BSc Individual Project
2019-2020

Image-Sound Conversion Mobile Application ArtSense

Mahum Hashmi
1707898

Supervised by Dr. Alfie Abdul-Rahman

23.04.2020

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 25,000 words.

Mahum Hashmi
23.04.2020

Abstract

Music and art are parts of life that we simply cannot ignore, as they have proven to be a means of education, meditation and innovation. Unfortunately, there is a demographic of people unable to reap the benefits of these art forms in a conventional manner, due to physical disabilities they are either born with, or have acquired after experiencing trauma. Given how rapidly technology is transforming the world around us, there is a way to help these users access art in a more convenient, albeit unconventional manner.

ArtSense aims to address this issue by scientifically converting a visual artistic experience into an aural one, without forgoing the emotion and nuance of the original for a clinical text description of its subject matter. The main functionality of this iOS mobile application is that the user is able to take a picture, in which each pixel represents a certain colour. The pixel colour data is then collected, and following Sir Isaac Newton's colour wheel theory of linking colours to musical notes, a frequency is assigned to each pixel. The user is then able to touch different parts of their image and hear the corresponding frequency.

The application has been designed ensure the interface meets the special needs of visually-impaired users. Avenues for further development for use in more complex contexts or for more sophisticated functionality have also been explored as part of this project.

Acknowledgements

I would like to sincerely thank my supervisor, Dr. Alfie Abdul-Rahman, for her tireless efforts in guiding me to put forth my best and most innovative ideas. Without her constant support, this project would not have been possible.

I would also like to thank my sister Heba Hashmi for her assistance in editing and proofreading the final report. Finally, I would like to thank my parents for providing me with endless support and opportunities to pursue my interests.

Contents

Chapter 1 | Introduction

1.1 Introduction: An artistic experience for the visually impaired

It is art through which humans are able to express their imagination, conceptual ideas or technical skills by creating visual, auditory or performance artefacts, with the intention of being interpreted based on their beauty and emotional power. Art is also a more abstract method of beginning dialogues on various economic, political and social issues. However, art is often designed to be perceived through a compound amalgamation of senses, and does not always cater to those whose senses are compromised by a disability they are either born with or have acquired after experiencing trauma, causing them to miss out on a mainstream artistic experience. How, for example, would a hearing-impaired individual relax to a melodious piece of music, or enjoy a theatre performance? How would a visually-impaired individual appreciate the works of renowned artists like Van Gogh or Da Vinci?

1.2 Aim: Converting image to sound while retaining the artistic experience

ArtSense attempts to bridge this gap by scientifically converting a visual artistic experience into an aural one, without forgoing the emotion and nuance of the original for a clinical text description of its subject matter. The main functionality of this iOS mobile application is that the user is able to take a picture, in which each pixel represents a certain colour. The pixel colour data is then collected, and following Sir Isaac Newton's colour wheel theory[Figure 2.3] of linking colours to musical notes, a frequency is assigned to each pixel. The user is then able to touch different parts of their image and hear the corresponding frequency. This allows them to experience a visual piece as sound, much like braille allows them to 'see' through touch. The tactile function allows the user agency over their experience, as compared to other applications that simply play the multiple frequencies of an image back automatically, once the image is processed.

For example, a visually-impaired user goes to an art gallery, and takes a photo of a painting with varying shades and tones. Once their photo is processed, as they move their finger over the image, they hear different frequencies. Darker shades have a relatively lower frequency, whereas lighter and brighter shades have a higher frequency. Although they cannot see the image, the user can still get a good idea of what shades and tones the painting consists of, taking them on a journey where they can hear every tone of the painting. The beauty of any artwork is that it is open to interpretation in every sense. This application will allow the user to engage with the artwork in a more intimate manner, and experience its nuances and dimensions in a way that able-bodied users may not even consider.

A key element of any work of art is the emotion or feeling that the artist is trying to evoke, which is communicated through a combination of composition, colour choice, brushwork, use of negative space, etc. An example of this can be Pietro da Cortona's painting, *Rape of the Sabines*. The painting describes a scene in which the Roman ruler Romulus has issued the abduction of the Sabine women, in order to support the growth of the new city of Rome, which was predominantly male at the time.

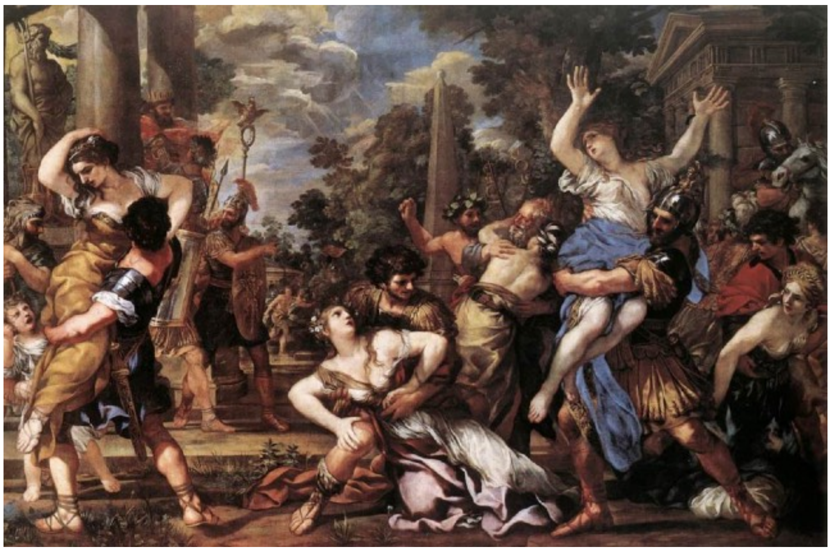


Figure 1.1 Pietro da Cortona: Rape of the Sabines

As can be seen in Figure 1.1, the painting conveys immense chaos and pain. While these emotions can be discerned through the multiple visual factors described above, when you hear the erratic and relatively lower frequencies generated by this image, the chaos and pain resonates with you in a way you would not experience by simply looking at the painting. This is more effective and immersive than a simple clinical description of the subject matter, a method often used in museums and galleries to assist visually-impaired visitors.

1.3 Objectives

In order to achieve the aims set out above, the following objectives form the core of this research project:

- To create an artistic experience for visually-impaired users by converting full-colour images to audio frequencies
- To understand the theoretical foundation that underpins the scientific link between colour, tone, hue and audio frequencies, in order to better inform the functionality of the mobile application
- To ensure the interface of the mobile application is designed to meet the special needs of visually-impaired users
- To allow for further development of the mobile application, for use in more complex contexts or for more sophisticated functionality

1.4 Structure of Report

This report mirrors the project objectives outlined above, and follows a systematic explanation of the different phases of development of ArtSense.

Chapter 2 contains an in-depth research study that shaped the parameters of the application, and includes a discussion of existing mobile applications that inspired this project, as well as an explanation of the physics and algorithms developed by other scientists, through which the implementation of this application has been developed. It also provides a theoretical background to the link between colour and sound, and offers an explanation for the models that were followed, as opposed to those not taken into consideration.

Chapter 3 highlights the initial requirements, both functional and non-functional, along with a use case and state diagram of the whole application.

Since this project is geared towards a very specific type of user, the design and user interface had to be taken into special consideration. Chapter 4 focuses on the design aspects of the application, explaining the choices of layout, colour, placement of buttons and so on, showing the initial plans and incremental display of the layout.

Chapter 5 explains the implementation of how an image is prepared before the physics and algorithms can be applied for the conversion, and how that conversion is then communicated to the user.

Chapter 6 poses an evaluation, and recounts the testing and feedback sessions with users who interacted with the application.

Chapter 7 highlights some of the legal and ethical concerns associated with this project.

Chapter 8 is the conclusion and also discusses the future prospects this project holds, with a discussion of how this application can be taken further to broaden its functionalities to cater to a wider audience.

Chapter 2 | Review

2.1 Background

2.1.1 Existing Applications

Preliminary research on this project involved scoping existing applications that converted images to different forms of audio output, for example, speech, music or basic frequencies.

The field of Computer Vision deals with processing all aspects of an image. This, paired with Artificial Intelligence (AI), then outputs information relevant to an application. Typically, neural networks[1] are used to gather pixel data, after which the networks are fed as 'pre-labelled images' in order to teach AI how to recognise images. A more complex version of this opens the field of 'object recognition and tracking', where the learned information about an object in an image is outputted in different forms.

TapTapSee[2] and Aipoly Vision[3] are two mobile applications that adopt the method of object recognition. In TapTapSee[2], the user takes a photo and after processing, with the help of the VoiceOver setting, a description of the recognised object is read out to the user. Aipoly[3] adopts a similar methodology, only once the photo is taken the user must select a category the image is most likely to fall in. Due to this added interaction in Aipoly[3], we decided to eliminate this application from consideration, and went ahead with further research into TapTapSee[2]. This is because we must keep in mind that our target is a visually-impaired user and therefore our aim is to limit interaction with the screen, for example selecting from a range of different buttons.

Further research into the basic functionality of TapTapSee[2], which was to gather information about an image and output it in the form of varying signals, led to the study of another application named PixelSynth[4], developed by Olivia Jack. In this application, the user can either experiment with pre-existing images, or upload their own image, which is then converted to greyscale, and processed to output frequencies in a melodic way. Since this application outputs a melodious sound rather than speech, a lot of music theory is applied to its functionality, in terms of assigning particular musical notes to different grey tones. Users are given various controls to alter the speed of sound, scale, octave, start note, etc. There is a cursor that moves from end to end over the entire image, which processes and outputs audio of the pixels that are within the parameters of the cursor. Along with adjusting the sound, they can also annotate and make real-time adjustments to the brightness, contrast and orientation of the image, all of which affect the output sound.

While PixelSynth[4] offers the user greater control over the image that is being processed into sound, there are three principle drawbacks that hinder the experience for a visually-impaired user. Firstly, the use of greyscale images prevents any correlation between music and colour from being explored in the audio output. Secondly, as the user simply has to listen to the output being produced, they have little agency over the playback process and how it relates to the converted image. Thirdly, while the wide range of image controls make the application more interesting, it is largely inapplicable to a visually-impaired user.

In an attempt to adapt this experience for its target audience, ArtSense builds on this basic idea of outputting audio based on pixel data, by assigning frequencies to multiple tones of different colours, as opposed to those of a greyscale image, making the output experience more diverse. It also allows the user to use their fingers as a 'cursor' as it moves over the image, creating a more interactive and tactile experience. Finally, it offers the user more control over the sound functionality, rather than the image itself, to make the experience better suited to the visually-impaired. Details of this implementation are discussed in Chapter 5.

2.1.2 Theoretical Background

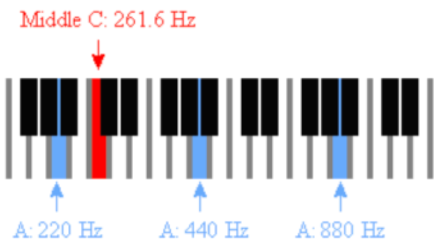
Alan Lundgard[6], when discussing the socioethical considerations for accessible visualisation design explain that there are two models to consider. The 'medical model[6]' and the 'social model[6]'. The medical model is one that deals with physical or psychological state of a person, and aims to cure them of any disability via medical intervention, even if it 'does not cause pain, illness or death (eg. deafness)[6]'. The social model, on the other hand, aims to address a *disability*, which has more social, economic and political connotations. The study of the social model is more relevant for this project as it helps in coming up with technological solutions that address problems of the 'infrastructural barriers that PWD (people with disabilities) experience in their everyday lives[6]'.

The social model seems to be a widely adopted ideology for developers in this field, as Cazan[5] mentions that 'image-sound conversion applications are simple, as they are not invasive in any way', as opposed to medical intervention such as tactile imaginators, retina implants or brain surgery. This project does not aim to cure the disability, but rather find new applications of it, in a way to help them use their disability to learn something new about art, and experience different factors of it in a more unconventional way.

2.1.3 Defining Frequencies

Before diving into the physics and algorithms of image-sound conversion, we must first consider the capabilities of the human ear. According to Drd. Alexandru Cazan[5], the 'frequency of sounds perceived by the ear ranges from 20 to 20,000 cycles/second (Hz). The ear is a complex structure that completes filtering, frequency spectral analysis and signal compression - all of these before transmitting the information to the brain'. In ArtSense, a similar analysis is done after the user inputs an image which has either been captured, or uploaded. However this analysis also needs to consider the correct range of frequency it transmits, so that when the audio is outputted to the user, it is heard in a comfortable and noninvasive manner. Cazan[5] also states that 'sight contributes up to 90% of the sum of information that we have about the surrounding world, so it is an important physiology aspect, not only in differentiating light, shapes or object's colours, but also in maintaining the equilibrium and cortical tonus (the attention)'. This fact should help developers realise how nuanced some application details will end up being.

