Chromaesthesia Mapping Tool for Consistency Testing

A thesis submitted to the National University of Ireland, Cork
in partial fulfilment of the requirements for the degree of
Master of Science in Interactive Media
in the Faculty of Science

October 2021

117424056
School of Computer Science and Information Technology

Contents

List of Figures

List of Algorithms

Abstract

In this thesis we present a novel approach for mapping chromaesthesia, or sound-colour synaesthesia for consistency testing. This is achieved by creating a tool using React and p5.js where subjects are presented with visual manifestations of chromaesthesia, or form-constants, and a synthesiser keyboard with a colour picker on each note. Exporting options allow researchers capture the input of synaesthetes as an image or spreadsheet, enabling them to conduct a consistency test (test of genuineness). The tool provides a means for those who experience chromaesthesia with a way of communicating the types of colours they experience in their everyday lives.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Signed:
117424056

Acknowledgements

I would like to thank my supervisor John O'Mullane for providing guidance and feedback throughout this project. Thank you Mum and Dad for your unconditional support. Thanks also to the Western Roaders and my friends.

Chapter 1

Introduction

Synaesthesia can be most figuratively described as a love story between the senses. As Cytowic put it, "about one in thirty people walk around with a mutation for an inwardly pleasant but apparently useless trait." (Cytowic, 2018). The history of synaesthesia throughout the ages traces variants of the same lexical term being used by ancient Greek and Latin scholars, but not in the context of the phenomenon of synaesthesia as we know it today. To trace this history, an exploration of the lexical term and how it developed over time is required. Information ascertaining forms of synaesthesia and the methods of research used explore the field to this extent reveals a disparity in the research conducted, such as is the case with grapheme-colour and sound-colour synaesthesia. Through exploring this disparity in further detail, the following research question is posed: how to conduct research for sound-colour synaesthesia in the way most grapheme-colour tests do, such to map chromaesthetic associations for consistency testing purposes. By undertaking this research, the relationship between the inputs and the outputs in people who experience chromaesthesia can be explored.

The research question builds upon existing research by providing a method of replicating chromaesthetic mappings as it has been depicted in the past. The objective of conducting this project is to facilitate researchers in conducting consistency tests. Consistency tests, also coined tests of genuineness, prompts subjects to map synesthetic colour associations as they perceive them. Participants chose the colour that best matches their synaesthetic response for the grapheme from a colour palette while control participants are instructed to imitate synaesthesia, this reveals that controls match colours in a much less consistent manner when compared to those with synaesthesia.

To undertake this research question, several steps must be taken. Following a discussion how the field has evolved over time and on and the level of depth to which chromaesthesia and associated visual manifestations have been researched to date, a field review acts as an analysis of the available tools that claim to address this research question. Subsequently, a literature review on types of synaesthetic artefacts and their associated variables are explored along with a discussion on generative-art and creative coding to best approach the research question. Analysing the problem in more depth provides clarity on how best to frame the outcome of the research question while the design section offers a few solutions to best solve the

research question. Implementation of the proposed solution involves a process of describing the architecture of the system as well as the technology used to map chromaesthetic experiences as synaesthetes see them. Finally, evaluating this implementation and its associated strengths and shortcomings gives room for further discussion on future work that can be conducted using this technology.

Chapter 2

Analysis

2.1 Defining Synaesthesia

Synaesthetic occurrences can be described as usually automatic and involuntary, consistent over time, present from early childhood, and genetic, and typically affects about 4% of the population. The most frequent basic 'units' of synaesthesia are grapheme, colours, time units, musical sounds, general sounds, as will be discussed in further detail. Tracing the history of synaesthesia can prove difficult, as the name as it exists today only formally came into existence in 1895. To begin, it is important to define the criteria by which to frame what can be described as synaesthesia and what cannot. For this purpose, Cytowic's five diagnostic criteria will be described.

In the late 1980s, Cytowic described five diagnostic criteria that encapsulate a profile of an idiopathic synaesthete, regardless of the type of the synaesthesia. He stated that synaesthesia is involuntary though elicited; synaesthesia is projected, mostly perceived close to the face; synaesthesia percepts are durable and discrete; synaesthesia is highly memorable; and synaesthesia is emotional (Cytowic, 1989). Using this diagnostic lens, we can proceed to discuss the history of synaesthesia as it progressed throughout different fields as well as different forms of synaesthesia.

2.1.1 1700s

As outlined in *'The evolution of the concept of synaesthesia in the nineteenth century as revealed through the history'*, German poet and philosopher Johann Gottfried Herder in 1172 poem *Treatise on The Origin of Language* can said to be one of the earliest references to the phenomenon of synaesthesia on record. He states: "I am familiar with more than one example in which people, perhaps due to an impression from childhood, by nature could not but through a sudden onset immediately associate with this sound that colour, with this phenomenon that quite different, obscure feeling, which in the light of leisurely reason's comparison has no relation with it at all—for who can compare sound and colour, phenomenon and feeling?" (Herder and Irmscher, 1966). As stated previously and according to Cytowic's diagnostic criteria of

synaesthesia, Herder's formulations "could not but" and "immediately" indicate both involuntary and automatic elicitation of "this sound that colour". Herder's writings do not give an explanation for this occurrence as he coined it an "obscure feeling" but meets the criteria of what Cytowic would characterize as synaesthesia.

While Herder can be cited as the first mention of the phenomenon of synaesthesia in loose terms, the term itself was first used in a modern context in 1892 in the United Kingdom by Frederick W. H. Myers, according to Jewanski (Jewanski et al., 2020). Myers conceived of the term in his article Subliminal Consciousness, calling it *'synæsthesiæ'*. The question of 'who said it first' is still an ongoing discussion among the scientific community and can be said to be a case of having yet to discover an earlier reference to the term than what has already been outlined.

2.1.2 1800s

Famous French physiologist Alfred Vulpian inserted the term *synesthésie* in three lectures over the course of two years, but his definition of the word differed greatly to our modern understanding of synaesthesia. His use of the phrase was to give a name to the occurrence of coughing and sneezing being linked to touch or light: "mechanical irritation of the external auditory canal gives rise to a special sensation, a tickling in the throat, which makes people cough. The impression on the eyes of a bright light, sunlight for example, causes a particular tickle ... indirectly provokes a fit of sneezing in certain susceptible people." (Vulpian, 1866). This demonstrates the lack of a scientific foundation experienced by synaesthetic research at the time-a reflection on the scientific community as it existed in the 1800's than a deliberation on the term itself.

Following on from this, Puerto Rican ophthalmologist Ferdinand Suárez de Mendoza described synaesthesia using the term *fausse sensations secondaire*, translating to 'false secondary sensations' (de Mendoza, 1899). This indicates the nature of the cross modality of synaesthesia, which would act as a core understanding of synaesthesia in years to come. From conducting research on synaesthesia in the United States, William O. Krohn narrowed down the phenomenon to a stimulus-colour synaesthesia or *"pseudo-colour sensation"*. On this, he wrote that, in the majority of cases, "pseudo chromaesthetic phenomena arise from some sort of cerebral work which is the outcome of the cortical centres, which are connected by numerous associational fibers, notably the visual and auditory centres" (Krohn, 1892). He comments on the neurological nature of synaesthesia, integrating several ideas of previously conceived synaesthesia concepts.

2.1.3 1900s

Interest in the topic was accelerated by Sir Francis Galton's writing's on "visualized numbers" in the journal *Nature*. (Galton, 1881). Following the standardisation of the term *'synaesthesia'*, the scientific community still didn't have an answer as to the exact connections of brain areas associated with it, nor was there any apparent link between cross-modal correspondences, or mental images. During the 20th century, however, this began to change as a multitude of articles

surrounding the psychological, neurological, and philosophical nature of the phenomenon arose. This closely followed the trends in psychology at the time with the introduction of behaviourism. Articles that deal with a form of synaesthesia focused on coloured hearing to this day often use the term *audition colorée*, according to Jewanski. (Jewanski et al., 2020).

During the 19th century, the field of psychology lacked the maturity to accurately describe the phenomenon of synaesthesia. They "knew little about how fetal brains develop, the powerful role of synaptic pruning, or how interactions between genetics and environment uniquely sculpt each brain" (Cytowic, 2018). These concepts were only conceived of years later and allowed researchers to review synaesthesia periodically once new scientific findings were uncovered. New findings resulted in a paradigm shift in our understanding of how brains are organised and revealed that on an individual basis, not everyone sees the world the same way as the next person: "synaesthesia highlights how each brain filters the world in its own uniquely subjective way" (Cytowic, 2018). This marks the beginning of investigating synaesthesia as its own field. Researchers have been researching synaesthetic children in great detail since 1980. Historically, synaesthesia has been said to be shaped at a young age through association by fridge magnets where the letter seven may be the colour yellow. It was believed that children may have been conditioned by these objects in their environment.

The history of synaesthesia has undergone many iterations, from a purely art focused expression of an unknown phenomenon, to neurobiological studies on the cross-modal correspondences of a widely accepted term. As has been discussed, the development of the term *'synaesthesia'* is not linear, but instead developed in accordance with external, sometimes non pertinent, factors like developments in surrounding topics. From the 1700's to the 1980's, the phenomenon was not investigated in detail, but as a general term that was speculated upon. This was due to the topic's lack of foundational, outward, and convincing evidence. Multiple disciplines had their own unique way of describing synaesthesia, but none claimed ownership of it. This has led to many different methods of researching the plethora of types of synaesthesia from a variety of angles in their research methods.

2.2 Methods of Research and Types of Synaesthesia

Empirical confirmation of synaesthesia meant that synesthete's experiences could be validated behaviourally if they were consistent over time. This was a crucial factor in the elevation of synaesthesia as a tractable field (Baron-Cohen et al., 1987). During the early 2000's, synaesthesia could no longer be disregarded. This was due to developments in brain scans that established a foundational basis for the phenomenon. Grapheme-colour, time-space, and coloured-hearing synaesthesia will be investigated for the purpose of this background research. Other forms include, but are not limited to time to colours, phonemes to colour, emotions to smell, general sounds to colours, musical notes to taste, and vision to taste.

