change the tool into simplified mode
T13 – Delete all effects in the scene
T14 – Apply the effect in the controlled character
T15 – Apply the visual effect in the bombs
T16 – Turn down the volume effect in the bombs
T17 – Apply the effect in the “Spitter” enemy

Table IV
INDEX OF TASK EXECUTION RESULTS

Tasks	U1	U2	U3	U4
T1	🟩	🟩	🟩	🟨
T2	🟩	🟩	🟩	🟩
T3	🟨	🟨	🟩	🟩
T4	🟩	🟨	🟩	🟨
T5	🟩	🟩	🟩	🟩
T6	🟩	🟩	🟩	🟩
T7	🟩	🟩	🟩	🟩
T8	🟩	🟩	🟩	🟩
T9	🟩	🟩	🟩	🟩
T10	🟩	🟩	🟩	🟩
T11	🟩	🟩	🟩	🟩
T12	🟩	🟩	🟩	🟩
T13	🟩	🟩	🟩	🟩
T14	🟩	🟩	🟩	🟩
T15	🟨	🟩	🟩	🟩
T16	🟩	🟩	🟩	🟨
T17	🟨	🟨	🟨	🟩

Legend: 🟩 No mistakes in completing the task
🟨 Task completed with mistakes

completion during the evaluation process.

Table IV specifies tasks completed with no mistakes and with mistakes for each user. This table shows which tasks concentrated the most errors on their execution.

Task 17 (T17 - Apply the effect in the “Spitter” enemy) had a greater incidence of conclusion with a mistake. Tasks 3 and 4 (T3 - Make the effect range bigger, T4 - Change the effect speed in an object) also added to the number of mistakes, with half of the participants making mistakes on the task execution.

The errors in Task 17 allowed us to identify the need to add more messages for the user to better recover from mistakes. While performing the task, the tool did not inform users if they applied the effect on objects that did not have an `AudioSource` attached. Players tended to run the project multiple times to see if the effect worked until they found out that they had just added the effect in an object without an `AudioSource` component.

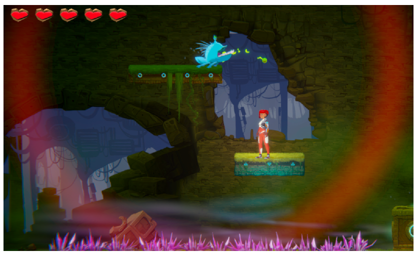


Fig. 18. UtopicSense applied to a 2D complex scenario.

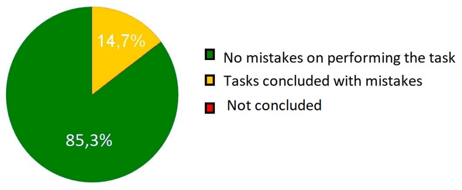


Fig. 19. Ratio of task execution results in the user evaluation.

In Task 3, users, in general, tried to complete the task by changing the effect’s speed emission first before figuring out they had not changed the range with that option. In the questionnaire answered by the participants, the results were as follows:

- 3 of 4 users considered that the tool completely met the requirement of being easy to learn. Only one considered that it partially complied and recommended the use of more messages to recover from errors.
- All users found that the tool met the requirements for ease of use, flexibility, productivity, satisfaction, and usefulness.
- Regarding the safety of the tool for the end user’s (player’s) health, 2 of the participants considered that this analysis does not apply to the situation of UtopicSense and the others stressed the importance of making it clear somewhere in the tool about the association of specific color patterns to epilepsy.

Based on these results, we incorporated a warning about specific color patterns in epilepsy cases. We highlight this subject in the help window in the tool’s color section.

In general, the participants expressed positive reactions towards the tool, describing it as intuitive. All users tended to complete tasks already involving subsequent steps before being asked to do them. This situation allowed the users to build prior knowledge about some features before using them in a subsequent task. This situation indicates that theinterface instigated curiosity and the software functions complement each other.