The twelve musical notes are C, C#, D, D#, E, F, F#, G, G#, A, Bb and B. These twelve notes are repeated in different octaves which varies their frequency (more details of octaves are discussed ahead in this chapter). Therefore, when selecting a frequency of a single note to be outputted, we are given a choice of which octave we are selecting the note from. The following figure shows some varying octaves and the note frequencies:



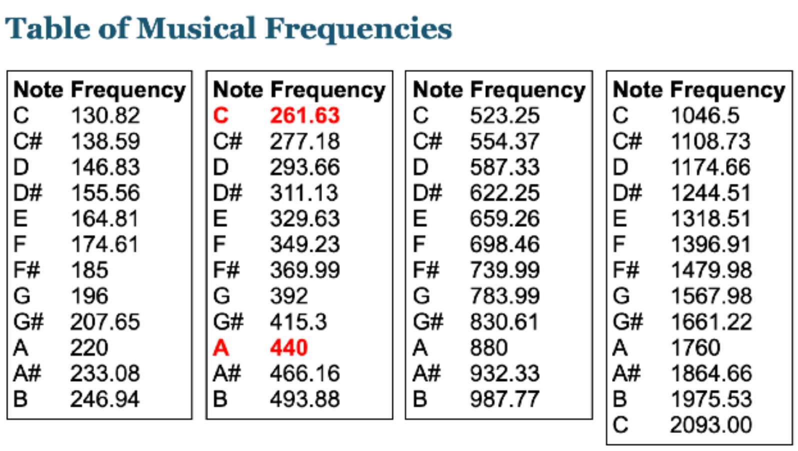


Table of Musical Frequencies

Note	Frequency	Note	Frequency	Note	Frequency	Note	Frequency
C	130.82	C	261.63	C	523.25	C	1046.5
C#	138.59	C#	277.18	C#	554.37	C#	1108.73
D	146.83	D	293.66	D	587.33	D	1174.66
D#	155.56	D#	311.13	D#	622.25	D#	1244.51
E	164.81	E	329.63	E	659.26	E	1318.51
F	174.61	F	349.23	F	698.46	F	1396.91
F#	185	F#	369.99	F#	739.99	F#	1479.98
G	196	G	392	G	783.99	G	1567.98
G#	207.65	G#	415.3	G#	830.61	G#	1661.22
A	220	A	440	A	880	A	1760
A#	233.08	A#	466.16	A#	932.33	A#	1864.66
B	246.94	B	493.88	B	987.77	B	1975.53
						C	2093.00

Figure 2.1 Frequencies of notes in different octaves

As can be seen in figure 2.1, the note C's frequency ranges from 130.82 Hz all the way to 1046.5 Hz. These are well within the capable hearing range, so any of them can be selected for ArtSense. However, it is more a question of which is more pleasing to hear. If we chose to go with the higher frequencies, it should be noted that the entire audio output the program transmits will lie in that higher octave. If it is a very populated image in terms of colour, there will be many high pitched notes heard, which would not be very pleasing to hear all together. This is why the octave chosen for the development of this project lies in the lower medium range. Using octaves to make the output audio more diverse is discussed in chapter 7.2 of this report.

2.1.4 Mapping Pixels to Sound (Greyscale Images)

Having the range of frequencies as a parameter, we can move on to processing pixel data. There have been various implementations developed for processing pixel data. Before working with coloured images, it was necessary to first examine the algorithms for greyscale images. Xuan Zhang[7] and her colleagues wrote a brilliant paper that explains an updated formula on the original 'vOICe System' - which is a vision aid technology that uses the Discrete Fourier Transform (DFT). Although the original vOICe System is simple and convenient, it is computationally very complex. Zhang[7] describes an updated method which uses the Inverse Fast Fourier Transform (IFFT), to make image-sound conversion more efficient.

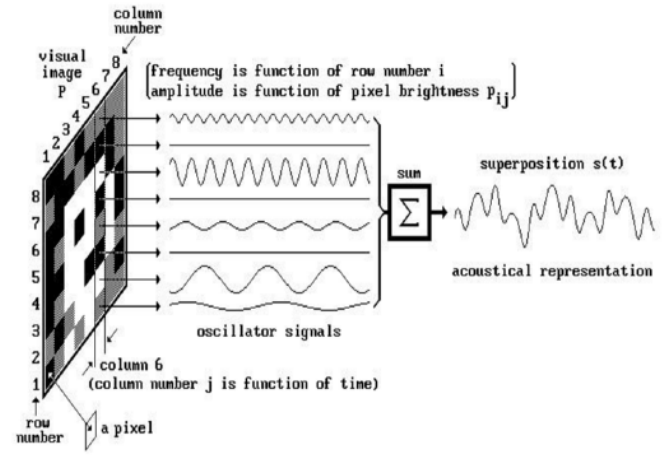

column number
visual image P
frequency is function of row number i
amplitude is function of pixel brightness p_{ij}
sum
superposition s(t)
acoustical representation
oscillator signals
column 6
(column number j is function of time)
row number
a pixel

Figure 2.2 The image-sound conversion principle diagram of the vOICe system

Figure 2.2[7] shows how pixels are mapped to sound. For each pixel, the vertical position is mapped to frequency, the horizontal position into time, and the brightness into oscillation amplitude. For a given column j, every pixel in the column generates a sinusoid in the audible range of 20 Hz to 20,000 Hz. The more number of columns there are, the longer the length of time. Brightness is represented in grey tones - which range from 0-255 in which white (255) corresponds to a maximum of 1, and black (0) which corresponds to a minimum value of 0 - which implies growth of amplitude of the oscillating wave when it gets larger. PixelSynth[4] also uses this concept for processing the grey tones. Every pixel is assigned a value between 0 and 255, and the assigned frequency varies accordingly. Taking rows, columns (number of pixels) and frequency into account, the final mapping results of the whole image are calculated using the following formula:

$$f^i(n) = \sum_{m=1}^{N} a_m^i \cos\frac{Kmn}{N}$$ **Equation 1.1**

where K is a constant and N is the maximum number of rows, and a_m^i denotes pixel values in the column *m* and row *j*. This formula shows calculations using DFT.

The next step is to convert DTF to IDFT, which is the Inverse Discrete Fourier Transform, which simulates the direct synthesis of the vOICe system. IDFT is defined as:

$$x(n) = \frac{1}{N}\sum_{m=0}^{N-1} X(m)\left[\cos(2\pi mn/N) + j\sin(2\pi mn/N)\right]$$ **Equation 1.2**

where *x*(*n*) is the *n*th IDFT output, *n* is the index of output sequence in the time domain (*n*=0, 1, …, *N*-1), *X*(*m*) is the input sample sequence, *m* is the index of input sample sequence in the frequency domain (*m*=0, 1, …, *N*-1), *N* is the total number of sampling values in the discrete sequence. Zhang[7] shows proofs which demonstrate that the calculations of DFT and IDFT are consistent.

IFFT is used to improve efficiency of output calculations using the above mentioned formulas. Details of multiple equation manipulations are explained by Zhang[7] to reach the final formula with which calculations using Discrete Fourier Transform and Fast Fourier Transform are done using:

$$\frac{N^2+N(N-1)}{\frac{N}{2}\log_2 N+N\log_2 N}\approx\frac{2N^2}{\frac{N}{2}\log_2 N}\approx\frac{N^2}{\frac{N}{2}\log_2 N} \qquad \textbf{Equation 1.3}$$

presuming the sequence we have taken is N points of finite length. This shows that by using IFFT instead of the direct synthesis process, the operation speed is improved, computational complexity is reduced and the overall performance of the system is improved.

The Fast Fourier Transform is used to characterise the magnitude and phase of a signal, and this is used in ArtSense to calculate the amplitude that is displayed to the user. However, as efficient as this method of IFFT is, it is not used in ArtSense to calculate frequency, as there are set frequencies already in place that are extracted from Pridmore's[8] colour wheel, details of which are discussed below.

2.1.5 The Colour Wheel

Figure 2.3 shows Newton's original colour wheel (1704) in which he correlated musical notes with colour, beginning with red and dividing the wheel to represent the scale starting with D and ending with the octave of D.

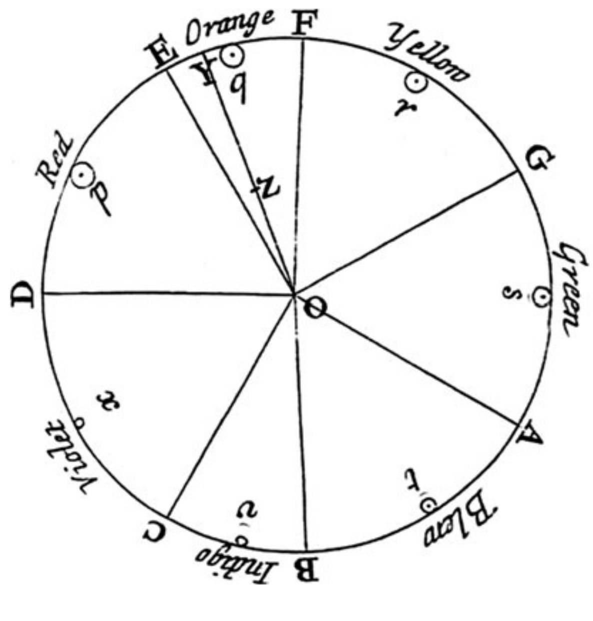


Figure 2.3 Newton's Colour Wheel - Opticks (1704)

Newton first passed a white light through a prism which fanned out into a rainbow, through which he identified seven consistent colours: red, orange, yellow, green, blue, indigo and violet. This finding was based on his idea that the colours of a rainbow were analogous to the notes of the musical scale. Newton introduced two colours, orange and indigo which corresponded to half steps in the octatonic scale, however, whether he 'delineated indigo and orange on the wall or whether Newton added those colors to his diagrams in order to better fit his analogy is unclear[18]', says Peter Pesic, physicist, pianist and author of *Music and the Making of Modern Science.*

As Newton consistently worked on his theory, he later noted that 'according to the relationship between radii of colored rings, the range from red to violet was equivalent not to an octave, but to something more like a major sixth[19]', but persisted in his theory that a major sixth was still equivalent to an octave. However, as musicians and physicists today know, a major sixth and an octave are not the same thing. The octave cycle is physically a range of frequencies from *x* to 2*x* (for example, middle C has a frequency of 262 Hz and C has one of 524 Hz), and their wavelengths are inversely proportional to their frequencies. However, 'the wavelengths of visible light range from 400 to 700nm, which if translated to sound waves would be approximately equivalent to a major sixth[19]'. Despite the fact that Newton's colour and music theory was not successful, many scientists still did further research into the subject and came to their own conclusions.

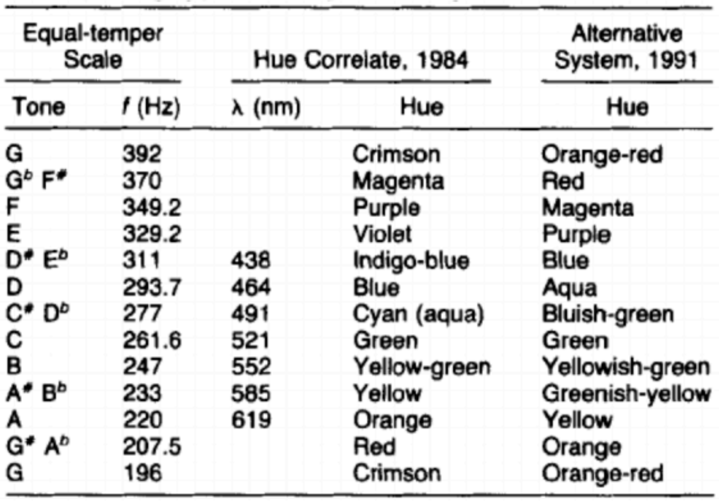
The relationship between colour and sound fascinated many minds, including composers like Beethoven and Rimsky-Korsakof. According to Pridmore, 'the best known modern attempt at correlation is Walt Disney's, relating music to color patterns in his classic film Fantasia (1940)[8]'. Rachel Sebba[9] of Technicon (Israel Institute of Technology, Haifa - Technicon) conducted an experiment in which her students rendered short pieces of music (four bars, within a very limited range of 10-13 seminotes) as painted colours. It was noted that brighter and lighter colours like yellow and white, corresponded with higher frequencies. Pridmore[8] goes on to explain that 'tone comprises some nine cycles (octaves), each repeating the same 12 semitones, whereas hue is limited to only one cycle of hues'. He also explains that although Sebba's[9] experiment shows a correlation between sound and colour, 'it does not stand expansion to the gamut of semitones[8]'.

From a psychophysical perspective, the correlation between sound and colour should be derived from a physical stimuli, in which there are two variables. The first is amplitude, which causes the loudness or brightness/lightness, and the second is the wavelength which causes a musical note or hue. Pridmore[8] explains that 'correlation of tone and hue is also indicated by the cyclic nature of each, as octave cycle and hue cycle (or color wheel)'. Psychologically, the octave cycle is defined as the 'range of tones from a given tone to the same tone repeated[8]'. The hue cycle is defined only psychologically because it does not comprise of a physical octave, but rather a spectrum from 400-700 nm, and the nonspectral purples (compound colours). As mentioned before, the normal audible hearing range is between 20 Hz to 20,000 Hz, but at the ends of the spectrum (red and violet), the frequency falls below our sensitivity.