2.2.1 Grapheme-Colour Synaesthesia

According to a statistical study by Sean Day involving 871 case reports, as in Figure 2.1, grapheme-colour synaesthesia makes up just under 65% of all recorded cases. As one of the

Table 8.2 Types of Synesthesia Reported by Synesthetes

Type of Synesthesia	Percentage Reporting	Type	Percentage Reporting
Graphemes → colors	64.9	Smells → temperatures	0.1
Time units → colors	23.1	Smells → touch	0.6
Musical sounds → colors	19.5	Sounds → kinetics	0.5
General sounds → colors	14.9	Sounds → smells	1.6
Phonemes → colors	9.2	Sound → tastes	6.1
Musical notes → colors	9.0	Sound → temperatures	0.6
Smells → colors	6.8	Sound → touch	3.9
Tastes → colors	6.3	Tastes → sounds	0.1
Pain → colors	5.5	Tastes → temperatures	0.1
Personalities → colors	5.4	Tastes → touch	0.6
Touch → colors	4.0	Temperatures → sounds	0.1
Temperatures → colors	2.5	Touch → smell	0.3
Orgasm → colors	2.1	Touch → sounds	0.3
Emotions → colors	1.6	Touch → tastes	1.1
Emotion → Smell	0.1	Touch → temperatures	0.1
Emotion → Taste	0.1	Vision → smells	1.1
Kinetics → sounds	0.3	Vision → sounds	2.6
Lexeme → Taste	0.6	Vision → tastes	2.8
Musical notes → tastes	0.2	Vision → temperatures	0.2
Personalities → smells	0.3	Vision → touch	1.5
Personalities → touch	0.1		
Smells → sounds	0.5		
Smells → tastes	0.1		

Note: Types of synesthesia found in 871 case reports.
Source: Sean Day, http://home.comcast.net/~sean.day/html/Types.htm, accessed February 2007.

Figure 2.1: Types of Synaesthesia Found in 871 Case Reports (Van Campen, 2010)

most researched out of the pool of types of synaesthesia, grapheme-colour synaesthesia refers to the perception of every single letter as a different colour. According to Hubbard, even when tested repeatedly at extended intervals, this type of synaesthesia is very consistent (Hubbard and Ramachandran, 2005). A common form of testing this type of synaesthesia involves research subjects performing detection tasks whereby target letters must be detected among deviation letters or target shapes embedded in a background of deflection letters. A repeatable outcome of this test is that when a synaesthete performs the target detection task, their performance is "superior over non-synaesthetes [only] for targets within a visual angle of about 10 degrees of fixation", therefore the repeatable "pop-out" effect is achieved (Laeng et al., 2004). Various neurological tests are then used to back up this effect so that it is reflected scientifically.

2.2.2 Time-Space Synaesthesia

Additionally, time-space synaesthesia has undergone significant research. This form constitutes synaesthetes perceiving time units as spatial forms and, as with grapheme-colour synaesthesia, it is consistent when tested repeatedly. Unlike grapheme-colour synaesthesia, however, it is triggered by temporal units such as a year, day, or month, rather than by external sensory stimuli (Brang et al., 2011). This can be investigated by the use of the *SNARC effect* (Spatial Numerical Association of Response Code) where subjects are presented with different months of the year to the left or the right. This is used on subjects who have strong numerical and spatial associations. They are asked to use their left or right hand to select a stimulus. When the stimulus and response-side polarities match, the participant responds faster than when they are not matched. In other words, if subjects see January on the left they are more likely to use their left hand in response to seeing January presented as a stimulus. (Price and Mentzoni, 2008). Other, less popular, forms of synaesthesia differ greatly from these types, however.

By investigating different synaesthesia types and the methods of research associated with them, we can gain an understanding of not only what constitutes *'synaesthesia'*, but what constitutes effective synaesthetic research. The types of synaesthesia discussed demonstrate the most employed forms of research. However, synaesthetes listening to music might see colours in addition to hearing sound in the form of *'chromaesthesia'*.

2.3 Chromaesthesia

Coloured-hearing synaesthesia, also known as chromaesthesia, is the perception of tones and/or tone-intervals as being of a distinct colour; in this way, it can be loosely defined as a *'tinted overlay'*. Discussing other types of synaesthesia makes it evident that there are multiple lenses that one can use to research and investigate various forms of synaesthesia, whether purely neurological or neurobiological, physiological, or in art where the exploration of the phenomenon has freedom of interpretation that is sensitive to each artist.

Chromaesthetic artefacts appear immediately after auditory perception, demonstrating an instinctive coupling of sounds and colours, as per Cytowic's diagnostic criteria discussed previously (Cytowic, 1989). In addition to this, three-dimensional artefacts, or shapes manifest in specific spatial locations of the *'tinted overlay'* of the synaesthete: "we find that when synaesthetes listen to music, their visual responses—colour, form, movement—may shift, surge, and ebb according to musical key, to musical pattern, to musical progression" (Marks, 2014). Sonic attributes such as pitch alter the brightness, contrast, or angular distance of the synaesthetic experience.

As we explore chromaesthesia over the last three centuries, its nuances become clear in comparison with the research of grapheme-colour synaesthesia. To speak to a chromaesthetic in specifics on a certain key, chord, tone, instrument, etc. often leads to assumptions being made due to the subjective nature of what is being described: "often described in terms of looking at a stained-glass window— and linked to equally complex sounds that go beyond just pitch or timbre" (Cytowic, 2018). Often the difference between two pitches, or the 'interval', in each musical arrangement rather than the pitch itself is responsible for the artefact seen by the

chromaesthetic. Chromaesthesia is evidently more subjective as it is about more than 'colours' and respectively measuring inputs and outputs. Therefore, there exists the propensity for a more artistic outlook on chromaesthesia compared to its synaesthetic counterpart forms.

Figure 2.2: Ocular Harpsichord
(Jones et al., 2012)

There are multiple accounts of such occurrences where composers and artists seem to lack the vocabulary to describe exactly what is occurring in their mind's eye when they hear a musical sound. An early exploration of light and visual music can be seen in the *ocular harpsichord* as in Figure 2.2. This instrument was outfitted by Louis Bertrand Castel with sixty multi-coloured lanterns that were set to be exposed when specific notes were sounded. It can be described as a musical instrument with the ability to paint sounds.

As musician Laurel Smith put it, out of tune notes are seen as variations of halos, sharp notes as white halos and flat as dark halos but no halos if they are inherent to the given key or 'group' of pitches. Higher sounds are lighter, lower sounds are darker but are all dependent on the type of instrument the tone is originating from. Laurel describes these as 'gross oversimplifications' as her 'tinted overlay' can be seen in three dimensions (Cytowic, 2018).

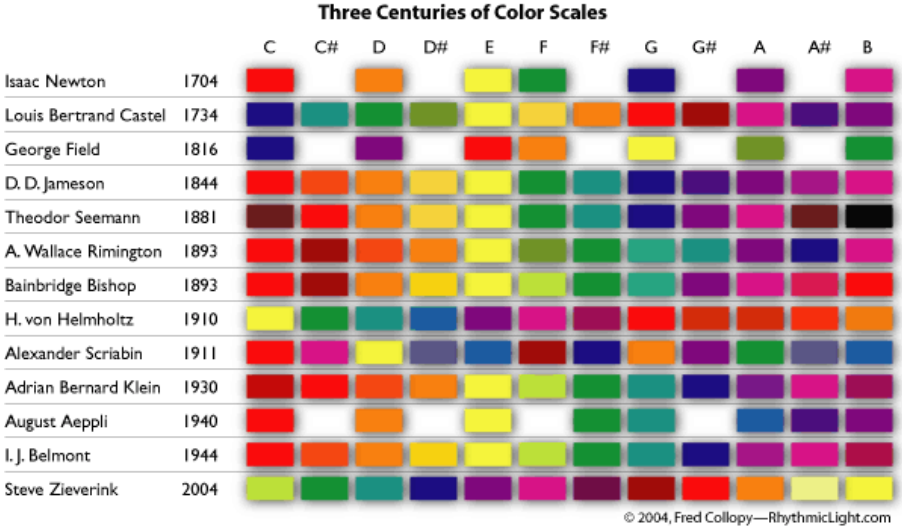

Figure 2.3: Three Centuries of Colour Scales
(Brougher et al., 2005)

Lawrence E. Mark's *The Unity of the Senses* described Isaac Newton's belief in "a real analogy between elementary colours and the notes of the musical scale" (Marks, 2014). Newton named seven elementary colours of the spectrum, each parallel to a note on the musical scale. However, a standardized method of reducing this phenomenon to something tangible has been documented throughout history by Fred Collopy in 2004 in his Three Centuries of Color Scales chart, as in Figure 2.3. The chart depicts mappings of colours to specific notes as documented throughout history, from Isaac Newton in 1704 to Theodor Seeman in 1881 to I. J. Belmont in 1944. Marks reiterates this by stating: "[the principle that underlies the colour organ is that] some musical synaesthetes claim that particular musical notes regularly and repeatedly arouse specific colours" (Marks, 2014). The key takeaway from this chart is to exhibit the differences in the perceptions of colour to note mappings throughout history. There are similarities between several notes, such as the note 'C' being a variation of a deep red colour, 'D' being an orange hue, and 'A' being a shade of purple.

2.3.1 Visual Manifestations

Chromaesthesia is one of many types of abstract visual expressions. An appropriate analogy to describe the perception of visual manifestations of chromaesthesia as seen by a synaesthete would be to liken it to fireworks- it would be very difficult for an individual to articulate the exact shapes and combinations of the fireworks as they exploded and how they morphed and formed as time went on. What would be immediately identifiable would be the colour of

the explosion and the basic shape made by the firework. Discussing this analogy gives us a foundational basis from which to investigate Heinrich Klüver's *'form-constants'*.

Visual manifestations, allso known as *'qualia'*, are the result of duplex input to colour perceptions in centres of the brain, from both visual and auditory inputs. Redness, brightness, and sharpness are all subjective characteristics of perception which influence visual manifestations occurring in synaesthetic experiences. In the 1920s, Heinrich Klüver historically categorized recurring geometric shapes into tunnels, spirals, honeycomb gratings, and cobwebs as *'form-constants'*. This term is used to describe the consistency of configuration that is present in all visuals of this type which applies to both drug-induced hallucinations and chromaesthetic artefacts (Bressloff et al., 2002). Studies on the direct and indirect responses to a series of experiences reveal that chromaesthesia can be induced with chemical agents.