On the other hand, users also recommended some modifications to favor the tool’s communicability. Some UtopicSense’s section titles could be more clearly associated with correlated functionality. An example of this situation was the suggestion to change the word “Type”, which defines effect types to the term “Effect”, in addition to the idea to change “Change Effect

Intensity" to "Change Effect Speed."

Other problems frequently observed resulted in more messages to help avoid mistakes in the tool. An example of this situation is that most users tended to apply effects to objects that did not have an `AudioSource` component. Users also found it difficult to perceive some features of the tool that involved changing Unity's parameters outside the tool's interface, such as the maximum distance of a source.

VII. Conclusion

The objective of this work was to build and present a tool to promote the use of synesthesia as an assistive resource in 2D games inside Unity. By using UtopicSense, Unity game developers can leverage the resources implemented in the tool to add algorithmic synesthesia to their projects without the need for programming. Following the proposed methodology to create the tool, we first gathered requirements based on the observation of aspects to improve accessibility in existing 2D games. After that we implemented a tool prototype with base functions for the synesthesia approach. Subsequently, we added an interface with more features and control of the parameters to allow the customization of the generated visual effects. We then tested the tool in different game projects and conducted a preliminary user test focused on the usability of UtopicSense. From such tests, we identified improvements and limitations. Many test results will also serve as input for the developing new features and future corrections.

This work brings contributions such as the reflection on problem occurrences that can compromise the accessibility of deaf and hard-of-hearing people in 2D games. The most prominent contribution is the possibility, through UtopicSense, of improving accessibility in games. The tool can also contribute to studies and analyses related to the mapping of sound signals into images (e.g., analysis of frequency incidence in audio). In addition, the availability of the tool enables new studies on the effectiveness and efficiency of synesthesia to communicate sound information in games.

As future work, it is intended to increase functionality in the tool and extend its application to 3D games, which have more weaknesses regarding accessibility for people with some hearing impairment. Also, we will implement gradual improvements in the usability and communicability aspects of UtopicSense to make the application of visual effects in the tool even more flexible and increasingly attractive to developers. We will also use the tool to study the use of synesthesia as an assistive resource with the aid of deaf people and people with hearing impairment to improve the tool based on the perspective of this audience.

References

[1] A. Carmo, "Esporte, lazer e os "deficientes"," *Deficiência física: a sociedade brasileira cria, recupera e discrimina*, pp. 127–156, 1991.

[2] K. Bierre, M. Hinn, T. Martin, M. McIntosh, T. Snider, K. Stone, and T. Westin, "Accessibility in games: Motivations and approaches," *White paper, International Game Developers Association (IGDA)*, 2004.

[3] A. S. Bastos, R. F. Gomes, C. C. dos Santos, and J. G. R. Maia, "Synesthesia: A study on immersive features of electronic games," *SBC Journal on Interactive Systems*, vol. 9, no. 2, pp. 38–51, 2018.

[4] F. R. S. Coutinho, "Revisiting game accessibility for deaf and hard of hearing players," 2012, [Master thesis, UFMG].

[5] C. L. Siying Liang, "Synesthesia and its implications on video game design," *Design Engineering*, pp. 995 – 1002, Dec. 2020. [Online]. Available: http://thedesignengineering.com/index.php/DE/article/view/1136

[6] GameDesign, "The top 10 video game engines," https://www.gamedesigning.org/career/video-game-engines/, 2018, access: 2018-12-03.

[7] F. Tanant, "The top 10 video game engines," https://www.websitetooltester.com/en/blog/best-game-engine/, 2018, access: 2018-12-03.

[8] Unity, "Games made with unity," https://unity3d.com/pt/games-made-with-unity, 2018, access: 2018-12-03.

[9] S. R. Basbaum, "Synesthesia and digital perception," *Subtle Technologies Festival, Toronto*, pp. 01–20, 2003.