Pridmore[8] aptly summarises the main differences between light and sound as '(a) one hue cycle in contrast to nine or ten octave cycles; (b) hues mix completely (to form new hues) whilst tones do not (they remain semidiscrete as chords); and (c) the hue cycle varies in amplitude (brightness/lightness maxima at yellow, cyan and magenta) whilst the octave cycle is uniform in amplitude (loudness), per unit of radiant or sonic flux'. The correlation between hue and tone can be calculated using the formula:

$$\lambda = C / f \qquad \textbf{Equation 1.4}$$

where λ is wavelength measured in nm and limited to optimum colour stimuli, C is the speed of light (2.997 x 10^8 m/s), and *f* is the frequency in Hz. When calculated in 1984, the table displayed in Figure 2.4[8] was produced, which led to the final version of the colour wheel as shown in Figure 2.5[8]. This colour wheel is what the frequency assignments in ArtSense is based on.



Equal-temper Scale		Hue Correlate, 1984		Alternative System, 1991
Tone	*f* (Hz)	λ (nm)	Hue	Hue
G	392		Crimson	Orange-red
G^b $F^\#$	370		Magenta	Red
F	349.2		Purple	Magenta
E	329.2		Violet	Purple
$D^\#$ E^b	311	438	Indigo-blue	Blue
D	293.7	464	Blue	Aqua
$C^\#$ D^b	277	491	Cyan (aqua)	Bluish-green
C	261.6	521	Green	Green
B	247	552	Yellow-green	Yellowish-green
$A^\#$ B^b	233	585	Yellow	Greenish-yellow
A	220	619	Orange	Yellow
$G^\#$ A^b	207.5		Red	Orange
G	196		Crimson	Orange-red

Figure 2.4 Correlation between hue and tone in the equal-temper scale

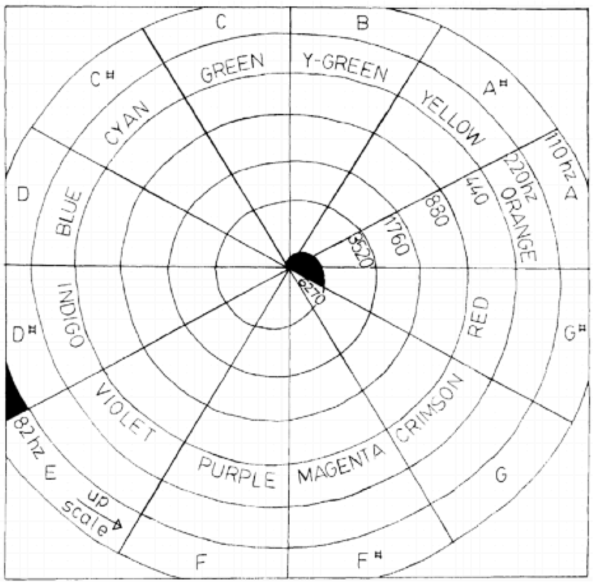


Figure 2.5 Layout of final (1984) display panel

In conclusion, while the techniques highlighted by Zhang[7] and implemented by Jack[4] are precise ways to calculate pixel data, they work with greyscale images. Since the premise of ArtSense is to convert coloured images, these techniques were ultimately not applied during the implementation process. As shown in Figure 2.5[8], Pridmore's colour wheel was more relevant for assigning frequencies in ArtSense, based on the colour value extracted from each pixel. This process is described in detail in Chapter 5. However, future development of this application, which involves collecting sample values using techniques that also incorporate the IFFT, can also be carried out. This is discussed in Chapter 8.

2.2 Socio-ethical Considerations

Given that a major part of technological research and development is geared towards optimising user interface and user experience, strict ethical and professional frameworks have been designed to govern all interaction with and data collection from a target audience. These parameters have been especially emphasised for interactions with people with disabilities. In his research paper *Please Touch the Art: Experiences in Developing for the Visually Impaired*[10], Iain Emsley highlights the different factors that must be taken into consideration while developing for a disabled user. In his TactilePicture application, 'the major challenge in the project was understanding how the world is perceived with different senses[10]'. He also emphasises the importance of clearly delineating when a user becomes part of a user group. In the case of the visually-impaired, for example, were they born with their impairment, or did they acquire it over time? This helps gather information on the 'level of familiarity and confidence in touch interpretation[10]'.

As mentioned earlier, ArtSense is an application that is mainly to be used in art spaces like galleries and museums. It is important for a developer to understand how to 'improve access to the art collections for the audience and a re-usable technology to deliver audio in a non-linear fashion to the audience within a gallery[10]'. However, keeping in mind that many museums have policies about photography in galleries, there are some legal issues that must be considered as well. More on this is discussed in Chapter 7 of this report. This is also why reading about the TactilePicture[10] project was useful, as its domain and target audience is very similar to that of ArtSense.

An important consideration offered by Emsley is that the 'device needed to allow the visitor to navigate through the audio at their own pace and order of points to be touched, and to find levels of information at their own pace[10]'. Through experimentation, many researchers agreed that 'touch was more nuanced than anticipated[10]', which led to the idea of conducting a touch/pressure focus group experiment, to see which parts of a screen visually-impaired users are more comfortable using. This theory has significantly helped inform the design aspect of ArtSense, and is further discussed in Chapter 4 of this report.

Emsley further states that a 'participatory design approach allowed us to get feedback on how the application might be used and perceived by users[10]', and inspired an agile approach to developing ArtSense, where regular feedback would direct further development. This inclusive approach helps with establishing requirements, need-finding and general design in a far more efficient way. It also encourages interdependence between researchers and stakeholders.

As part of the participatory approach, time and compensation are among the other socio-ethical considerations to be explored, and Lundgard[6] emphasises that any time a disabled person devotes to the project should be compensated for 'at a rate that is greater than that of an average user study participant[6]'. Non-intervention is the practice of asking the question 'whether any technological intervention is appropriate at all?[6]', since there are instances where technological intervention may worsen the problem or be harmful to the participant. It is also important to clearly communicate the expectations of the project to all stakeholders, including the time frame, available resources and whether or not maintenance is required. Given how expensive technological resources are, it should be made clear whether or not specialised equipment is readily available, and how frequently they will be needed for this project and proceed accordingly.

Unfortunately, due to the global pandemic crisis of 2020 (outbreak of Covid-19), the desirable participatory approach was ultimately not a viable option for the development of ArtSense - further details of which are discussed in Chapter 6 of this report. This project therefore had to rely on the 'parachute research' approach, which involves complete disengagement with the target audience after the initial research phase. It is recommended that all future development be carried out using the participatory approach.

Chapter 3 | Requirements

3.1: Requirements and Specification

The basic functionality of this project is that a visually-impaired user should be table to take a photo using the camera of their smartphone, which is then translated to the corresponding audio feedback. Given that the scope of this project was to develop a prototype for ArtSense, it was decided to limit it to an iOS application. As discussed in Chapter 8 of this report, there is a wider of scope of future development of an Android application as well. With iOS development, there is a VoiceOver setting which is an inbuilt system allowing visually-impaired users to hear every icon they press on before selection, in order to guide them.

The following subsections discuss the specific functional and non-functional requirements of ArtSense. It should be noted that these are the initial requirements, which were updated during development, following the agile approach. There are a few requirements that are not completely met by the final version of this project, which are then discussed in Chapter 8 regarding future work. These requirements may not have been met due to technical issues involving frameworks and issues with implementation, and are explained in Chapter 5.

3.1.1: Functional Requirements

1. Users should be able to take a photo with the smartphone's camera

Although ArtSense is primarily a mobile application, it can be further adapted to be installed on devices like iPads or tablets. Basically, it can be used on any device that has a camera, either rear, front or both. The first time the application is installed, it asks the user to allow the application to access the device's camera. In the reverse process of ArtSense (discussed in Chapter 8) which records an audio and converts it to an image, for hearing-impaired users, the application then also requests permission to access the device's microphone - these permissions should also be added to the information property list of the project being developed in Xcode. When run on an iPhone, the format of the camera is the same as the generic iPhone camera, with the button in the lower part of the screen, the flash controls on the top, the 'Cancel' button which allows the user to retake a photo, and the 'Use Photo' button to select the picture taken.

2. Users should be able to enable flash

As mentioned above, when the camera is displayed to the user, the flash settings are visible on top of the screen, providing On/Off controls for the flash. It should be noted that the use of flash will vary the frequency output, as in some cases it does tend to change the tones of the original image, and add shadows too. If the photo is taken in a decently lit space, the flash could tend to make certain tones brighter than necessary, leaving the pixel data captured inconsistent with the actual shade of the corresponding area of the object being captured. However, in cases of a dimly lit setting, the flash will translate the object being captured in a more consistent fashion.

3. Users should be able to upload a photo from their gallery

By selecting the ‘upload photo’ button, instead of a camera, the device’s photo gallery will be displayed to the user. They are then able to select any image from their gallery to be processed. Any form of editing is not enabled in ArtSense. This requirement was added in a later stage of development, after the initial processing methods had been implemented, since the image uploaded is still processed in the same way.

4. Users should be able to receive audio feedback

After a captured or uploaded photo is selected and processed, the user should receive audio output on the last viewcontroller of ArtSense. The volume of this output is according to the current volume of the device, and can be made louder or softer by adjusting the volume as you would normally do on your device.

5. Users should be able to touch converted image and only receive feedback when their touch is within image bounds

After the selected image is processed, it is displayed back to the user in the form of a synthesiser. This means that as the user taps on different parts of the image, they will hear the corresponding frequency. This requirement also states that following the logic of processing the image, audio output should only be heard within image bounds, because outside the bounds, there is no pixel data being processed. In the final version of ArtSense, there are a few problems within implementation (discussed in Chapter 5), causing some audio to be heard outside image bounds, and some audio not heard within the bounds. Due to time constraints, this requirement was not fully met.

6. Audio feedback should be repeatedly given (according to touch)

As many times as the user taps the image, the corresponding frequency should be heard, and it should not only be on the first attempt.

7. Users should be able to update details of the sound (for example chose between Sinusoid oscillation, Sawtooth Oscillation, Triangle Oscillation, Square Oscillation and White Noise Oscillation.)

Initially, ArtSense gave the above mentioned oscillations as options to the user. These different types of oscillations produce a different kind of tone, however the frequency remains constant for each pixel.

8. Users should be able to update details of sound like scale, instrument, octave

When this requirement was initially added to the list, it was considered a tentative one, which would be further judged at a more advanced phase of development. Unfortunately, due to issues with framework implementation (discussed in Chapter 5), this requirement was not met. This states that users should be able to adjust the sound they hear, by changing aspects like starting note, octave, scale, instrument and so on, which was inspired by the controls in the application PixelSynth[4].

3.1.2: Non-functional Requirements

1. **Application should be iOS compatible**

As mentioned above, ArtSense was developed to be an iOS application, however can be taken further to be an Android Application as well, as discussed in Chapter 8. The premise of ArtSense can also be used to create a whole new device that is just used for image-sound conversion (or sound-image), as this would make it cheaper and accessible to those who cannot potentially afford a regular smartphone.

2. **Application should be able to convert input images to sound**

Initially, ArtSense was developed to only convert images to sound, but does have the potential of development to convert sound to images as well (discussed in Chapter 8).

3. **Application should be able to use VoiceOver**

VoiceOver is an iOS feature for visually-impaired users which uses speech to guide the user throughout the device. On a single tap, it tells you which icon you have tapped on, and on a double tap allows you to interact with it. The UIButtons and UILabels are all recognisable by ArtSense, allowing the user to navigate in a way they are familiar with.

4. **Application should not require internet connection**

ArtSense should and does not require any form of internet connection. If a user wants to download an image from the web, they can do so separately, add it to their gallery and upload the image to ArtSense, however this does pose some legal issues that are discussed in Chapter 7.