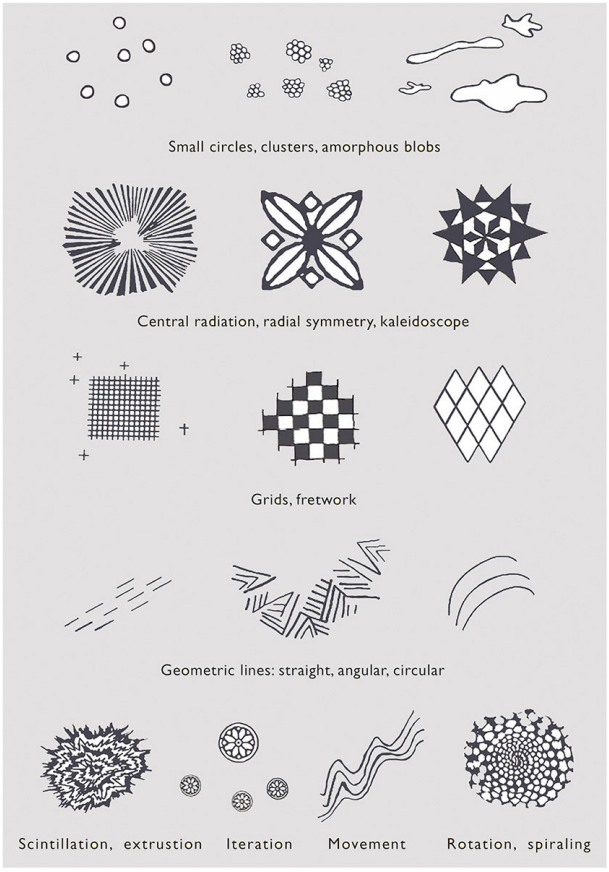

Figure 2.4: Heinrich Klüver's Form-Constants
(Steen, 2019)

Also coined as hallucinations, Klüver states that all artefacts consist of shapes in one of these categories with varying degrees of irregularities in chromaesthetic experiences: "We may call them form-constants, implying that a certain number of them appear in almost all mescal visions and that many "atypical" visions are upon close examination nothing but variations of these form-constants" (Lewis-Williams, 2004). The nature of this synaesthetic perception is that the brain appears to respond in a finite number of ways to a diverse range of inputs or

sounds in this. 2.4 demonstrates the diverse nature of axial or radial symmetry found in the world of art to natural phenomena like synaesthesia to cave paintings of primitive cultures.

The term *'form-constants'* would lead one to believe that the artefacts seen by the synaesthete are stationery and invariant, when they are continuously remorphing in an "incessant interplay of concentric, rotational, pulsating, and oscillating movements through which one pattern replaces another." (Cytowic, 1989). Concentric organizations explain the experience's spatial and temporal flow in greater detail, while colour changes provide finer gradation.

In a way, chromaesthesia reveals the existence of peculiarities of perception as it differs from one person to another, even if it offers a universal base set for all peoples. The way this is investigated, however, differs greatly compared to other forms of synaesthesia due to the complexity of exploring a synaesthete's percept.

Chapter 3

Field Review

Investigation into the state-of-the-art technology surrounding grapheme-colour and sound-colour synaesthesia types reveals a disparity between the respective types when it comes to research analysis tools. The synaesthesia battery type provides us with a 'gold standard' of research testing in this area. It is an online test consisting of eighty questions. The test standardised methods for comparing, contrasting, and pooling synaesthetes through a calculated set of questions and research methods by collecting data from subjects. Websites like synaesthesia.com act as approximate tools of whether a subject has synaesthesia or not and the Polychrome Project offers a modern method of intuitively capturing synaesthetes percepts with emphasis on richness and detail.

3.1 Synaesthesia Battery Test

As discussed, synaesthesia has undergone many iterations in various sectors of psychological, neurobiological, and in artist research throughout the last 200 years. The issue with this lineage of research is that there has been no single protocol for comparing, contrasting, and pooling synaesthetic subjects across groups. The standard battery test therefore provides a quantifiable scoring system and a standard phrasing of questions for data collection and comparison. These standardized tests provide procedures for testing and comparing subjects for analysis in synaesthetic research (Eagleman et al., 2007).

The test questionnaire is an online test consisting of eighty questions that collects data on everything from the type of synaesthesia the test subject believes they have, to quantifying traits among groups of synaesthetes, directing the subject to appropriate tests online for their form of synaesthesia, and neurophysiological data such as links to dyslexia, head trauma, tumours, and autism (Eagleman et al., 2007). The reason that it is an online test is to facilitate sharing of data between subjects and researchers on a web portal. The focus of research in the standard battery test is a grapheme-colour consistency test.

The grapheme-colour consistency test works by firstly presenting participants with randomly sorted graphemes, such as letters, numbers, weekdays, and months. Participants chose the colour that best matches their synaesthetic response for the grapheme from a colour palette.

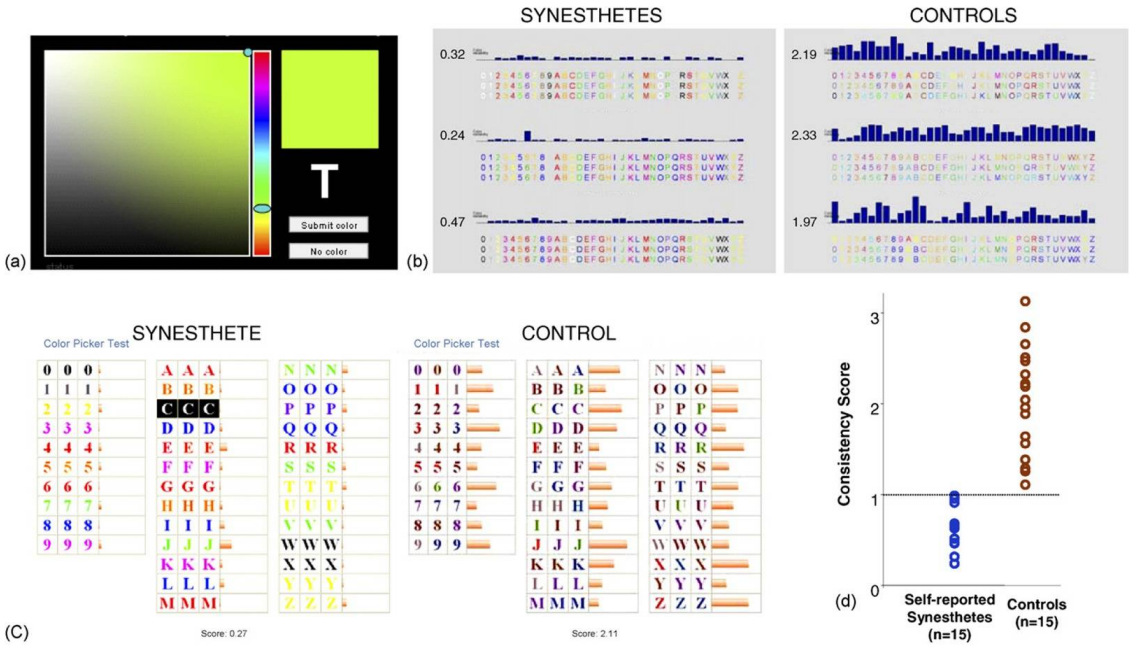

Figure 3.1: Colour Match Consistency Testing
(Eagleman et al., 2007)

This is repeated three times. The blue bars as illustrated in Figure 3.1 indicate the calculated distance in colour space between each answer. This is the range of the colours chosen by subjects of the test. Subjects who do not experience synaesthesia, or 'control' participants, are then instructed to imitate synaesthesia. This allows for methods such as free association and associative memory to be used to test for consistency in synaesthetic subjects. This reveals that controls match colours in a much less consistent manner when compared to those with synaesthesia. Subjects who had synaesthesia scored below 1.0 in their consistency score while controls had much more variance in their colour choices. (Eagleman et al., 2007). Participants of the synaesthesia battery register on http:///synesthete.org and are given the option to enter in a researcher's email address, allowing them to share test results privately. This provides a means for worldwide and independent research of synaesthesia.

At present, technological limitations have inhibited synaesthetic research. Clear examples of this are taste to shape and taste to colour synaesthesia, yet the synaesthesia standardized battery test provides a flexible framework for continued research on the topic.

3.2 Other Tests

3.2.1 Synaesthesia.com

Based on the synaesthesia battery test, the test outlined by synesthesia.com as developed by Sensorium operates as a grapheme-colour consistency test but loosely tests if the subject in question has synaesthesia. It is advertised as a not 100% correct diagnostic tool and uses 7 tests to determine whether someone has synaesthesia or not.

Tests of this type adopt the hue, saturation, and brightness (HSB) colour system to choose a colour that matches best to a synaesthete's percept. The test in question, however, adopts 16,777,216 colours for the RGB space as the colour picker portion of the test. The test claims that one can experience synaesthesia through meditation by providing over 150 exercises to 'engage all senses' to "support [the user] in discovering and experiencing synaesthesia in [their] daily lives" (synesthesia.com, 2021).

3.2.2 The Polychrome Project

The Polychrome Project is a research project on synaesthesia and language based at the University of Sussex, UK. The aim of this project is to study synaesthesia as it is actually experienced by synaesthetes. It does this by developing more modern synaesthesia research methodologies such as the use of a multi-colour picker as opposed to a set number of colours and colour names from a list to choose from, like in traditional grapheme-colour research.

The project takes psycholinguistic theory into account and acknowledges how many grapheme-colour synaesthetes experience compound words as two different colours. Therefore, the project proposes an adequate database to test this theory. Finally, the project created a resource for synaesthetes and researchers. For synaesthetes, this gives them the opportunity to share their experiences to their family and friends in a way like never before. For researchers, the findings allow them to examine the database and test different hypotheses on how language and synaesthetic experiences interact.

From conducting a field review on the available tools to test for synaesthesia, a recurring theme amongst the majority of testing tools seems to be that they are no longer operational or have not undergone updates for maintenance purposes to ensure that they are kept online and operational. This is the case for chromaesthetic.com, for example, which claims to be a sound-colour synaesthesia/chromaesthesia test- a needle in the haystack of grapheme-colour research tools. Other tests, such as the *MULTISENSE Adaptable Synaesthesia Toolkit* which claims to be a "library of tests for synaesthesia and related mental processes" (syntoolkit.org, 2021) including grapheme-colour synaesthesia, sequence-space synaesthesia and sound-colour synaesthesia, which is only available for viewing by researchers and subjects due to GDPR reasons, further highlights this issue regarding synaesthesia research.