[10] ——, *Sinestesia, arte e tecnologia.* Annablume, 2002, vol. 173.

[11] R. T. Dean, M. Whitelaw, H. Smith, and D. Worrall, "The mirage of real-time algorithmic synaesthesia: Some compositional mechanisms and research agendas in computer music and sonification," *Contemporary Music Review*, vol. 25, no. 4, pp. 311–326, 2006.

[12] N. Sagiv, R. T. Dean, and F. Bailes, *Algorithmic synesthesia.* na, 2009.

[13] A. S. Bastos, R. F. Gomes, C. C. dos Santos, and J. G. R. Maia, "Synesthesia: A study on immersive features of electronic games," *SBC Journal on Interactive Systems*, vol. 9, no. 2, pp. 38–51, 2018.

[14] *Journey*, (2012) Thatgamecompany. [Online]. Available: http://thatgamecompany.com/journey/

[15] J. M. E. d. Silva, A. d. Castro Callado, and P. M. Jucá, "Representing sentiment using colors and particles to provide accessibility for deaf and hard of hearing players," 2018.

[16] F. Coutinho, R. O. Prates, and L. Chaimowicz, "An analysis of information conveyed through audio in an FPS game and its impact on deaf players experience," in *2011 Brazilian Symposium on Games and Digital Entertainment.* IEEE, 2011, pp. 53–62.

[17] *The Legend of Zelda: link to the Past*, (1991) Nintendo, [SNES Cartridge].

[18] *Limbo*, (2010) Playdead. [Online]. Available: http://playdead.com/games/limbo

[19] *Mark of the Ninja*, (2012) Klei Entertainment. [Online]. Available: https://www.klei.com/games/mark-ninja

[20] *Ori and the Blind Forest*, (2015) Moon Studios. [Online]. Available: http://www.oriblindforest.com/

[21] *Rayman Origins*, (2011) Ubisoft. [Online]. Available: https://www.ubisoft.com/en-gb/game/rayman/origins

[22] *Rayman Legends*, (2013) Ubisoft. [Online]. Available: https://www.ubisoft.com/pt-br/game/rayman/legends

[23] *Shadow Dancer*, (1989) Sega, [Sega Genesis Cartridge].

[24] *Sonic Mania*, (2017) Sega. [Online]. Available: https://www.sega.com/games/sonicmania

[25] *Sonic the Hedgehog*, (1991) Sega, [Sega Genesis Cartridge].

[26] *Sonic the Hedgehog 2*, (1992) Sega, [Sega Genesis Cartridge].

[27] *Streets of Rage*, (1991) Sega, [Sega Genesis Cartridge].

[28] B. Yuan, E. Folmer, and F. C. Harris, "Game accessibility: a survey," *Universal Access in the Information Society*, vol. 10, no. 1, pp. 81–100, 2011.

[29] D. N. Nogueira, F. R. Coutinho, W. Soares Jr, R. O. Prates, and L. Chaimowicz, "Analyzing the use of sounds in fps games and its impact for hearing impaired users," *Proceedings of SBGames SBC*, pp. 127–133, 2012.

[30] R. da Silva Lima *et al.*, "Da nota ao som: explorando territorios harmonicos," Ph.D. dissertation, Universidade Estadual de Campinas (UNICAMP). Instituto de Artes, 2009.

[31] F. J. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978.

[32] J. S. Bendat and A. G. Piersol, *Random data: analysis and measurement procedures.* John Wiley & Sons, 2011, vol. 729.

[33] C. Roads and J. Strawn, *The computer music tutorial.* MIT press, 1996.

[34] D. Hammersho/i and H. Mo/ller, "Sound transmission to and within the human ear canal," *The Journal of the Acoustical Society of America*, vol. 100, no. 1, pp. 408–427, 1996.

[35] S. Barbosa and B. Silva, *Interação humano-computador.* Elsevier Brasil, 2010.