3.2: Use Case

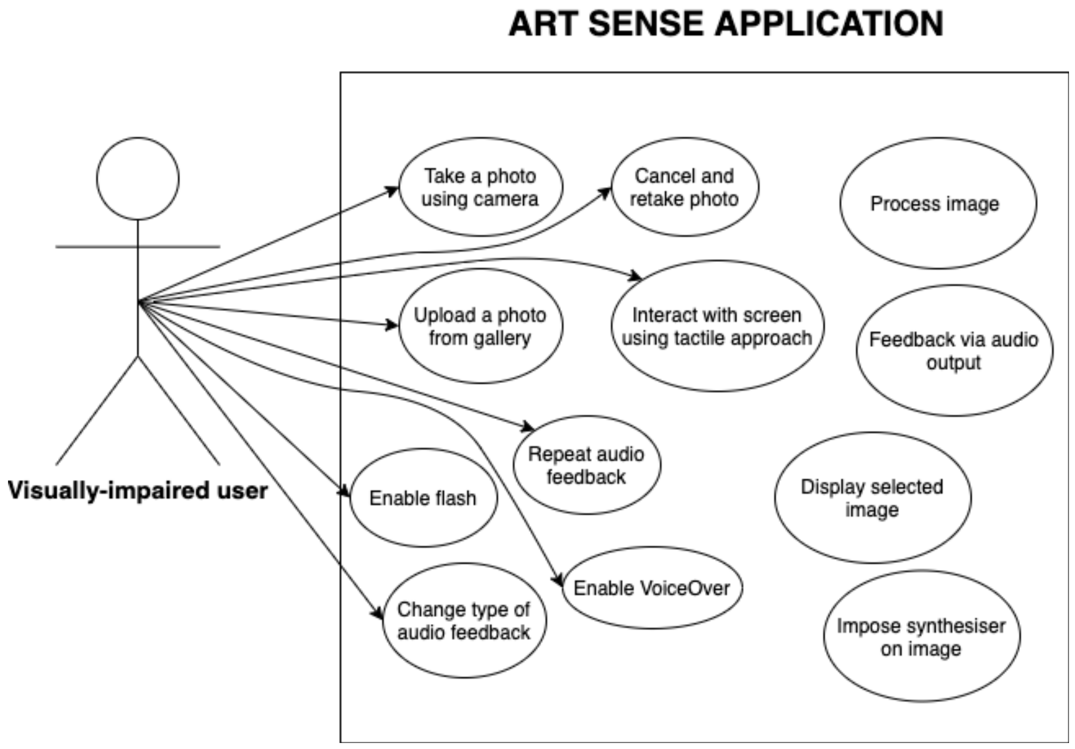

ART SENSE APPLICATION
Visually-impaired user
Take a photo using camera
Cancel and retake photo
Process image
Upload a photo from gallery
Interact with screen using tactile approach
Feedback via audio output
Repeat audio feedback
Enable flash
Display selected image
Enable VoiceOver
Change type of audio feedback
Impose synthesiser on image

Figure 3.1 Use Case describing a visually-impaired user's interaction with ArtSense

The above diagram shows an initial interaction of the visually-impaired user with the application, and everything they should be allowed to do. This includes take a photo, cancel and retake a photo, upload a photo, repeat audio feedback, change the type of audio feedback, enable flash, enable VoiceOver, and interact with the screen using tactile interaction. These follow the requirements highlighted in the previous subsection. ArtSense on its own should be able to display the selected image back to the user, process the image, and give audio feedback via a synthesiser that is imposed on the displayed image.

3.3: State Diagram

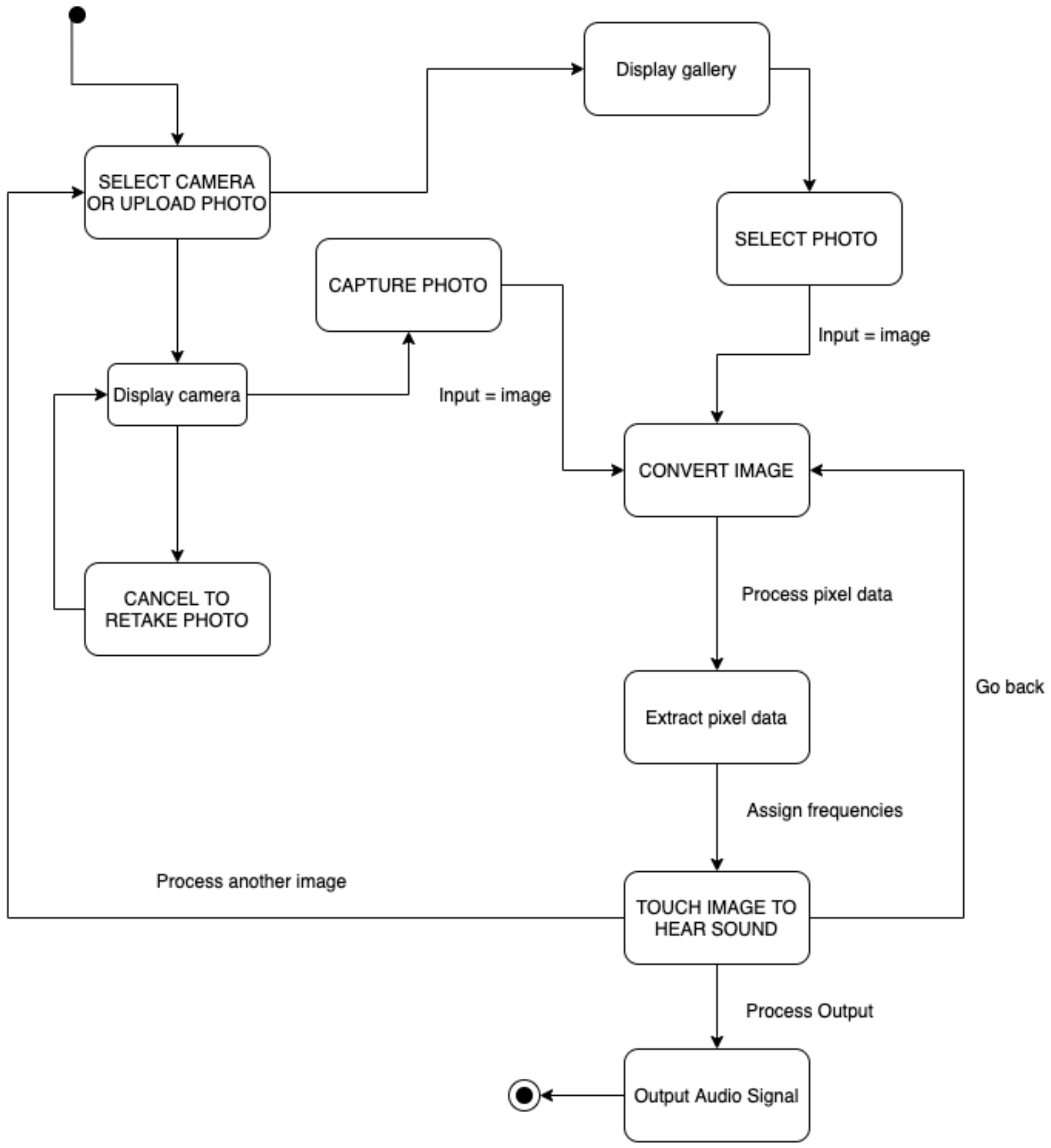


Figure 3.2 State diagram showing how a user navigates through ArtSense

The above diagram shows the different states of ArtSense, and how a user navigates through it. They start off by choosing to either upload a photo from the device's gallery, or take a photo from the camera. If they choose the camera, they have the option to retake the photo and the camera is displayed to them again. After selecting a photo, it is converted, as the pixel data is extracted, and they also have the option to go back to the main page. They can then touch the image to receive audio output, and have the option to go back to the previous page. From there, they can either go back to the home page and start the process again with a new image, or terminate the application.

Chapter 4 | Design

4.1: Initial Mockup

Initially, the application ArtSense aimed to cater to both hearing and visually-impaired users. For the case of hearing-impaired users, they would be able to record an audio and receive an image as output. However, due to time constraints on the project life, ArtSense ultimately became an application only for visually-impaired users who would receive audio feedback. Figure 4.1 shows an initial mockup of how both users would be able to navigate through the application.

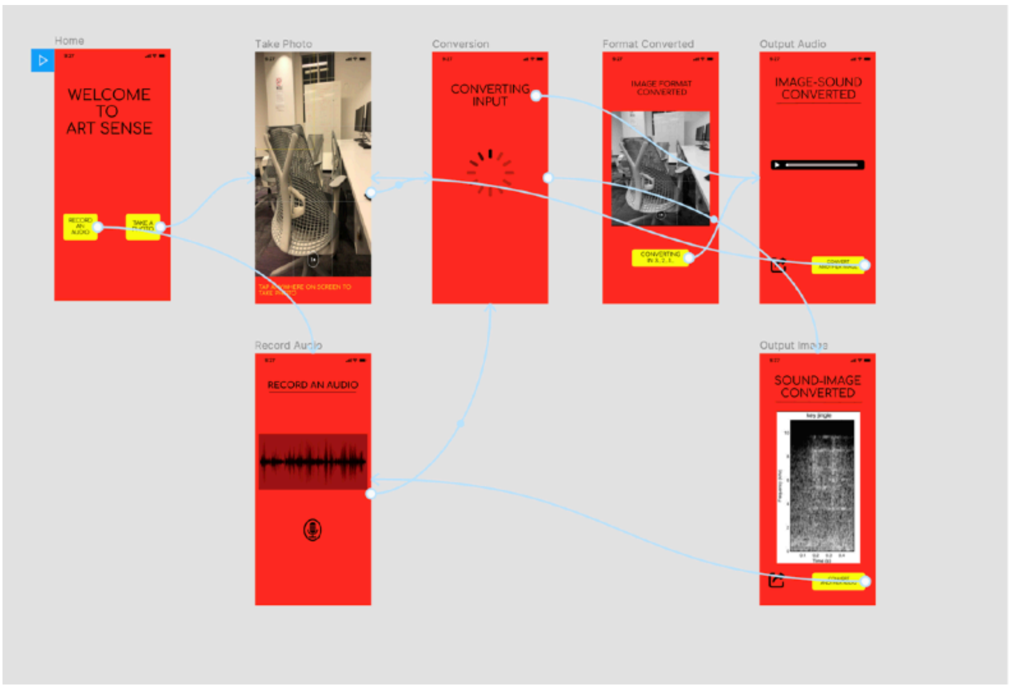


Figure 4.1 Initial mockup of ArtSense

Figure 4.1 also shows that the image is being converted to greyscale, as this was the initial design plan, before processing colour was considered. After the image is converted and processed, an audio track would be played to the user. This was before the idea of creating a synthesiser was considered, and before any tactile interaction was allowed. After a few interviews and further research, it seemed that allowing tactile interaction would have a deeper affect on the users, in allowing them to communicate and experience the artwork. Since ArtSense was only outputting frequency, the simply hearing an audio track of different frequencies would not be very pleasant to listen to. At least when the user is interacting with the image, there is a slight distraction from the current unpleasant sound. However, there is scope for future versions of this project to produce an audio track. The bottom section of Figure 4.1 shows how the user is able to record an audio, and a corresponding image is displayed.

4.2: Further Design Development

As work began on implementation, the first decision made was to eliminate the functionalities that catered to the hearing-impaired users, reasons for which are mentioned in Chapter 4.1. As can be seen in Figure 4.2, the overall application design is not too complex and there is a simple path to navigate. Through research and interviews, it was clear that limiting the amount of interaction with the screen, in terms of navigation, ensures that users do not get overwhelmed.

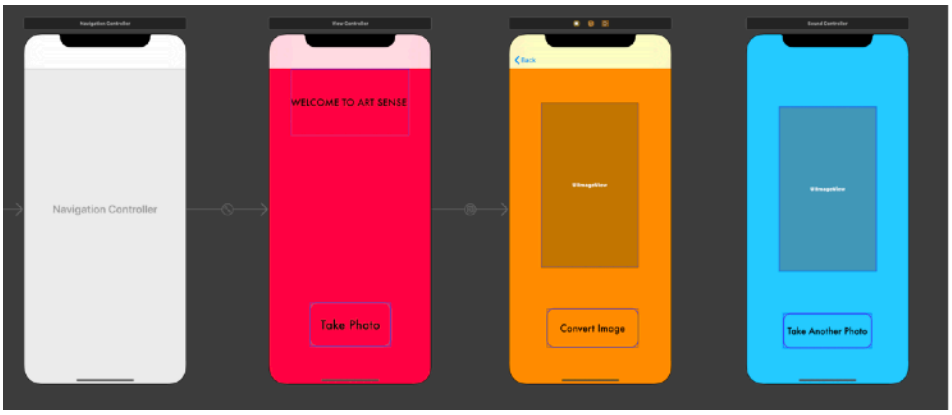


Figure 4.2 Updates to ArtSense after excluding 'Record Audio' functionality

This design shows that when the application starts, there is a simple layout of a title, and a button. The button is placed in the centre at the bottom of the screen so that the user does not have to go through the difficulty of locating the button to proceed in a misguided manner. Ergonomic studies[11] of UI and UX (Figure 4.3[11]) show how different users tend to hold a mobile phone, and which is the most popular method.

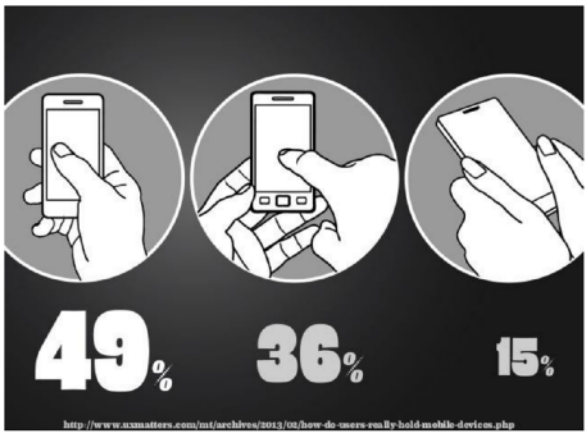

49%
36%
15%
http://www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php

Figure 4.3 How users tend to hold a mobile phone

In most practices, the thumb has easy access to the location of the home button, and since that is the most frequently used feature of all smartphones, it is the most familiar. This is why the buttons in the design for ArtSense are located near the home button (except for the Back buttons - due to Xcode conventions). Even when the camera is displayed (which is a standard iOS camera format), the button to capture an image is also large and lower centrally located, maintaining continuity throughout the application.