Chapter 4

Literature Review

The merging of computer based interactive creative coding and synaesthetic visualizations or 'form-constants' can be approached in several ways. With the context of viewing synaesthesia as an extension of the field of neurobiology and in addition to art allows for a discussion on the relevance of art, specifically visual music, within coding and computational art. As discussed in the previous analysis sections, sound-colour synaesthesia is an involuntary evocation of an experience of colour, form, and moment from the perspective of the synaesthete. The aforementioned 'form-constants' as outlined by Klüver depicts a geometrical basis for various drug and non-drug induced hallucinations, which encapsulates the synaesthetic percept. Therefore, a discussion on generative art theory is required to gain an understanding of the lens within which to view chromaesthesia. Following on from this, an analysis of creative coding within the field of computer science will allow us to frame an appropriate approach to translating sound-colour synaesthesia visualizations to a creative coding context.

4.1 Taxonomy of Synaesthesia and Hallucinations

There are many parallels in the histories of abstract art and synaesthesia. Therefore, viewing these visualisations through the lens of art gives us a greater understanding of how to translate them to a creative coding context. *A Taxonomy of Abstract Form Using Studies of Synesthesia and Hallucinations* by Michael Betancourt provides us with a more thorough classification of these visualisations with the purpose of understanding how to replicate them using computer-driven generative systems.

At the outset of his text, Betancourt states: "For centuries artists and philosophers theorized that there should be a visual art form as subtle, supple, and dynamic as auditory music—an abstract art form, since auditory music is basically an art of abstract sounds." (Betancourt, 2007). He outlines the various characteristics by which to classify form-constants to provide a framework for generative systems. The geometrisation of form-constants for the purposes of replicating them using computer-driven generative systems can be classified into three categories: shapes as symmetrical or repeating, colour or 'surface', and motion (Betancourt, 2007).

These properties can be viewed as variables within the chromaesthetic experience and are translatable to a generative art system, as will be discussed.

The 'shape' classification outlined by Bentancourt constitutes the form-constants that are "frequently repeated, combined, or elaborated into ornamental designs and mosaics of various kinds; and the elements constituting these forms, such as squares in the chessboard design, often have boundaries consisting of geometric forms" (Betancourt, 2007). By this, Betancourt returns a set of factors whose relationship and combination can affect the form's look, curvature, and organisation. Symmetry differs from repetition in that symmetrical patterns must be regular, whereas repeated forms might be asymmetrical. The more axes a symmetrical structure has, the closer it gets to infinite symmetry. Forms can be repeated in any number of dimensions or scales, including lateral, vertical, and recursive repetition.

Following on from this, the property of colour or 'surface' encapsulates hue, saturation, and brightness values, as well as luminosity or the 'relative brightness'. The key implications of these properties for a generative art system means that they are directly translatable to a variety of colour systems supported by the system, such as *HSL* (hue, saturation, lightness) and *HSV* (hue, saturation, and value). These distinctions in properties are made to differentiate forms from each other where they might be similar otherwise in all other properties: "The qualities of the forms, when differentiated from their shape, symmetry and motion, can be described by two variables: color and luminosity" (Betancourt, 2007). In this way, the varying qualities of colour, due to the breadth of the electromagnetic spectrum, allow us to adequately describe forms. The outlined property of 'colour' therefore has a significant impact on the way one can research chromesthesia using computer-driven generative systems as it reduces the need for exact replication of each of the properties outlined by Betancourt's. Abstracting all other properties from the forms, form-constants can be adequately represented using colour and luminosity values as supported by a variety of colour systems in generative art.

The final parameter is categorised as 'motion'. Betancourt classifies 'time' within this taxonomy as form-constants are not unchanging or immobile frames: "By necessity, [motion] requires and implies both time and a volumetric space... The apparent depth of this space is either two-dimensional (flat) or three-dimensional (moving within a volume)" (Betancourt, 2007). By this definition, motions can be divided into their own distinctive categories. Regular motions consist of movement along the same axis (x, y, or z), while irregular motions consist of a complex series of regular motions. Irregular motions establish a shape's perceivability within 3D space. For example, the shape may arc as it follows a curve, move in a circle or spiral to become concentric, rotate, or scale to be perceptibly bigger or smaller if multiple series of movements on the z-axis are observed.

This oversimplification of the classification of form-constants, specifically their colour properties, can more adequately be described as a spectrum of colours. Investigating Betancourt's writings provides a basis on how to depict synaesthetic form-constants in a manner that is translatable to computer-driven generative systems. Before proceeding to the specificities of how this can be achieved, it is first important to gain an overview of the generative art and its relationship with creative coding as an artform. For this purpose, several texts will be explored.

4.2 What is Generative-Art?

A key principle to generative art that differentiates it from traditional art is that it allows for the input of a computer as part of the creative process, or as part of the art itself, depending on the desired result. As Margaret A. Boden and Ernest A. Edmonds eloquently put it: "In principle, these machines can (and do) offer us an entire symphony orchestra, and an infinite set of visual images and sculptural forms–indeed, an infinite range of virtual worlds" (Boden and Edmonds, 2009). Computer-generated art doesn't strictly have to be a case where the user gives an input and the computer outputs a piece of art, but often allows for direct involvement in the outcome with the possibility of the additional aspect of irregularity generated by the computer itself. All generative systems can be said to work within a larger environment from which they can obtain information or input on which to act. This is the basis of 'creative coding', as outlined by Ina Greenberg.

4.3 Creative-Coding

Creative coding requires a reworking of the methodology used to go about programming, as discussed by Ira Greenberg's *Processing: Creative Coding and Computational Art*. In one of the opening sections of the book, he discusses the way traditionally programmers approach problems. He uses the examples of a friend of his, comparing how his analytical approach to various stages of the development pipeline of the Processing programming language differed from his own, stating that "to be really effective, you need to approach a problem both analytically and intuitively or even holistically" and "as children, most of us scribbled with crayons and learned to make things without much planning. We learned to express ourselves with marks and gestures and weren't too self-conscious about making a mess." (Greenberg, 2007). The way a programmer tackles a coding problem is often framed by the way they learned to code. The way in which one instinctively figures out an approach to a painting or a design problem can be reflected in one's approach to coding as it poses new lenses from which to view coding.

More specifically, the aforementioned painting analogy is appropriate when discussing the JavaScript library p5.js, an interpretation of Greenberg's Processing, as it subscribes to his view on creative coding. This library provides a language for teaching programming to design and art students, as well as a more straightforward manner for more technical students to deal with graphics through a full set of sketching capabilities, where the entire browser acts as a 'sketch', with JavaScript acting as the paintbrush, and objects, text, video, webcam, and sound all act as different tools in the toolbox. This library and ethos will act as the foundational basis for addressing the proposed research question.

From investigating these texts, it is clear that there are multiple approaches to translating sound-colour synaesthesia visualizations to a creative coding context. The texts discussed provide us with a simple view of the topics being discussed, when they act as oversimplifications of fields that are grounded in neurobiology and geometry. This begs the question of why more texts haven't addressed the relationship between generative art and synaesthesia in more specific terms, as will be addressed in the analysis of the problem.

Chapter 5

Design

5.1 Analysis of the Problem

Research on synaesthesia has only in the last 40 years reached to a point where researchers are able to quantify what constitutes a synaesthete. If you dissect this further, it becomes evident that the research of chromaesthesia makes up a small proportion of synaesthetic research. There are a multitude of research tools concerning grapheme-colour synaesthesia, yet not many tools for music-colour synaesthesia or chromesthesia exist, even though its first mention was 200 years ago. Therefore, as outlined in the research question statement, the problem being investigated can be briefly described as such: how to conduct research for sound-colour synaesthesia in the way most grapheme-colour tests do, such to map chromaesthetic associations for consistency testing purposes.

As discussed, there is an evidently clear disparity between the research conducted on and research tools available for grapheme-colour synaesthesia and sound-colour synaesthesia. There are many tools to test whether an individual experiences synaesthesia, mainly grapheme colour synaesthesia, yet such tools are not prevalent when researching sound-colour synaesthesia or chromaesthesia. The subjectivity of chromaesthetic experiences can often make categorising a difficult task. Because they may not match the specified definitional requirements, many synesthetic individuals may be omitted from research. This is especially important because synesthetic persons can have a hard time differentiating between their own experience of reality and that of non-synaesthetic people.

Therefore, it is difficult to analyse how to effectively map chromaesthetic associations without making assumptions and taking shortcuts, unless, with new technology, you could see exactly a synaesthete percept. From research discussion it is possible to abstract this issue even further: how do you portray something as subjective as chromesthesia in a way that is translatable to research, such as a consistency test? A way to address the proposed issue is to replicate the chromaesthesia result format outlined in Fred Collopy's *Three Centuries of Color Scales*. Without trying to reinvent the wheel, this provides a basis to record sound-colour synaesthesia results in a format that dates to 1704.

5.2 Introduction

The proposed solutions offer two considerably different approaches to the problem of mapping synaesthetic artefacts for consistency tests. The first involves the assembly of a Virtual Studio Technology (VST) plugin built using a combination of the JUCE framework, a C++ application framework for creating VST, VST3, AU, AUv3, RTAS and AAX audio plug-ins, and/or Max4Live devices. (Robinson, 2013). This solution is user facing with the target audience being musicians, artists, and producers who wish to experience synaesthesia as part of their creative process. The second solution differs significantly from this as it implements a tool where researchers would present a stimulus to a subject and the subject would be able to modify the output to match their synaesthetic percepts. Built using a combination of React and P5.js, the proposed solution would adapt the result format of Fred Collopy's *Three Centuries of Color Scales*. (Brougher et al., 2005).

5.3 Design of a Solution

The aim of the first solution is to create a tool in which the end user, a musician or anyone using a Digital Audio Workstation (DAW), would be able to accurately replicate a synaesthetes percept in real-time as part of the 'creative process'. The purpose of the tool would be to act as inspiration or an influence over the 'ideas' part of this process, while simultaneously portraying chromaesthetic properties, such as colour, in a way that is translatable to a consistency test. The result of this influence of chromaesthetic interpretations can influence anything from live performances to videos accompanying musical performances, or any other creative aspect to go alongside the music. An example of this can be seen in https://youtu.be/qlEP_pjcFDQ.

5.3.1 Solution One

An iteration of this would include using a JavaScript framework whereby a web application would be created. The user would upload an audio file. Following this, the web application would analyse the audio file and render a visual element that morphed and changed according to the waveforms detected as if it were a chromaesthetic artefact. An export option would be presented where the user could download a video file, with future development including a plugin for music creators that offers real-time rendering, Max4Live communication, and exporting options to render it as a VR environment.

A strength of this solution is that the artefacts illustrated would be dynamic or reactive in nature. No two artefacts would be the same in this way. Integration with music creation software would be a seamless experience for the user, providing an unhindered experience of chromaesthesia. However, the analysis of waveforms would require a large sample size of synaesthetes to conduct consistency testing. As it stands, the research required to build a system in this way doesn't exist. If the user was a synaesthete, they would not be able to interactively provide feedback to improve the accuracy of the artefacts shown. Audio Digital Signal Processing (DSP) requires extensive knowledge of C++ which would extend the development time of this project significantly.

5.3.2 Solution Two

Alternatively, the second solution involves a combination of JavaScript libraries including React, P5.js and Tone.js. React is a declarative JavaScript framework for building interfaces or UI components (Fedosejev, 2015). Accompanying tools provided by React support developers in the development process. It can act as the basis for both single-page web applications or mobile applications. Tone.js is a Web Audio framework that allows for creative interactive music in the browser using JavaScript. As discussed previously, p5.js is a JavaScript library that places an emphasis on creative coding, making the coding process accessible to artists, designers, educators, and beginners alike.

The proposed design is much like a modern iteration of the ocular harpsichord, as developed by Louis Bernard Castel. Aimed primarily for research purposes, this tool assumes that the test subject experiences sound-colour synaesthesia. The proposed research tool operates similarly to the grapheme-colour consistency test as outlined in DM Eagleman's *A standardized test battery for the study of synesthesia* outlined in chapter 3, to be a sound-colour consistency test by allowing subjects to map synesthetic colour associations as they perceive them.

Non-synaesthetes act as controls in this instance to conduct the consistency test. The proposed solution consists of a selected set of form-constants converted to p5.js creations of 'sketches'. Alongside this is a controlled synthesiser in the form of twelve piano keys with an individual colour picker for each note. This gives synaesthetes the opportunity to map the colour they experience when a certain tone is played alongside a given form constant. The 'sketches', built using p5js, act as near-accurate replications of form-constants. Connected to these sketches is a controlled synthesiser built using Tone.js. By 'controlled', we mean that every note has the same attack, decay, sustain, and release (ADSR) that is not adjustable by the research subject. In this way every user hears the same tone respective of the note they click. The result of this is a tool which allows the researcher to export the synaesthetes colour coded piano in the format of Fred Collopy's *Three Centuries of Color Scales*, as in Figure 2.3. In this way, researchers can explore the relationship between the inputs and the outputs in people who experience chromaesthesia.

A strength of this solution is that it allows subjects to provide feedback to presented stimuli by way of a colour picker tool. This spectrum allows for a deeper analysis of the colour aspect of synaesthetic experience as the colours available are not limited to simple colour names like 'red', 'blue', or 'green'. As the proposed solution is directed on the colour aspect of synaesthetic experiences, the constants in the proposed research tool, such as the synthesiser and set number of notes and form-constants, allow the colour aspect to be the only variable in the research. However, by addressing only the colour property of the synaesthetic experience, the proposed solution omits other equally important aspects of the chromaesthetic percept like motion, spatiality, or shape of the form constant. The potential for libraries to be incompatible is also an apparent weakness of this solution.

5.4 Proposed Solution

Solution two acts as a more comprehensive solution to the proposed research question. Through combining React and p5.js, the synaesthetic experience can be said to be more adequately represented were it designed in this manner. Providing several constants while placing the emphasis on the research on the 'colour' aspect of chromaesthesia is reflective of research done in the past and, in this way, provides a modern way of researching chromaesthesia.

Chapter 6

Implementation

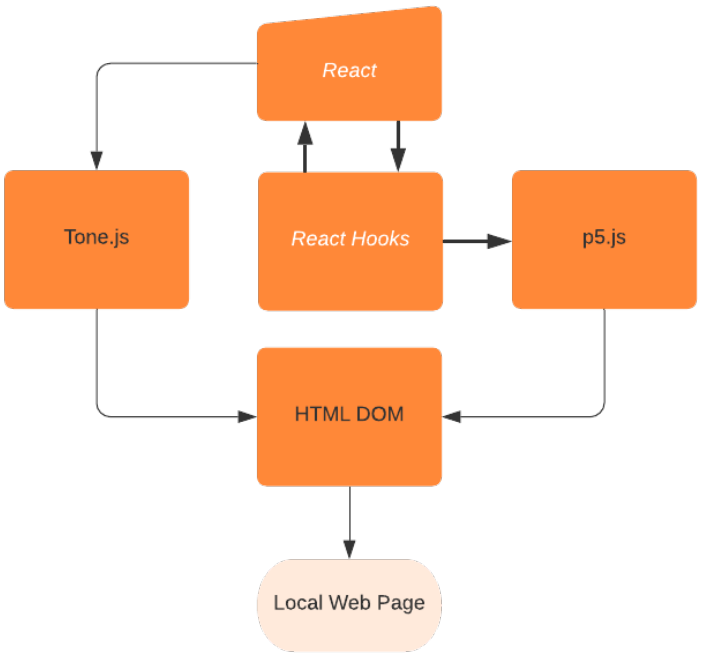

Figure 6.1: Implementation Flowchart

The development process includes a high level view of how the system is assembled, made up of several tools that aid in the implementation of React and p5.js. React as the basis of the implementation with support for React Hooks allows for the integration of p5.js form-constants in 'instance mode' within this framework, while Tone.js provides the audio aspect of the project. Moreover, options for exporting the data input of synaesthetes is possible by using 'Save Image' and 'Save Data'. The manner in which the proposed solution is implemented can be illustrated using Figure 6.1:

6.1 Development Process

The development process of the research tool consisted of several tools acting in unison to either support either the efficiency of the development process or for live testing and debugging purposes. Among these tools are Visual Studio Code, React, Chrome, GitHub, and Heroku.

The terminal feature of Visual Studio code allowed for a fully integrated terminal command-line interface as if you were using PowerShell or command prompt, all within the VS Code environment. The integration of VS Code, GitHub, and Heroku provides a seamless testing environment. Heroku provided a means by which to build, run, and operate the application entirely on the cloud. Whenever the GitHub repository is updated, the project is automatically deployed by Heroku using npm build. This serves the purpose of debugging on different devices and testing the build process on a cloud server. Major changes to the code were managed using Git in the VS Code command line. Regular changes were committed to the project and uploaded to the repository using the push feature. Other third parties like npm, Material UI, Material-UI-Color (used for the colour picker), speed up the development process drastically. To code all these features individually would drastically slow down the development process. At the core of this project is the use of a JavaScript library 'React', which will be discussed in more detail.

6.2 Architecture

For the purpose of this section, the following flowchart will illustrate the process of how data flows in the system of the proposed solution. Using React as a foundational basis, the solution uses several libraries that interact with each other. In this case, p5.js acts as a way to sketch form-constants. These are created through taking existing projects and converting them to 'instance mode' for compatibility with React. Instance mode allows p5.js to act independently by wrapping up all p5.js related code under a particular name. Tone.js provides the audio aspect of the implementation while the colour picker allows subjects to tweak the synaesthetic experience according to their percept. Finally, export options for researchers allow for easy downloading of the data and facilitates the tool for research purposes,

6.2.1 React

For the purposes of the implementation of the proposed solution, React can be described as a declarative JavaScript framework for building interfaces or UI components for single page applications. It provides a framework for building modern applications in a way that is more efficient in the development process than other tools, libraries or using vanilla JavaScript (Fedosejev, 2015). JavaScript XML or JSX, similar in appearance to HTML, provides a structure to component rendering within React. This is the basis of implementing 'components'. These are independent and reusable snippets of code that help us to write clearer and more concise code. All elements seen on the webpage of the implementation are represented in this function, everything from the Material-UI layout components to p5.js to the Tone.js synthesiser and the colour picker.

'Create-React-App' is characterised as a framework within React that facilitates the automatic creation of a development environment It supports third party integration with live CSS and JavaScript editing and an integrated toolchain that facilitates scaling to many files and components, all with little configuration needed. It creates a frontend build pipeline from which to build upon and acts as a foundational basis with which to build upon. This is crucial to the development process as every edit was instantly reflected in a local web page. Following on from this, React functions, classes and other lines of code are implemented into the project. A static webpage is generated based on the code. The third-party library 'Material-UI' is crucial in this step as it allows for easy prototyping in the development process. It is an open-source project that features React components that implement Google's Material Design. CSS baselines provided by Material-UI maintain a level of congruency within the project. In this instance, 'Grid', 'Paper', and 'Button' were the primary components that made up the implementation of the proposed solution, providing consistency in the layout and appearance. These components are written in JavaScript XML, and are returned as HTML in the static web page. Material-UI was used to horizontally split the page into two whereby the p5.js sketches in the top half with the Tone.js synthesiser and colour picker buttons in the bottom half by using Paper and Grid components.

Within React, Components can be defined in two ways: as functions or as classes. A React function component is a JavaScript function that accepts properties, such as colour, as an argument and returns a React element. A React class component, on the other hand, requires you to extend from React and create a render function that returns a React element (Fedosejev, 2015). The following gives the component access to React's various component features while naming it 'Chromesthesia' while the three dots represent the code that is structured within the class component, as in Algorithm 1:

Algorithm 1 React Class Component

```
class Chromesthesia extends Component {
    ...
}
```

For every class component in React there must be the functions constructor() and render() included. The render() function is used to convert a React element to a web page. Within React class components, as outlined above, a constructor function can be called at the start of the class as 'constructor()'. This can be described as a function that is called when the function is called before everything else. Within the constructor function, component properties are kept as objects called 'state'. These allow the developer to reuse stateful behaviour between components. These state properties are not static in the way that values are stored in them, belong to the component in question, and are used to provide a shorthand way of reading and writing information within or between components and are updated using the 'setState' hook. When the 'state' object changes, this re-renders the component according to the new value(s). This is vital for the implementation of the interaction between React and p5.js, as they enable the functionality of React Hooks, which will be described in further detail. Before continuing it is important to gain an understanding of the integration of Tone.js and p5.js within the React

framework.