Once an image is taken, it is resized and displayed for the user - reasons for resizing are discussed in chapter 5.2 of this report. The user then clicks the 'Convert Image' button and is redirected to a new view controller, where the processed image is displayed again and can be interacted with.

After establishing this basic initial layout, updates were made that show the camera and the final page where tactile interaction takes place. This can be seen in the following Figures 4.4, 4.5 and 4.6:

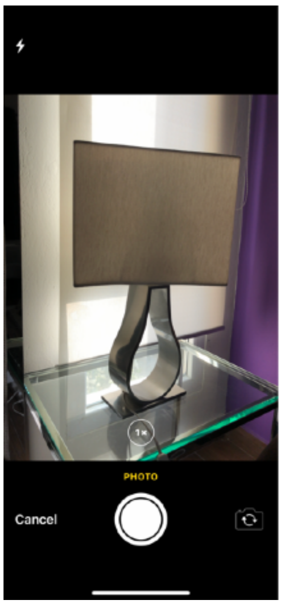


Figure 4.4 The displayed

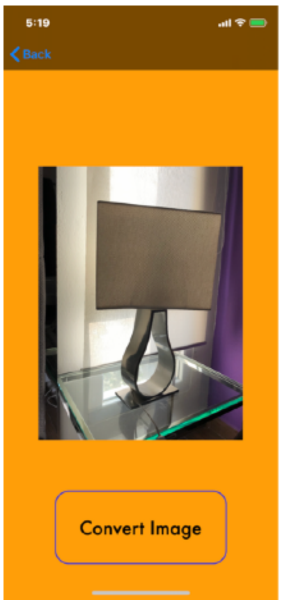


Figure 4.5 Image displayed to user

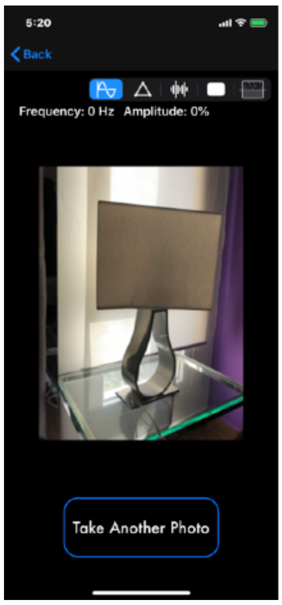


Figure 4.6 Processed image on which tactile interaction is done

Once the functionality of converting images to give audio output was done, another functionality of uploading a photo was added. This gives users the choice to upload a photo from their gallery, which is then displayed to them and converted in the same way. In Figure 4.6, the 'Take Another Photo' button was replaced with a 'Home' UIButton, which would redirect the user to the home page, displaying the options to take a photo or upload a photo.

As mentioned before, iOS has a VoiceOver feature, which when enabled should be compatible with ArtSense. When tested, all buttons and labels were recognised and transmitted to the user as expected, however there was the issue of blocking the output audio when the screen is touched over the Synthesiser (more details in Chapter 5.3). To overcome this, the Accessibility feature in Xcode had to be enabled, which further enables User Interaction. Once this was done, users would hear the required output audio even with VoiceOver enabled.

Figure 4.7 shows the final layout of ArtSense, in which the 'upload image' feature has been added. The 'take another photo' button in the last view controller was replaces with a 'Home' button, which takes the user back to the main page, to allow them to process another image that has either been captured or uploaded.

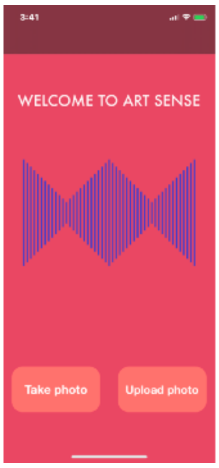


Figure 4.7 Home Page of ArtSense

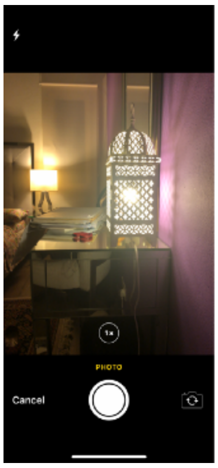


Figure 4.8 Take photo via camera

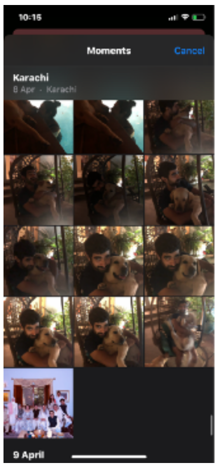


Figure 4.9 Upload photo from gallery

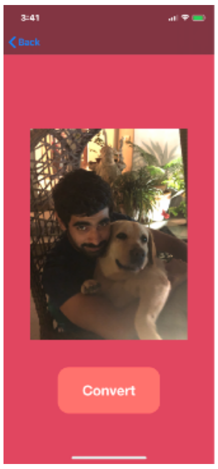


Figure 4.10 Selected image displayed

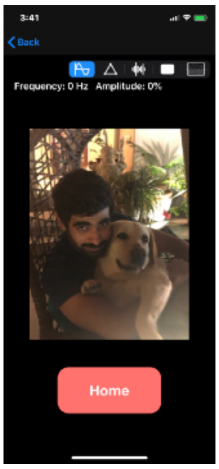


Figure 4.11 Synthesiser imposed on image for audio output

Figure 4.8 shows the screen when the user chooses to take a photo whereas Figure 4.9 shows the screen when the user chooses to upload a photo from their gallery. The selected image is then displayed back to the user in Figure 4.10, and the processed image is displayed to the user on which they can tap to hear output (Figure 4.11).

Chapter 5 | Implementation

5.1: Initial Development

As mentioned before, ArtSense adopted the agile approach of software engineering which is when during the project life, requirements and solutions keep evolving. As can be seen in Chapter 4 of Design, many changes and updates are made as development progresses.

The implementation of functionality also adopted this approach, where the updates that came about mostly had to do with the output audio. This chapter is a step by step run through of how the application was developed.

5.1.1 Programming Languages and Development Environment

Before any work could begin, we had to consider which programming language to use. Since this version of the project was decidedly an iOS application, the environment used for development used is Xcode. Xcode is an environment developed by Apple to assist iOS, macOS and tvOS development.

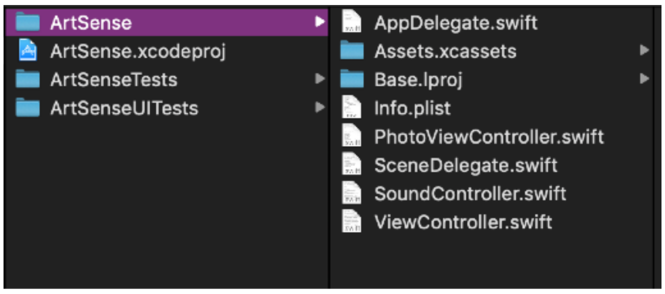
The main languages used for Xcode development are Swift, Objective-C and C++. Out of these, our experience was best with Swift which is why the ultimate decision to develop using Swift was made. Along with familiarity, Swift is a very vast and dynamic programming language which is very commonly used for mobile app design. Along with this, many of the audio frameworks and analysing tools that were being considered for development were Swift compatible.

When it comes to layout design, Xcode helps with visually mapping out as you design through its Storyboard feature, where you can get a good sense of what your product will look like, without having wait for code to compile and run, which made development more efficient. You are able to display multiple view controllers at a time so you can see how your application navigates. Again, this helped with getting a good idea of what has been developed so far, and what is to be done.

With the Storyboard feature, we were able to visually map out phases that would not be worked on till much later. This helped in creating a timeline and further organising Trello boards, which gave a good estimate on the timeline of the project as a whole.

5.1.2 Frameworks and File Structure

The project directory has the following structure:



```
ArtSense          ▸   AppDelegate.swift
ArtSense.xcodeproj    Assets.xcassets        ▸
ArtSenseTests     ▸   Base.lproj             ▸
ArtSenseUITests   ▸   Info.plist
                      PhotoViewController.swift
                      SceneDelegate.swift
                      SoundController.swift
                      ViewController.swift
```

Figure 5.1 Project file structure

Each .swift file corresponds to the viewcontrollers. The folder *Assests.xcassets* contains all the images used in the application, and *Base.lproj* contains and launches the storyboard in Xcode.

The application ArtSense, as shown in the diagrams in Chapter 4.2, consists of three main view controllers, and one navigation controller. A view controller is used to manage the `UIKit` application's interface. It manages a single root view, and may contain several subviews, creating a view hierarchy. Any user interaction with this hierarchy is managed by the view controller, which also has the ability to communicate with other objects of the application as needed. A `UIKit` is a framework that is used most often in iOS development, as it contains the core components that are needed, including buttons, labels, ranging all the way to table views, navigation controllers and so on.

Foundation is another core framework which provides a base layer of functionality. Its functionalities include data storage and persistence, text processing, date and time calculations, sorting and filtering, and networking. The classes and protocols defined by this framework are not only used in iOS development, but also macOS, watchOS and tvOS SDKs[12].

Each view controller has its own class along with any other classes it may need to reference, and other structures needed. Each class is defined in its own .swift file, for organisational purposes. Apart from the view controller files, there is another swift file named AppDelegate.swift. This is an extremely crucial class required for any form of iOS development, as it is where everything begins. A *delegate* is a class that does something on behalf of other classes. `AppDelegate` handles special `UIApplication` states, and consists of several functions called by iOS. In the project bundle there is also an Info.plist which is an information property file. This contains essential configuration information for a bundled executable. In ArtSense, when a user attempts to take a photo, the key `Camera Usage Description` must be added with a value, in order to allow the application to use a camera and capture images. Similarly, if a user was to record an audio (case of hearing-impaired user), the `Microphone Usage Description` should also be enabled, to allow the application to use the in-built microphone.

5.2: Image Processing

The first page the user opens is the main title page of ArtSense. The `UIButtons` on this screen are 'Take Photo' - which when clicked, redirects the user to the next page of the application, where they can take a picture - and 'Upload Photo' - which redirects the user to their gallery, allowing them to select an image to upload. The buttons belong to the `DesignableButton` class, through which they can be formatted according to the design decisions highlighted in Chapter 4.2.

A `PhotoViewController` class is created, which inherits the superclasses `UIViewController, UIImagePickerControllerDelegate` and `UINavigationControllerDelegate`. Depending on whether the image was captured or uploaded, the `takePhoto()` and `uploadPhoto()` methods are triggered. In the case of `takePhoto()`, the function first checks if there is a camera on the device. An instance of `UIImagePickerController()` is created, and the `UIImagePickerControllerDelegate` and `UINavigationControllerDelegate` are set. Once these preconditions are set, the camera is displayed to the user. In the case of `uploadPhoto()`, the function first checks if the source type `.savedPhotosAlbum` is available, and then continues as explained above. The image is then stored in a global variable, and displayed to the user. While taking a picture, the user has the option to retake it as many times.

The next step for the user is to convert the displayed image. When they tap the 'Convert Image' `UIButton`, the functions involved in image processing are triggered. The image is to be processed before being sent to the next view controller.

In a general way, image processing for this application refers to the extraction of RGBA (red, green, blue, alpha) values from each pixel in the image, and assigning the corresponding frequency. The correspondence of colour and sound has been established from Pridmore's[8] colour wheel.

The image first has to be resized. This is because the original image has a very large number of pixels, and having to iterate through that number multiple times makes the program inefficient. It is resized to a new width of 280, and the height is calculated by the following formulas:

$$\text{scale} = 200 \div \text{old width} \qquad \textbf{Equation 5.1}$$

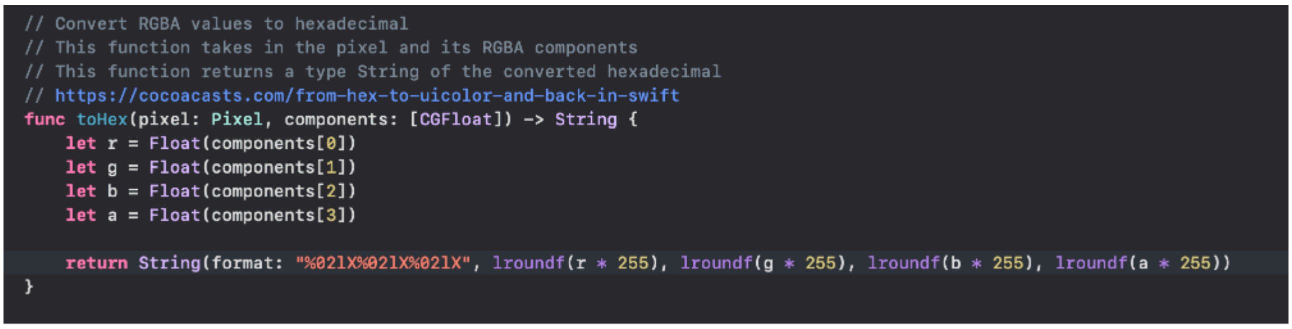
$$\text{new height} = \text{old height} \times \text{scale} \qquad \textbf{Equation 5.2}$$

It should be noted here that since the image is being resized, the quality is being reduced. Therefore, to ensure the program still functions as expected, the original picture must be of good quality, to be able to endure the quality reduction after resizing and still output a comprehensive output.

Because the image pixels are now temporarily stored in memory, they have bytes associated with them. Because of this, processing cannot be done using the image format `UIImage`. Therefore, the `UIImage` needs to be converted to a `CGImage`, which can only represent bitmaps, and this is needed because we require access to the actual bitmap data.

An array of type `[Pixel]` is created to store each pixel's RGBA values. `Pixel` is a separate structure that is defined in order to create efficiency and ease of access to the pixel data. It contains variables `r, g, b,` and `a` to store the red, green, blue, alpha values respectively, and variables `row, col` which store the location of these values in the image. It should be noted that these `row` and `col` coordinates are in reference to the local frame of the image. Because the output is based on where the user touches the screen, coordinates are a very important factor here. As mentioned above, each pixel is assigned a frequency, and the output frequency is based on which pixel is tapped. Because of this, we need create a dictionary, which holds pixel coordinates in reference to the superview or global frame, and the assigned frequency.

For each pixel, once the colour components are extracted, they then need to be converted to a hexadecimal value, which is stored as type `String`. The following figure is an extract from the source code, displaying a function that does this conversion.



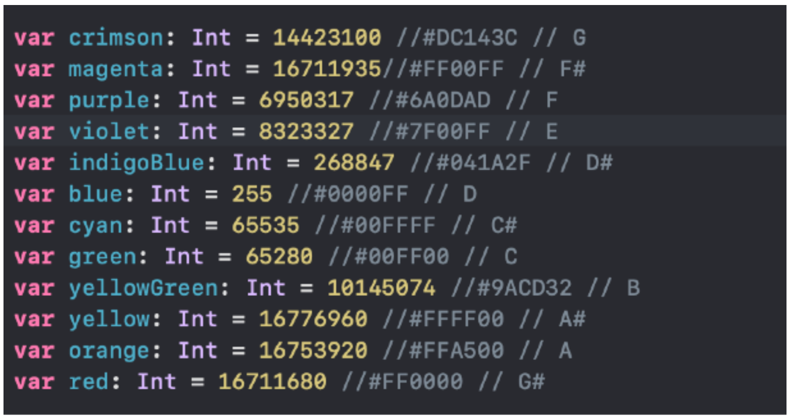
```swift
// Convert RGBA values to hexadecimal
// This function takes in the pixel and its RGBA components
// This function returns a type String of the converted hexadecimal
// https://cocoacasts.com/from-hex-to-uicolor-and-back-in-swift
func toHex(pixel: Pixel, components: [CGFloat]) -> String {
    let r = Float(components[0])
    let g = Float(components[1])
    let b = Float(components[2])
    let a = Float(components[3])

    return String(format: "%02lX%02lX%02lX", lroundf(r * 255), lroundf(g * 255), lroundf(b * 255), lroundf(a * 255))
}
```

Figure 5.1 Convert RGBA to hexadecimal for a pixel

The hexadecimal value is then converted to a decimal in order to be compared to the existing colours from the colour wheel[8].

The `row, col` coordinates are converted to the corresponding coordinates in the global frame. This is done using the function `view.convert(CGPoint:from:UIView)`. Now that we have the global coordinates and the corresponding decimal values of each pixel and its colours, we can compare them to the existing values of the colour wheel. Following the colour wheel[8], a dictionary is created (`pixelSound`) which stores notes of type `String` as they keys, and the corresponding colour as values - in decimal form - in the manner: "G": crimson, "F#": magenta, "F": purple…

Each colour has the following decimal values:

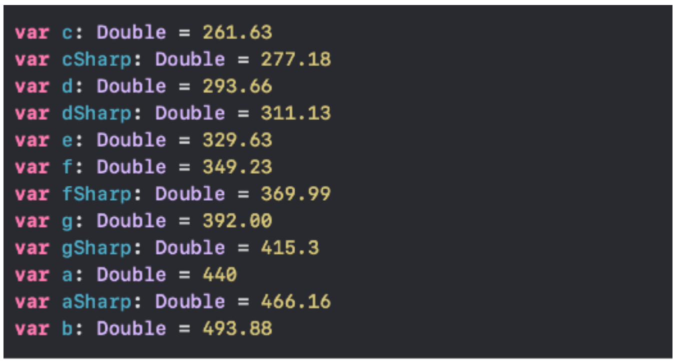


```
var crimson: Int = 14423100 //#DC143C // G
var magenta: Int = 16711935//#FF00FF // F#
var purple: Int = 6950317 //#6A0DAD // F
var violet: Int = 8323327 //#7F00FF // E
var indigoBlue: Int = 268847 //#041A2F // D#
var blue: Int = 255 //#0000FF // D
var cyan: Int = 65535 //#00FFFF // C#
var green: Int = 65280 //#00FF00 // C
var yellowGreen: Int = 10145074 //#9ACD32 // B
var yellow: Int = 16776960 //#FFFF00 // A#
var orange: Int = 16753920 //#FFA500 // A
var red: Int = 16711680 //#FF0000 // G#
```

Figure 5.2 Colours with their corresponding decimal value

Each pixel's colour decimal value is compared to colours stored in this dictionary, and the closest one is set. This is because RGBA values can differ by the slightest decimal point, which is why capturing all the possible colours is near impossible and very inefficient.

The image is almost done being processed, the only thing left is assigning frequencies. Similar to the `pixelSound` dictionary, there is another dictionary created (`tones`), which stores notes of type `String` as keys, and frequencies of type `Double` as values. Again, for each pixel, which now has an assigned note of type `String`, a frequency from the existing `tones` is assigned. As discussed in Chapter 2, the same 12 notes have varying frequencies, depending on which octave they are played in. Keeping in mind the audible range of frequencies, the following values were assigned:



```
var c: Double = 261.63
var cSharp: Double = 277.18
var d: Double = 293.66
var dSharp: Double = 311.13
var e: Double = 329.63
var f: Double = 349.23
var fSharp: Double = 369.99
var g: Double = 392.00
var gSharp: Double = 415.3
var a: Double = 440
var aSharp: Double = 466.16
var b: Double = 493.88
```

Figure 5.3 Frequencies of the 12 notes

Now the image is done being processed, and can be transferred to the next view controller, that is `SoundController`. The two properties that are being transferred are the resized image, and the dictionary `imageCoordFreqs` of type `[Point2D: Double]`. This holds the pixel coordinates (x, y) of structure `Point2D` which is a `hashable` structure, and the frequency.

5.3: Audio Output

The swift file SoundController.swift corresponds to the last viewcontroller, which is where the audio is heard by the user as they tap or drag their finger over the image. Since we are working with audio and visuals, the AVFoundation framework needs to be imported. This framework provides services for working with time-based audiovisual media on Apple operating systems, with both Objective-C and Swift interfaces.

With the information from Chapter 5.2, we now have a processed image. The main functionality of this view controller can be seen in the Synth class that is located in the SoundController.swift file. This class sets up a very basic synthesiser, which uses an updated version of code posted on a public forum[13]. The most important component of this class is the AVAudioEngine (variable name audioEngine). This hosts the AVAudioNodes that make sound, which are added to the signal chain. The main mixer node (mainMixer) constant is a singleton that is connected to the outputNode on initialisation. It acts as an intermediary between the source nodes and the outputNode[13]. A format is created by calling inputFormat for bus 0 on the outputNode. This will provide default audio settings for the device on which ArtSense is running. The deltaTime variable is the duration each sample is held for, and is the inverse of the sampleRate. For example, if the sampleRate is 44100 Hz, it would take one second and divide it into 44100 to represent each of the samples[13].

Before instantiating the AVSourceNode (variable name sourceNode), we must set up a typealias Signal, which is the closure type that takes in a time value of type Float and returns an audio sample of type Float. A variable of type Signal is then added to the Synth class.

In order to complete the block for the sourceNode variable, two more variables are needed, namely frameCount and audioBufferList. This buffer list holds an array of audio buffer structures that will be filled with the custom waveforms, which are discussed ahead in this chapter. Buffers are used generate samples within the render block, and they generally contain between 128 and 1024 samples[13]. We now want to iterate over a range of values from 0 to our frameCount value. Frames are sets of samples that occur at the same time. Because this synthesiser is monophonic (using only one channel of transmission), both sample values are the same. However, if we choose to use stereo audio, the left ear sample and the right ear sample will be different, and require two channels of transmission. While iterating through these values, we generate a sample value by calling the previously instantiated signal closure, and advance by deltaTime. For each element in the UnsafeMutableAudioBufferListPointer, we can index the buffer at the current frame and set it to the sample value that was previously obtained.

Now that the sourceNode variable has been set up, we can continue to initialising the class Synth. We create a variable called inputFormat of type AVAudioFormat, inputting the parameters of a common format, the sample rate (sampleRate), limiting the number of channels to 1 and setting it to being interleaved. After this, we can attach the sourceNode to the audioEngine and connect it to the mainMixer. We then also connect the mainMixer to the outputNode. The initial volume is set to 0 so that there is no sound until the user requests audio output. Finally, we can try starting the `audioEngine`, and print an error if there is a problem. Now that the properties of `Synth` have been established, we need to set a public accessor method to set the signal.

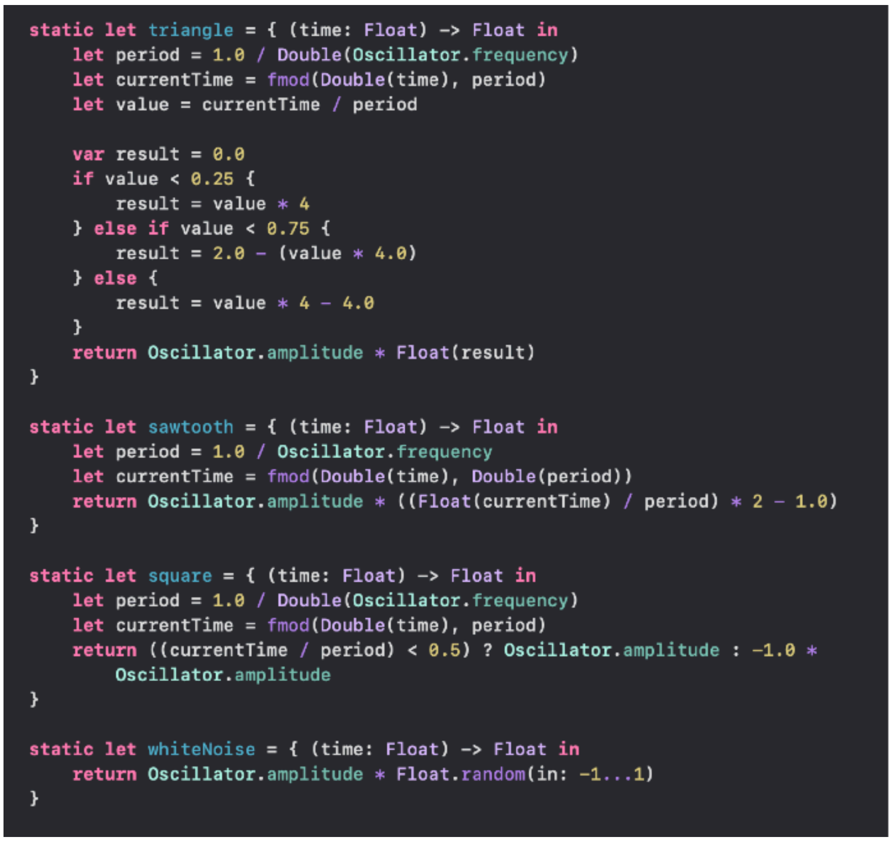
As can be seen in Chapter 4 showing the design of ArtSense, the user has the option of five different oscillators which changes the audio that is heard. This is dealt with in a structure called `Oscillator`. This structure varies the frequency and amplitude to create different sounds that are, by convention, named Sine, Triangle, Sawtooth, Square and White Noise. Calculations for each of these oscillators are done using trigonometry. For example, trigonometrical calculations tell us that sine is a periodic function of time, where the period is equal to

$$2\pi \div b$$ **Equation 5.3**

where b is the factor that time is being multiplied by. In this case, b is equal to

$$2\pi \times \texttt{Oscillator.frequency}$$ **Equation 5.4**

This means that the period of the sine wave is `1 ÷ Oscillator.frequency`. This makes sense because frequency is in Hz or cycles per second. If there are 440 cycles per second (concert A), each cycle is allotted 1/440th of a second[13]. The calculations for the other types of oscillators can be found in Figure 5.4:



```swift
static let triangle = { (time: Float) -> Float in
    let period = 1.0 / Double(Oscillator.frequency)
    let currentTime = fmod(Double(time), period)
    let value = currentTime / period

    var result = 0.0
    if value < 0.25 {
        result = value * 4
    } else if value < 0.75 {
        result = 2.0 - (value * 4.0)
    } else {
        result = value * 4 - 4.0
    }
    return Oscillator.amplitude * Float(result)
}

static let sawtooth = { (time: Float) -> Float in
    let period = 1.0 / Oscillator.frequency
    let currentTime = fmod(Double(time), Double(period))
    return Oscillator.amplitude * ((Float(currentTime) / period) * 2 - 1.0)
}

static let square = { (time: Float) -> Float in
    let period = 1.0 / Double(Oscillator.frequency)
    let currentTime = fmod(Double(time), period)
    return ((currentTime / period) < 0.5) ? Oscillator.amplitude : -1.0 *
        Oscillator.amplitude
}

static let whiteNoise = { (time: Float) -> Float in
    return Oscillator.amplitude * Float.random(in: -1...1)
}
```

Figure 5.4 Calculations for Triangle, Sawtooth, Square and White Noise oscillators

A subview of this view controller consists of a `UISegmentedControl` (variable name `waveformSelectorSegementedControl`). A segmented control is simply a set of multiple segments, which function as mutually exclusive buttons. In ArtSense, this allows the user to select which oscillator (type of sound) they would like to hear.