6.2.2 P5.js

The primary purpose of p5.js in the implementation of this solution is to display dynamic form constants within a React environment that can change colour depending on a synaesthete's input. The generation and display of form-constants similar to those experienced by synaesthetes makes use of p5.js' instance mode and a third-party library 'react-p5-wrapper', which will be discussed in further detail.

For each p5.js sketch there must be a draw and setup functions. The setup function is called once the program starts and is used to define initial environment properties such as the screen size. Within this implementation, the setup function first clears the previous sketch, sets the colour mode to HSB, sets the framerate to 30 frames per second (FPS), and defines the width and height of the sketch. Following on from this, the draw function specifies what should be looped through at specified rate- the framerate as defined in the setup i.e., the draw function is iterated over at a rate of 60 times per second for sketches one and two and 40 times per second for sketch three.

Figure 6.2: Central Radiation Form-Constant

Figure 6.3: Rotating Form-Constant

Figure 6.4: Scintillation Form-Constant

The form-constants in this project are made by converting existing sketches that resemble form-constants and tweaking them for accuracy. Sketch one (an interpretation of https:

//editor.p5js.org/cdaein/sketches/BkxxNlniW) (cdaein, 2021) as the default sketch depicts a tunnel form constant, or 'central radiation', as in Figure 6.2. Sketch two (an interpretation of https://editor.p5js.org/AhmadMoussa/sketches/TxnpilImt) (Moussa, 2021) is a rotating or spiralling form-constant while sketch three (an interpretation of https://editor.p5js.org/EdCavett/sketches/DRvCPgi-) (Cavett, 2021). illustrates scintillation in Figure 6.3 and Figure 6.4 respectively. Each p5 sketch in the project is an iteration of pre-existing p5.js sketches. These sketches act as individual components that can be imported into a React component as such: import sketch from "./sketch"; etc. This is made possible using 'instance mode' within p5.js, which will be explained in further detail.

6.2.3 Tone.js

A controlled synthesiser is created by defining a PolySynth globally with a fixed note length using the third-party library Tone.js. The PolySynth is an synthesiser instrument that is capable of producing multiple tones at one. From here, piano notes are created using HTML buttons with className for styling adjustments and an onClick event 'playNote' to bind it to a specific note and keyboard key, and to handle setting a colour. A handleKey() function is created to bind each note to a keyCode. Within algorithm 2, keyCode 65 represents an actual key on the keyboard. In this case, the letter 'A', meaning that when the letter A is pressed it will play the note 'C4'.

Algorithm 2 keyCode Binding

```
if(e.keyCode === 65) {
    this.playNote("C4");
}
```

6.2.4 Instance Mode

The issue with interactivity between React and p5.js is that they are not natively compatible, so a third-party library is required. For this purpose, the library 'react-p5-wrapper' is used. In order to utilise the react-p5-wrapper library to 'connect' React with a specific p5.js sketch, instance mode is required. In its default mode, p5.js does not allow you to add multiple sketches to the screen. Therefore, instance mode allows p5.js to act independently by wrapping up all p5.js related code under a particular name. In this case, every p5.js property is prefaced by 'p.'. This ensures that there are no conflicts with other libraries. It is used within this implementation to integrate p5.js sketches in React apps, facilitated by the third-party component 'react-p5-wrapper', where properties are passed to the p5 sketch. Figure 3is how instance mode is used in this project whereby the three dots represent the code that is structured within:

Algorithm 3 Instance Mode

```
export default function sketch (p) {
    ...
}
```

The implication of instance mode is that not only are the draw and setup functions defined within the p5.js sketch, but also a new 'myCustomRedawAccordingToNewPropsHandler' function, as defined by react-p5-wrapper. This function is what allows us to send information from React to p5.js, such as states. Within this function, the state object 'p5Colour' is used to send a colour value from React to p5.js. In this way, the state 'p5Colour' is unidirectional in that p5.js can read the value of the state but cannot update it, as illustrated in the Figure 6.1.

Algorithm 4 myCustomRedrawAccordingToNewPropsHandler

```
p.myCustomRedrawAccordingToNewPropsHandler = (newProps) => {
    if (canvas) {
        p.fill(newProps.p5Colour);
    }
};
```

As stated, every property of p5.js must be prefaced with 'p.', as illustrated by 'p.fill' in algorithm 4. This is the function fill(), which is used to set the background colour of the sketch using the hexadecimal colour mode. 'p5Colour' refers to the colour of the background of the sketch in this way.

Algorithm 5 P5 Wrapper Component

```
<P5Wrapper sketch={this.state.sketch} p5Colour={this.state.p5Colour} />
```

Centring our attention back to React, algorithm 5 outlines the definition of the react-p5-wrapper component as it exists within the render() function of React. It allows us to pass properties, such as a colour value, from the React to the p5.js sketch and for them to be immediately updated.

To allow the user to change the sketch (the form-constant showing on the screen), three buttons are created on the web page labelled 1, 2, and 3 respectively. The property named 'sketch' in algorithm 5 refers to the current p5.js form-constant being shown on the screen. The object 'sketch' is stored as a state. Using an onChange event in the buttons, the web page changes the 'sketch' state property to the sketch that was clicked and is updated on the web page. These properties are accessible from inside the component's class to which the property is passed by using 'this.state'. The keyword 'this' is used to refer to a JavaScript element depending on the scope or use of its context. In this case, it refers to the state of the class component it is used in. The properties being sent in this case depend on the colour picker which uses a 'handleChange()' function that is responsible for assigning the colours to the Tone.js notes.

Algorithm 6 handleChange() Function

```
handleChange = (value, key) => {
    ...
}
```

6.2.5 Colour Picker

A colour picker is created as a component within a HTML button using the material-ui-color third-party library. Using CSS, the colour picker components were appropriately placed on top of each note. While placing a component within a HTML button is not best practice, it is appropriate for the purposes of the implementation of this solution. The default value of each note is modelled off I. J. Belmont's colours scheme in *Three Centuries of Colour Scales*, stored as individual states of the component. An 'onChange' attribute named 'handleChange()', as in algorithm 6, is included in the colour picker component. This passes the properties 'value', used to refer to the updated colour chosen from the colour picker dialogue, and 'key', the respective note being changed. This indicates that whenever a new value is picked, the 'handleChange()' function will be called. Placing this within a HTML button component facilitates two key features of the implementation: the storing of the note to be used at a later point and the immediate changing of the colour of the sketch to the given note colour. The 'value' argument as in algorithm 6 corresponds to the hexideciaml value of the picked colour, which requires some adjustments before its state is updated:

Algorithm 7 Convert Hex Value

```
var hex = stringifyObject(value.hex, {
    doubleQuotes: false,
});
hex = hex.replace(/´/gi,");
```

The hexadecimal variable in algorithm 7 corresponds to the value sent from the colour picker (value.hex) and is converted to a hexadecimal format to be compatible with p5.js. Key refers to the note which the value was sent from. The value returned is a JSON object. As p5.js does not support JSON objects, it must be converted to a string using a third-party library called *'stringifyObject'*. This also removes the double quotations of the JSON object value. The colour of the sketch is then set to the note that was pressed and stored into setState using 'if' and 'else if' statements. For each note, a hash symbol had to be added, as in algorithm 8:

Algorithm 8 Hex Value Assignment

```
if(key === "C") {
    this.setState(c: '#' + [hex]);
}
```

This process is repeated for all twelve notes. While the handleChange function is responsible for setting the state value of each note, the playNote function handles setting the state of the current colour of the sketch which is read by p5.js or the interaction between React and p5.js. An 'onClick' attribute is added to the button element of each note where when the note is clicked, the playNote function is called with the arguments stating the note that is to be played. Each note is mapped to its relevant state property by creating an array of notes called 'colourForNotes', listing the note assigned to 'this.state.x', x indicating the note. When the playNote function is called, it sets the 'p5Colour' state to the respective note and colour, as in algorithm 9:

Algorithm 9 Change Sketch Colour

```
this.setState({p5Colour: colourForNotes[note]});
```

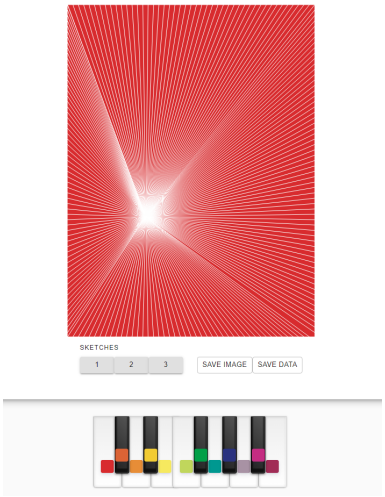

Figure 6.5: Screenshot of the Implemented Solution

6.2.6 Exporting

To facilitate the functionality of researchers being able to capture the input of synaesthetes, a 'Save Image' and pianoToImage function is created. This employs the htmlToImage third-party library and converts the piano keys to a PNG image format as it has lossless compression and supports 24-bit colour. For a more specific and data driven approach, another option is available in the form of a 'Save Data' button and a CSVLink component is used which saves a .CSV file through the use of the third-party library 'CSVLink'. The data is input under two columns: note and colour.

Chapter 7

Evaluation

The resources needed to implement the tool in its current state are minimal as it can be hosted as a static web application. There are two primary ways that the tool can be used, which will lead to the same results: By cloning the GitHub repository, the tool can be installed and run on a localhost server, provided the researcher has Node.js and npm installed on their desktop. Alternatively, any desktop which can access the web will be able to access the web application by navigating to: https://rocky-everglades-10694.herokuapp.com/.

$$\Delta E^*_{ab} = \sqrt{(L^*_2 - L^*_1) + (a^*_2 - a^*_1)^2 + (b^*_2 - b^*_1)^2} \tag{7.1}$$

In its current state, the implementation of the solution, researchers can conduct colour difference testing to achieve a 'mean' colour value for each note within a colour range provided by the subjects. This will likely be the case as 16,777,216 colour values are possible using the Material-UI-Colour component. The Delta E value of the computer monitor being presented to subjects should be taken into consideration to ensure consistency in the representation of the colours being displayed to each subject. Delta E, a standard measurement for calculating the difference between two colours on a computer screen and can therefore greatly reduce the rate of error in the results. Measured on a scale between 0 and 100, Delta E is a color difference scale that ranges from 0 to 100, with 0 indicating no color difference and 100 indicating total distortion (Backhaus et al., 2011). This is accomplished using a mathematical equation in Figure 7.1.

A value of less than one indicates that the color is not visible to the naked eye, whereas values of 2 to 10 indicate that the colours are discernible at a look. Furthermore, numbers ranging from 11 to 49 indicate that the colours are more similar and opposed, with 100 representing complete opposition. From this it can be concluded that a computer displays of a Delta E value of less than five or lower should be used when using the tool for research purposes. Furthermore, it can be said that monitors with a wider gamut will provide a richer colour to the subject as they are using the tool. This may limit the conduction of the research to an in person and controlled environment.

7.1 Assessment

Throughout the course of this project, there was a lack of collaboration with synaesthetes. This is because the project morphed throughout the course of the development process. Would need to test out the tool using actual research subjects to measure its effectiveness as a tool for measuring synaesthesia. This includes control subjects who would be required to undertake a consistency test.

From conducting research on synaesthesia, there is a clear disparity between two forms of synaesthesia studied in this project, grapheme-colour, and sound-colour synaesthesia. This proved difficult to build a foundational basis and assumptions had to be made regarding various aspects of the project, whereas this would not be the case if it were a grapheme-colour research tool. An example of this is the form-constant generation, where the shapes should ideally be evoked by the sounds and drawn in by the subject. Hence it can be said that the colours and shapes exhibited by the exposure of a musical tone to a synaesthetic subject are much more complex and particular than is displayed by the chromesthesia research tool.

The way in which p5.js was incorporated into this project proved to be somewhat time consuming and took up an extensive amount of the project timeline as the project turned into more of a case of how to make libraries compatible. Multiple libraries to support the integration of p5.js as well as alternatives to p5.js, such as Pixi.js, were iterated through before arriving at the final solution. For this reason, a more robust way of more natively incorporate third party libraries into React would greatly speed up the development process

Chapter 8

Conclusion

If this solution were to be implemented as a research tool, mobile support would have to be improved to facilitate a wider demographic. For the purposes of this research project, and in the time allocated for the project, the variable 'colour' was the only variable explored. However, given more time and more extensive research, the following variables could be explored and implemented: octave, instrument, note sequence, note length, cut-off frequency etc. could be implemented by utilising various features offered by the Tone.js library. In this way, a more robust synthesizer where the relationship between the cut-off frequency of the synthesizer and its relationship with the given form-constant would be adjustable depending on what the user saw. With further research into the psychological aspect of this research tool, the ability to map each tone combination to both a colour and a corresponding emotion could also be implemented.

Support for chords i.e., multiple tones playing at the same time and what the synaesthete experiences when it is being played is another feasible option in this case. True chromaesthesia as accounted by those who perceive it would suggest that not just one shape is discerned at a given point in time, but multiple form-constants at the same time. Therefore, future work would include support for layering of form constants on top of each other or, more succinctly, compounded form-constants. The project disregards the spatiality aspect of the chromaesthetic experience- a vital aspect that gives depth perception to the experience. Therefore, future work would include extending the tool to support virtual reality to facilitate spatiality as a variable in the research. Providing a cloud-based solution to store and view research data in different formats, such as on graphs, would alleviate much of the difficulty in interpreting the research data.

Form-constants being a non-essential element of the solution leaves room for future research where subjects would easily be able to construct or draw their own form constants and make them change dynamically depending on the tone. The tool in its current state facilitates this as researchers or subjects could write another sketch in instance mode and import it to the React project, given they include the appropriate functions in the myCustomRedrawAccordingToNewPropsHandler function. This would simplify the process of extending the research tool to other types of synaesthesia to be a one stop shop for all things synaesthesia research based.

In conclusion, the solution implemented enables the conduction of sound-colour synaesthesia consistency testing, insofar as the right research environment is provided and extra data analysis is undertaken. In the context of the field of synaesthesia, this is a novel approach as it provides a tool for colour mapping associations as synaesthetes see them. The manner in which the data is presented is grounded in pre-existing chromaesthesia research. Overall, the tool provides a means for those who experience chromaesthesia with a way of communicating the types of colours the experience in their everyday lives.

Bibliography

Backhaus, W. G., Kliegl, R., & Werner, J. S. (2011). *Color vision: Perspectives from different disciplines*. Walter de Gruyter.

Baron-Cohen, S., Wyke, M. A., & Binnie, C. (1987). Hearing words and seeing colours: An experimental investigation of a case of synaesthesia. *Perception*, *16*(6), 761–767.

Betancourt, M. (2007). A Taxonomy of Abstract Form Using Studies of Synesthesia and Hallucinations. *Leonardo*, *40*(1), 59–65. https://doi.org/10.1162/leon.2007.40.1.59

Boden, M. A., & Edmonds, E. A. (2009). What is generative art? [Publisher: Taylor & Francis]. *Digital Creativity*, *20*(1-2), 21–46.

Brang, D., Teuscher, U., Miller, L. E., Ramachandran, V. S., & Coulson, S. (2011). Handedness and calendar orientations in time–space synaesthesia [Publisher: Wiley Online Library]. *Journal of Neuropsychology*, *5*(2), 323–332.

Bressloff, P. C., Cowan, J. D., Golubitsky, M., Thomas, P. J., & Wiener, M. C. (2002). What geometric visual hallucinations tell us about the visual cortex. *Neural computation*, *14*(3), 473–491.

Brougher, K., Strick, J., Wiseman, A., & Zilczer, J. (2005). *Visual music: Synaesthesia in art and music since 1900: The museum of contemporary art, los angeles,[13 february-22 may 2005]: Hirshhorn museum and sculpture garden, smithsonian institution, washngton dc,[23 june-11 september 2005]*. Thames & Hudson.

Cavett, E. (2021). *Perlinpaint*. https://editor.p5js.org/EdCavett/sketches/DRvCPgi-

cdaein. (2021). *Trippy pattern*. https://editor.p5js.org/cdaein/sketches/BkxxNlniW

Cytowic, R. E. (1989). *Synesthesia: A Union of the Senses*. Springer-Verlag. https://doi.org/10.1007/978-1-4612-3542-2

Cytowic, R. E. (2018). *Synesthesia*. The MIT Press.

de Mendoza, F. S. (1899). *L'audition colorée: tude sur les fausses sensations secondaires physiologiques et particulièrement sur les pseudo-sensations de couleurs associées aux perceptions objectives des sons*. Doin.

Eagleman, D. M., Kagan, A. D., Nelson, S. S., Sagaram, D., & Sarma, A. K. (2007). A standardized test battery for the study of synesthesia [Publisher: Elsevier]. *Journal of neuroscience methods*, *159*(1), 139–145.

Fedosejev, A. (2015). *React. js essentials*. Packt Publishing Ltd.

Galton, F. (1881). Visualised numerals. [Publisher: JSTOR]. *The Journal of the Anthropological Institute of Great Britain and Ireland*, *10*, 85–102.

Greenberg, I. (2007). *Processing: Creative coding and computational art*. Apress.

Herder, J. G., & Irmscher, H. D. (1966). *Abhandlung über den Ursprung der Sprache*. Reclam Stuttgart.

Hubbard, E. M., & Ramachandran, V. S. (2005). Neurocognitive mechanisms of synesthesia [Publisher: Elsevier]. *Neuron*, *48*(3), 509–520.

Jewanski, J., Simner, J., Day, S. A., Rothen, N., & Ward, J. (2020). The evolution of the concept of synesthesia in the nineteenth century as revealed through the history of its name [Publisher: Routledge _eprint: https://doi.org/10.1080/0964704X.2019.1675422]. *Journal of the History of the Neurosciences*, *29*(3), 259–285. https://doi.org/10.1080/0964704X.2019.1675422

Jones, C., Carey, J., & Richardson, E. (2012). The saint-aubin livre de caricatures: Drawing satire in eighteenth-century paris. *Studies on Voltaire and the Eighteenth Century*, (6).

Krohn, W. O. (1892). Pseudo-chromesthesia, or the association of colors with words, letters and sounds [Publisher: JSTOR]. *The American Journal of Psychology*, *5*(1), 20–41.

Laeng, B., Svartdal, F., & Oelmann, H. (2004). Does color synesthesia pose a paradox for early-selection theories of attention? [Publisher: SAGE Publications Sage CA: Los Angeles, CA]. *Psychological Science*, *15*(4), 277–281.

Lewis-Williams, J. D. (2004). Neuropsychology and upper palaeolithic art: Observations on the progress of altered states of consciousness. *Cambridge Archaeological Journal*, *14*(1), 107–111.

Marks, L. E. (2014). *The unity of the senses: Interrelations among the modalities*. Academic Press.

Moussa, A. (2021). *Chromatic wheel copy copy*. https://editor.p5js.org/AhmadMoussa/sketches/TxnpilImt

Price, M. C., & Mentzoni, R. A. (2008). Where is January? The month-SNARC effect in sequence-form synaesthetes [Publisher: Elsevier]. *Cortex*, *44*(7), 890–907.

Robinson, M. (2013). *Getting started with juce*. Packt Publishing Ltd.

Steen, C. (2019). Two kinds of visions, synesthesia and hypnagogia: A comparison. *Iris 39/2019 Synesthésies visuelles*.

synesthesia.com. (2021). *Are you a synesthete? 7 synesthesia tests to find it out*. Retrieved September 28, 2021, from https://synesthesia.com/blog/synesthesia-tests/

syntoolkit.org. (2021). *Multisense research toolkit*. Retrieved September 28, 2021, from https://www.syntoolkit.org/

Van Campen, C. (2010). *The hidden sense: Synesthesia in art and science*. Mit Press.

Vulpian, A. (1866). *Leçons sur la physiologie générale et comparée du système nerveux: Faites au Muséum d'histoire naturelle*. Germer Baillière.

Appendix A

Appendix

This appendix contains selected code listings.