After selecting a type of waveform, the program must update its output audio accordingly. This triggers the `updateOscillatorWaveform()` function, which simply goes to the `Oscillator` structure and calculates all the values necessary from the chosen waveform.

Now that our different audio feedback options are in place, we can deal with what happens when a user taps on the image - on which the synthesiser is imposed. Firstly, we must switch on the synthesiser which is done via the `setPlayBackStateTo(Bool:)` function, which sets the volume to `0.5` or `0` using a ternary operator. Whenever a point on the screen is tapped, the coordinates of that location are noted, and the function `setSynthParametersFrom(CGPoint:)` is triggered. This function takes in the coordinates of the tapped location as a parameter. The amplitude is calculated by the following formula:

$$\text{amplitude} = (\text{height} - y) \div \text{height} \qquad \textbf{Equation 5.5}$$

where `height` refers to the height of the superview of the view controller, and `y` is the y coordinate of the point on screen that is tapped.

The frequency is according to the frequency assigned to the pixel of the image that is being tapped. The dictionary that was passed on from the previous `PhotoViewController` class contains the coordinates of the image pixels in the global reference frame, and the corresponding frequency. Therefore, to get the current frequency parameter, we iterate over the dictionary and find the matching coordinates. Unfortunately, this is not as straightforward as that. A mobile phone screen contains a very large number of coordinates, and the image that we are processing is not as refined as the screen. Remember, we also resized the image to a smaller number of pixels. Therefore, there is a lot of approximation in matching the coordinates. A range is created of x and y coordinates, that look at the coordinate of type `CGPoint` that is noted upon the user tapping the screen. This range looks one floating point in the negative and positive direction. This is because the coordinates registered by the screen vary by small decimal points. So, when iterating over the dictionary's coordinates, the program checks whether those coordinates fall in the range of the currently tapped coordinate. If it does, return the frequency value of that coordinate from the dictionary. If there is no match (which will happen when the user taps outside the image bounds), return a frequency of `0`Hz, which will be silence.

Once the amplitude has been calculated and the frequency retrieved, this information is displayed to the user via a `UILabel`, and the output audio is heard.

This view controller also contains a `UIButton` 'Home', which redirects the user back to the main page, allowing them to repeat the process by taking or uploading another photo.

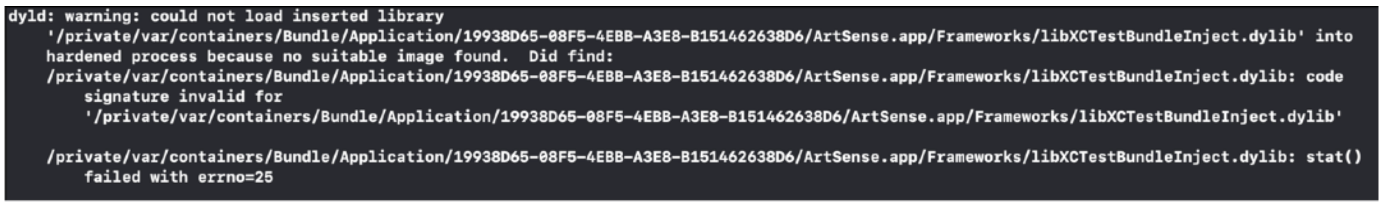
5.4: Bugs and Problems in Implementation

In the early stages of planning this project, the output was initially supposed to be a an audio track the user would hear, as opposed to hearing feedback through tactile interaction. The frequencies and pixel data were gathered as mentioned in Chapter 5.2 and 5.3, and after being processed it would be outputted in the form of a continuous sound with the varying frequencies. To achieve this, we explored the `AudioKit` platform. This is an audio synthesis, processing and analysis platform for iOS, macOS and tvOS. There is also an element of `AudioKitUI`, which incorporates the audio output with the user interface of the application. Different versions of this kit are available for download online[14], and once it is added to the project bundle, can be imported. However, when attempting to import this framework in the ArtSense project, there were error of loading the relevant modules. Due to time constraints, these errors could not be addressed, and therefore the implementation of audio output was changed to the basic synthesiser explained in Chapter 5.3 of this report. As discussed in Chapter 8, if future development requires instruments or a more diverse form of outputting audio, the `AudioKit` framework would be the best way to do so.

Another main issue with the delivered project is that within the top and left bounds of the image, the audio output is working as it should, however on the right bound, even when the user's tap goes out of image bounds, a frequency is still heard. There is also no audio heard in the bottom section of the image. The problem could be that the pixel values have somehow rotated 90^o which is why no audio is heard at the bottom of the image, but is heard outside the right side bounds of the image. Unfortunately, due to time constraints, this problem was not resolved.

5.5 Testing

In the project bundle, there are two folders names *ArtSenseTests* and *ArtSenseUITests.* These contain the test classes run for the main project. However, when trying to run functions from the test classes, the following error was displayed:



```
dyld: warning: could not load inserted library
    '/private/var/containers/Bundle/Application/19938D65-08F5-4EBB-A3E8-B151462638D6/ArtSense.app/Frameworks/libXCTestBundleInject.dylib' into
    hardened process because no suitable image found.  Did find:
    /private/var/containers/Bundle/Application/19938D65-08F5-4EBB-A3E8-B151462638D6/ArtSense.app/Frameworks/libXCTestBundleInject.dylib: code
        signature invalid for
        '/private/var/containers/Bundle/Application/19938D65-08F5-4EBB-A3E8-B151462638D6/ArtSense.app/Frameworks/libXCTestBundleInject.dylib'

    /private/var/containers/Bundle/Application/19938D65-08F5-4EBB-A3E8-B151462638D6/ArtSense.app/Frameworks/libXCTestBundleInject.dylib: stat()
        failed with errno=25
```

Figure 5.5 Error when running test classes

When this error was looked into, there seemed to be some problem regarding the certificates and signatures assigned to the project. However, when adjustments were made, other errors appeared in the code. It was therefore decided that test classes would not be implemented, owing to time constraints and a high risk of the code breaking.

Chapter 6 | Evaluation

6.1 Testing ArtSense on Visually-Impaired Users

ArtSense is a product that builds on basic ideas put forth by pervious computer scientists, and opens new avenues of exploring the correlation of music and colour. Originally, this project was supposed to be evaluated in increments with members of the BlindAid[15] and South London Resource Center for Visually Impaired People[16] communities. The initial layout of the application was based on the research carried out on how visually-impaired users operate smartphones, and then the plan for further development and updates would have been made after conducting interviews. However, due to the global pandemic crisis of 2020 (outbreak of Covid-19), the project did not reach the interview phase. To overcome this obstacle, informal telephone interviews were conducted with a select number of visually-impaired people through social contacts. The final version of ArtSense was therefore tested on family members also in quarantine with the developer at the time. This is why, during the evaluation process, VoiceOver was not enabled, and the session offered only preliminary understanding and critique.

6.1.1 Testing Sessions and Feedback

Before conducting an evaluation session, the participants were provided with a brief overview of the aims and objectives of ArtSense. The tester was then allowed to navigate through the application on their own, exploring every aspect at their own pace. During this process, a conscious effort was made to provide minimal prompts and assistance, so that the user's feedback would include any issues they experienced with the user interface.

Test 1:

The briefing for the first tester did not mention the purpose of the project, instead focused more on the different functionalities it offered, for example, the option to either take or upload a photo, after which the converted image can be interacted with to receive audio feedback. This led to the first question by the tester being 'what can this be used for?'. It was therefore noted that for future testers, the purpose of the project should be included in the briefing before the functionalities are explained.

A technical issue that was experienced by this tester was the delay in loading the image when they selected the photo from the gallery. This delay was about 2-3 seconds long. However, while looking at the terminal as the project was running, the print statements confirmed that the program had selected an image. Therefore the delay was in dismissing the gallery and showing the displayed image. This problem was noted, and kept in mind during the other testing sessions, however since it was not experienced with the other three testers, it was not an error that required us to revisit implementation.

Since the tester was a sighted and able-bodied user, feedback on the suitability of the interface for a visually-impaired user was insufficient. An attempt was made to obtain more information about this suitability through a series of pointed questions, for example:

1. Were the button labels clear and easy to understand?
2. Did the buttons clearly outline the next steps throughout the application?
3. How intuitive was the tactile interaction with the processed image?

Through these short questions, it was noted that the layout and navigation was easy to follow as the buttons were self-explanatory. However, this tester did not take their time to explore the converted image and try the different controls of changing the oscillation to vary the output sound, making this testing session insufficient.

Test 2:

A more detailed explanation of the purpose and uses of the project were added to the briefing for the second testing session, and it was emphasised to the user that they take their time to explore the different controls offered on the last page. It was noted that there was no delay on loading the image after being selected from the gallery. The tester pointed out that there is a slight dead zone at the bottom of the image in which there is no audio output, after which a short explanation was given regarding the implementation bugs. After acknowledging the comprehensive layout and easy navigation, the second tester focused most of their feedback on the output audio being heard and how it would be better if there were options of different instruments rather than a single, unmelodious tone being played back. They also interestingly identified that this application would work much better with abstract paintings, rather than portraits. This was an extremely relevant observation, as abstract paintings have more room for interpretation as compared to subject matter as well-defined as a portrait.

Test 3:

The third tester first ran the application by taking a photo, took their time to explore the converted image, and then re-ran the application by uploading a photo. After focusing on different points of the image, they noticed that some frequencies were not being outputted as expected. The reason for this was that the original image they had uploaded was of low quality, with a blurred background. This was explained to the tester, and when they uploaded a higher quality image, there was a marked improvement in the accuracy of the frequencies.

Test 4:

The fourth tester gave more in-depth feedback as compared to the others, as they explained every observation while navigating through the application. Comments on the design included an appreciation of the colour palette of the application, the logo and the visibility of the status bar of the iPhone. However, in the case of a visually-impaired user, this would not be as appreciated or relevant. It was appreciated that the camera format was familiar, and that there is audio feedback from the device to confirm that the image has been captured. When interacting with the converted image, they observed that the lighter colours had higher frequencies, and the darker tones had lower frequencies - as expected. The issue of the dead zone was noted, as in Test 2. As before, the possibility of choosing different instruments rather than oscillations was expressed. Tester 4 also asked what would happen if they were to use a more precise tool to interact with the screen (for example, a pencil rather

than a fingertip). In response to this, the explanation of coordinate approximations was offered, which is discussed in the following section of this chapter.
All four testers mentioned that they would like to hear a more pleasant sound, rather than the basic tone of the audio frequencies. A way to improve this in ArtSense would be to involve instruments when delivering output. Instead of choosing between the five different oscillators (details in Chapter 5), the options of five different instruments could be given, for example a piano, violin, guitar or harp. Further details are discussed in Chapter 8 of this report. To obtain a more continuous and melodious sounding output, the levels of gain and reverb of the instrument could also be adjusted.

6.2 Evaluation

Overall, the application achieves a significant percentage of its aims and objectives as set out in Chapter 1 and 2. In retrospect, although the aims were achieved by using Swift as the main programming language, C++ may have been made the program even more efficient, because of the different functions that are not available in Swift, which resulted in doing more work in achieving the same output. Computational complexity is a major factor in the ArtSense program, as there are multiple references to dictionaries that are defined and stored in memory, and therefore the number of times memory is accessed hinders the performance time. Even though the adding and extracting of frequencies to dictionaries is done in O(n) linear time (as explained in Chapter 6.2.1), if a more efficient way of doing this is found, the selected image may not have to be resized to such a small number of pixels, therefore refining the quality and allowing a greater range of outputted frequency.

Strengths:
1. Addresses the gap in the market based on an evaluation of existing applications by extracting data from coloured images, allowing the user agency through tactile interaction with the image and optimising functionality for visually-impaired users
2. The user interface and user experience are intuitive and extremely easy to navigate and assimilate well with the VoiceOver feature.
3. Is readily available for any device with a camera, and requires no external hardware to run, making it accessible to a wider segment of the target audience.

Weaknesses:
1. While the application retains the emotional experience of an artwork or image, the diversity of the audio output in terms of sounds and instruments can be significantly improved.
2. The implemented software could not be adequately tested, which may cause delays in further development

6.2.1 Performance

As discussed in Chapter 5, before any pixel data can be extracted, the image has to be resized. This is to reduce the number of pixels that need to be iterated through multiple times throughout the implementation. Before resizing was introduced, the application was run using the standard number of pixels, and the time taken to execute extraction of pixel data, and assigning frequencies was recorded. To extract pixel data, it took approximately 127.8540 seconds to load. This is clearly too long, and leaves more chances for there to be a memory leak when assigning frequencies. Therefore, the image was resized to reduce the number of pixels. The following table shows the differences in time to run the two functions, for when a photo is captured:

Functions (seconds)	**New Width (pixels)**			
	280	**310**	**350**	**400**
Extract pixel data	1.0447	1.2881	1.6413	2.1244
Assign frequencies	0.1017	0.1102	0.1279	0.1823

Figure 6.1: Performance of extracting and assigning frequencies to images captured, of different sizes

The figure below shows the differences in performance for an image that has been uploaded, of different sizes:

Functions (seconds)	**New Width (pixels)**			
	280	310	350	400
Extract pixel data	1.0671	1.3412	1.5153	2.0279
Assign frequencies	0.1054	0.1125	0.1288	0.1704

Figure 6.2 Performance of extracting and assigning frequencies to uploaded images of different sizes

These values were calculated using the same image, but of varying size. As it can be seen in figures 6.1 and 6.2, the larger the image width (and therefore calculated height), the more time the program takes to both extract pixel data and assign frequencies. This is why, the final model of ArtSense takes the new width of the resized image to be 280, and calculates the corresponding height.

When a key-value pair is updated into a dictionary, it does so in O(1) time which is a linear function. Iterating over a dictionary is done in O(*n*) time (as done in `setClosestFreqs()` function), where *n* is the size of the dictionary. Therefore, the total time taken to set and extract pixel information is O(*n*) time, making it a linear function. Although this shows efficiency, there is scope for further improvement through the exploration of implementing structures other than dictionaries.

6.2.2 Functionality and Ease of Use

The functionality of ArtSense can be divided into two main phases: image processing, and audio output. Image processing is when data regarding colour and position is extracted from each pixel and compared to existing colour values, and assigned a corresponding frequency. As has been mentioned before, there is a lot of approximation in comparing extracted colour values to existing ones. Once the extracted colour value is converted to a decimal, it is compared to a range of decimal values of existing colours, and the closest value is assigned. This, however, means that certain hues are overlooked and therefore there is the possibility that the assigned frequency is either over or underestimated.

The same can be said about the process of outputting audio. When a musical note is assigned to each pixel based on its colour value, it is again compared with an existing set of frequencies, based on its coordinates in the global reference frame. When a user taps on the screen, the coordinates are noted to approximately six decimal places. However, the coordinates of the pixel that have been converted to the global reference frame are rounded to one decimal place by default. When the program iterates through the dictionary of pixel coordinates, it will overlook corresponding whole values due to the precision of the tapped coordinates. To avoid this, a range is created which looks in one position in the positive and negative directions, and if any pixel coordinate lies within this range, the corresponding frequency will be outputted. This creates a few mismatched frequencies, and may also be the reason why, in some cases, output is heard outside image bounds.

Apart from these approximation issues, a majority of the outputted frequencies are consistent with Pridmore's[8] colour wheel assignment, and are conveyed to the user in a uniform manner.

In terms of ease of use, as was noted in the feedback sessions, the application is easy to navigate through, as the buttons are well labelled. Even if the user has no prior knowledge on what this application's functionalities are, they can still find their way. The only issue arises in the last viewcontroller, where if not told beforehand, the user may not be aware that tactile interaction is enabled. In order to address this, future versions of ArtSense could potentially have a pop up on the screen, prompting the user to touch the screen (this was not implemented in ArtSense due to time constraints). However, if this were to be implemented, it should be made sure that VoiceOver recognises the pop up and is able to convey the message to the user.

Chapter 7 | Special Issues

7.1 Legal Issues

Given that the core function of ArtSense is to capture and process images of artworks and convert them into sound, there are certain legal constraints that may affect the use of the application in this context.

The first issue is that of copyright infringement, pertaining to photographing artwork on public display. According to research conducted by Jennifer Saranow Schultz[17], the lawyers she consulted claim that 'snapping such a shot is generally O.K. in the United States if the work of art is in the public domain. For art, this generally means anything created before around 1923[17]'. Chris Sprigman, an intellectual property law professor at the University of Virginia School of Law, stated that 'if the painting is in the public domain, you can take a picture of it, you can reproduce it[17]'. However, the issue arises if the artwork is more recent, as the artist generally holds exclusive rights to any reproduction. Even if this reproduction is made for personal use, it is still a violation of the law. In European countries, Sprigman[17] notes, artists often retain these rights even after the copyrights have expired.

It should also be noted that some galleries and museums have their own policies, which do not allow any photography of the artwork. While ArtSense allows users to take a photo that is converted, it does not save or store the image on the user's device. Therefore, technically, the user is not reproducing the image, they are merely converting it to another medium and interacting with it in-situ. This technicality is still quite vague, and further discussion about the legal implications of its use in such contexts is merited. Implications of uploading an image of artwork acquired from a third party source to ArtSense also need to be explored further.

7.2 Ethical Issues

As mentioned in the previous chapter, during the evaluation phase, ArtSense was unable to be tested with visually-impaired users, its prime target audience. This raises an ethical concern regarding the authenticity of the application within the visually-impaired community. Despite the conscious reiteration in every stage of development that the intended user was visually-impaired, the application cannot be deployed until it has been sufficiently tested on and received feedback from its target audience. There is a chance that minute details or specifications were overlooked by the developer, which can only be identified by a visually-impaired user, as Emsley acknowledged that 'the major challenge in the project was understanding how the world is perceived with different senses[10]'. Given the risk of false assumptions that may have been made during the development of the application, any further development of ArtSense after the circumstances of Covid-19 should be made in strict consultation with the visually-impaired community.

An additional ethical concern arising from the unique circumstances in which this application was tested is the biased nature of the users, who were family members in quarantine with the developer. This may have affected the evaluation of the application.

Chapter 8 | Conclusion

8.1: Conclusion

Ultimately, the final version of ArtSense successfully achieves its aims and objectives as set out in Chapter 1 of this report. These objectives are evaluated as follows:

- ☑ **To create an artistic experience for visually-impaired users by converting full-colour images to audio frequencies:** This was achieved by creating a synthesiser that was imposed on the image in the mobile application.

- ☑ **To understand the theoretical foundation that underpins the scientific link between colour, tone, hue and audio frequencies:** This was achieved through in-depth research into theories outlined by Sir Isaac Newton and Ralph W. Pridmore, whose colour wheel forms the core of the implementation of the application.

- ☐ **To ensure the interface of the mobile application is designed to meet the special needs of visually-impaired users:** While every effort was made to intuit and adapt the user interface to meet the needs of the target audience, the unfortunate circumstances of Covid-19 prevented the application from being adequately tested to ensure this objective was successful.

- ☑ **To allow for further development of the mobile application, for use in more complex contexts or for more sophisticated functionality:** In the research phase of this project, alternative models of calculating frequencies were identified, which may be implemented in future versions of this application.

There are many existing technologies to help visually-impaired users in the field of medicine, education, economics and art. ArtSense has been added to this collection, in order to broaden the opportunities of authentic experience for these users in the realm of art. This product allows users to consider new ways of artistic interpretation, which can also be applied to sighted and able-bodied users, therefore it is not restricted to a specific type of user. Able-bodied users also have a lot to gain from this product, in terms of broadening their views on how to experience and consider different aspects of a work of art.

Through working on this project, a more in depth correlation between image pixels and sound waves was studied, which allowed us to introduce the previously-unexplored variable of colour. Studying previous methods of relating colour to sound waves taught us that there is still so much more that has not been considered in this relation, leaving room for more research into the nuances of colour theory, and vastness of music theory.

8.2 Future Work

Although ArtSense is an iOS application, there is room for it to be translated into an Android application as well, or even for creating a new, purpose-built device solely for image-sound and sound-image conversion, for those users who may not be able to afford a smartphone which comes with a host of additional features.

This opens up avenues into researching finer details, not only programatically, but also theoretically. The colour wheel proposed by Pridmore[8] builds upon Newton's original theory (Figure 1.3), but was developed in 1992. With so many technological advances having been made since, this colour wheel has the potential of creating an even finer correlation between music and colour.

By studying music theory more closely, the audio output for ArtSense can be made more melodic as well. This can be done by playing the frequencies in a certain scale rather than erratic notes, and varying octaves. Indian classical music has nuanced 'half notes', which would make the audio output even more full and rich. The option of different instruments given to the user would allow the output to range from a single instrument to an entire orchestra. As explained in Chapter 2, Newton's colour wheel in 1704 worked with a scale which was a major sixth. By studying other scales, and including a range of octaves, the output sound can be made more diverse, as ArtSense currently operates in a single octave.

An initial idea that could not make it to the final development, was that of catering to the hearing-impaired users by reversing the conversion process. Audio can be recorded and the output can be in terms of visual feedback, using the same basic physical and mathematical concepts discussed in this report. For example, a hearing-impaired user is attending an orchestra concert, and as they record the audio, different colours of varying tones and shades can be seen. A composer also makes as many choices as an artist, with the choice scale, octave, instrument and so on. These choices aim to evoke a certain feeling in the audience, so even if the music is loud and full, it may have an undertone of sadness and despair. By translating frequencies of music to colour, a hearing-impaired user would be able to experience that underlying tone of the music by, for example, seeing darker and duller colours.

These are the many possibilities that populate the future of ArtSense - making art accessible for everyone.

References

[1] Greene, T. 'A beginner's guide to AI: Computer vision and image recognition' *The Next Web, Source: https://thenextweb.com/artificial-intelligence/2018/07/18/a-beginners-guide-to-ai-computer-vision-and-image-recognition/*, 2020.

[2] 'TapTapSee Mobile Application' *Source*: *https://taptapseeapp.com/*, 2020.

[3] 'Aipoly Vision Mobile Application' *Source: https://play.google.com/store/apps/details?id=com.aipoly.vision&hl=en*, 2020.

[4] Jack, O. 'PixelSynth' *Source: https://ojack.github.io/PIXELSYNTH/*, 2020.

[5] Cazan, A. et. al. 'Algorithms and Techniques for Image to Sound Conversion for Helping the Visually Impaired People - Application Proposal', 2007. Maribor. *2007 14th International Workshop on Systems, Signals and Image Processing and 6th EURASIP Conference focused on Speech and Image Processing, Multimedia Communications and Services*. pp. 471-474.

[6] Lundgard, A. et. al. 'Socioethical Considerations for Accessible Visualization Design', October 2019, *2019 IEEE Visualisation Conference*. pp. 16-20.

[7] Zhang. X. et. al. 'An Efficient Method of Image-Sound Conversion Based on IFFT for Vision Aid for the Blind'. January 2014. *Lecture Notes on Software Engineering.* pp. 54-57

[8] Pridmore, W. R. 'Music and Color: Relations in the Psychophysical Perspective', February 1992. Color Research & Application / Volume 17, Issue 1. *John Wiley & Sons Inc.* pp. 57-61.

[9] Sebba. R. 'Structural Correspondence between Music and Color', April 1991. Color Research & Application / Volume 16, Issue 2. pp. 81-88

[10] Emsley, I. 'Please Touch the Art: Experiences in Developing for the Visually Impaired', March 2019. *Journal of Open Research Software.*

[11] Hoober, S. 'How Do Users Really Hold Mobile Devices?' *UX Matters, Source: https://www.uxmatters.com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices.php*, 2020.

[12] 'Apple Development, Foundation Framework', *Source: https://developer.apple.com/documentation/foundation*, 2020.

[13] SwiftMoji. 'Building a Synthesizer in Swift', *Medium: Better Programming, Source: https://medium.com/better-programming/building-a-synthesizer-in-swift-866cd15b731*. 2020.

[14] 'AudioKit Developer Documentation', *Source: https://audiokit.io/*. 2020.

[15] 'BlindAid', *Source: https://www.blindaid.org.uk/*. 2020.

[16] 'South London Resource Center for Visually Impaired People', *Source: http://www.southlondonvision.org/*. 2020.

[17] Schultz, S. J. 'When It's Illegal to Photograph Artwork', *The New York Times, Source: https://bucks.blogs.nytimes.com/2010/09/21/when-its-illegal-to-photograph-artwork/*, 2020.

[18] Pesic, P. 'Music and the Making of Modern Science', 2014. *MIT Press.*

[19] Taylor, P. A. 'Newton's Color Theory, ca. 1665', *The Scientist: Exploring Life, Inspiring Innovation, Source: https://www.the-scientist.com/foundations/newtons-color-theory-ca-1665-31931*. 2020.