A.1

Chromesthesia.js

```
import React, { Component } from 'react';
import CssBaseline from '@material-ui/core/CssBaseline';
import Paper from '@material-ui/core/Paper';
import Grid from '@material-ui/core/Grid';
import Button from '@material-ui/core/Button';
import { Typography } from '@material-ui/core';
import { withStyles } from '@material-ui/core/styles';

// P5
import P5Wrapper from 'react-p5-wrapper';
import sketch from "./sketch";
import sketch2 from "./sketch2";
import sketch3 from "./sketch3";

// Tone.js
import * as Tone from "tone";

// Colour Picker
import { ColorPicker } from 'material-ui-color';
import "./../style.css";

// HTML to Image
import * as htmlToImage from 'html-to-image';
import download from 'downloadjs';
```

```
// Misc
import stringifyObject from 'stringify-object';
import { CSVLink } from 'react-csv';

//////////////////////////////////////////////////

const synth = new Tone.PolySynth().toDestination();

const styles = theme => ({
  root: {
    height: '100vh',
  },
  paper: {
    margin: theme.spacing(6, 6),
    display: 'flex',
    flexDirection: 'column',
    alignItems: 'center',
  },
  color: {
    width: '36px',
    height: '14px',
    borderRadius: '2px',
    background: 'rgb(0,0,0)',
  },
});

class Chromesthesia extends Component {
  constructor() {
    super();
    this.wrapper = React.createRef();

    // Props to send to P5 Sketch
    this.state = {
      fill: '#D82B2E',
      sketch: sketch,

      // Default Colour Values
      c : '#D82B2E',
      db: '#DB6335',
      d : '#E78D37',
      eb: '#F2CB33',
```

```
    e : '#F6E95F',
    f : '#C0D65C',
    gb: '#009f47',
    g : '#009890',
    ab: '#2a327f',
    a : '#a892a4',
    bb: '#c42c82',
    b : '#a12c56'
  };
  // Key bind handler
  this.handleKey = this.handleKey.bind(this);
}

playNote(note) {
  synth.triggerAttackRelease(`${note}`, "8n");
  var colourForNotes = {
    'C4' :  this.state.c,
    'C#4':  this.state.db,
    'D4' :  this.state.d,
    'D#4':  this.state.eb,
    'E4' :  this.state.e,
    'F4' :  this.state.f,
    'F#4':  this.state.gb,
    'G4' :  this.state.g,
    'G#4':  this.state.ab,
    'A4' :  this.state.a,
    'A#4':  this.state.bb,
    'B4' :  this.state.b,
  };
  this.setState({fill: colourForNotes[note]});
}

// Binding keyboard keys to Tone.js
handleKey = (e) => {
  if(e.keyCode === 65) {
    this.playNote("C4");
  } else if (e.keyCode === 87 ) {
    this.playNote("C#4");
  } else if (e.keyCode === 83 ) {
    this.playNote("D4");
  } else if (e.keyCode === 69 ) {
    this.playNote("D#4");
```

```
  } else if (e.keyCode === 68 ) {
    this.playNote("E4");
  } else if (e.keyCode === 70 ) {
    this.playNote("F4");
  } else if (e.keyCode === 84 ) {
    this.playNote("F#4");
  } else if (e.keyCode === 71 ) {
    this.playNote("G4");
  } else if (e.keyCode === 89 ) {
    this.playNote("G#4");
  } else if (e.keyCode === 72 ) {
    this.playNote("A4");
  } else if (e.keyCode === 85 ) {
    this.playNote("A#4");
  } else if (e.keyCode === 74 ) {
    this.playNote("B4");
  }
}

componentDidMount() {
  document.addEventListener('keydown', this.handleKey);
}

componentWillUnmount() {
  document.removeEventListener('keydown', this.handleKey);
}

pianoToImage() {
  var domElement = document.getElementById('capture');
  htmlToImage.toPng(domElement)

    .then(function (dataUrl) {
      console.log(dataUrl);
      download(dataUrl, 'chromesthesia-keyboard.png');
    })
    .catch(function (error) {
      console.error('oops, something went wrong!', error);
    });
}

handleChange = (value, key) => {
  // stringify-object to remove double quotes
```

```
    var hex = stringifyObject(value.hex, {
      doubleQuotes: false,
    });
    hex = hex.replace(/\'/gi,'');

    if(key === "C") {
      this.setState({c: '#' + [hex]});
    } else if (key === "C#") {
      this.setState({db: '#' + [hex]});
    } else if (key === "D") {
      this.setState({d: '#' + [hex]});
    } else if (key === "D#") {
      this.setState({eb: '#' + [hex]});
    } else if (key === "E") {
      this.setState({e: '#' + [hex]});
    } else if (key === "F") {
      this.setState({f: '#' + [hex]});
    } else if (key === "F#") {
      this.setState({gb: '#' + [hex]});
    } else if (key === "G") {
      this.setState({g: '#' + [hex]});
    } else if (key === "G#") {
      this.setState({ab: '#' + [hex]});
    } else if (key === "A") {
      this.setState({a: '#' + [hex]});
    } else if (key === "A#") {
      this.setState({bb: '#' + [hex]});
    } else if (key === "B") {
      this.setState({b: '#' + [hex]});
    }
  };

  render() {
    const { classes } = this.props;
    var data = [
      ["Note", "Colour"],
      ["C4", this.state.c],
      ['C#4', this.state.db],
      ['D4' , this.state.d],
      ['D#4', this.state.eb],
      ['E4' , this.state.e,],
      ['F4' , this.state.f],
```

```
      ['F#4', this.state.gb],
      ['G4' , this.state.g],
      ['G#4', this.state.ab],
      ['A4' , this.state.a,],
      ['A#4', this.state.bb],
      ['B4' , this.state.b]
    ];

    return (
    <Grid container component="main" ref={this.wrapper} >
      <CssBaseline />
      <Grid item xs={12} sm={12} md={12} component={Paper} elevation={6} square>
        <div className={classes.paper}>
          <P5Wrapper sketch={this.state.sketch} fill={this.state.fill} />
          <Grid item lg={8} md={8} xs={8} sm={8} mx="auto">

          <Typography variant="overline" display="block">Sketches</Typography>
          <Button size="small" variant="contained" onClick={() => this.setState({sketch:
sketch})}> 1</Button>
          <Button size="small" variant="contained" onClick={() => this.setState({sketch:
sketch2})}> 2</Button>
          <Button size="small" variant="contained" onClick={() => this.setState({sketch:
sketch3})}> 3</Button>

          <Button variant="outlined" size="small" style={{marginLeft: '2em'}} onClick={this.
            Save Image
          </Button>
          <Button variant="outlined" size="small" onClick={this.pianoToCsv}>
            <CSVLink
              data={data}
              filename={"my-file.csv"}
              target="_blank"
            >
              Save Data
            </CSVLink>
          </Button>
          </Grid>
        </div>
      </Grid>
      <Grid item xs={12} sm={12} md={12}>
      <div className="note-wrapper" id='note-wrapper'>
        <div id="capture">
```

```
<button className="note white c1" onClick={() => this.playNote("C4")}>
  <div className="picker">
    <ColorPicker
      value={this.state.c}
      onChange={(e) => this.handleChange(e, "C")}
      disableTextfield
      disableAlpha
    />
  </div>
</button>
<button className="note black" onClick={() => this.playNote("C#4")}>
  <div className="picker-black">
    <ColorPicker
      value={this.state.db}
      onChange={(e) => this.handleChange(e, "C#")}
      disableTextfield
      disableAlpha
    />
    </div>
</button>

<button className="note white d" onClick={() => this.playNote("D4")}>
  <div className="picker">
    <ColorPicker
      value={this.state.d}
      onChange={(e) => this.handleChange(e, "D")}
      disableTextfield
      disableAlpha
    />
  </div>
</button>

<button className="note black" onClick={() => this.playNote("D#4")}>
<div className="picker-black">
    <ColorPicker
      value={this.state.eb}
      onChange={(e) => this.handleChange(e, "D#")}
      disableTextfield
      disableAlpha
    />
  </div>
</button>
```

```
<button className="note white e" onClick={() => this.playNote("E4")}>
<div className="picker">
    <ColorPicker
      value={this.state.e}
      onChange={(e) => this.handleChange(e, "E")}
      disableTextfield
      disableAlpha
    />
  </div>
</button>

<button className="note white f" onClick={() => this.playNote("F4")}>
<div className="picker">
    <ColorPicker
      value={this.state.f}
      onChange={(e) => this.handleChange(e, "F")}
      disableTextfield
      disableAlpha
    />
  </div>
</button>

<button className="note black" onClick={() => this.playNote("F#4")}>
  <div className="picker-black">
    <ColorPicker
      value={this.state.gb}
      onChange={(e) => this.handleChange(e, "F#")}
      disableTextfield
      disableAlpha
    />
  </div>
</button>

<button className="note white g" onClick={() => this.playNote("G4")}>
  <div className="picker">
      <ColorPicker
        value={this.state.g}
        onChange={(e) => this.handleChange(e, "G")}
        disableTextfield
        disableAlpha
      />
```

```
  </div>
</button>

<button className="note black" onClick={() => this.playNote("G#4")}>
  <div className="picker-black">
    <ColorPicker
      value={this.state.ab}
      onChange={(e) => this.handleChange(e, "G#")}
      disableTextfield
      disableAlpha
    />
  </div>
</button>

<button className="note white a" onClick={() => this.playNote("A4")}>
  <div className="picker">
    <ColorPicker
      value={this.state.a}
      onChange={(e) => this.handleChange(e, "A")}
      disableTextfield
      disableAlpha
    />
  </div>
</button>

<button className="note black" onClick={() => this.playNote("A#4")}>
  <div className="picker-black">
    <ColorPicker
      value={this.state.bb}
      onChange={(e) => this.handleChange(e, "A#")}
      disableTextfield
      disableAlpha
    />
  </div>
</button>

<button className="note white b" onClick={() => this.playNote("B4")}>
  <div className="picker">
    <ColorPicker
      value={this.state.b}
      onChange={(e) => this.handleChange(e, "B")}
      disableTextfield
```

```
                disableAlpha
              />
            </div>
          </button>
        </div>
        </div>
      </Grid>
    </Grid>
    );
  }
}

export default withStyles(styles)(Chromesthesia);
```

A.2 sketch.js

```
import "./../style.css";

export default function sketch (p) {
let canvas;

p.setup = () => {
p.remove();
p.loop();
canvas = p.createCanvas(p.windowWidth - 500, p.windowHeight - 350);
p.noStroke();
p.frameRate(60);
p.colorMode(p.HSB);
p.textSize(10);
};

p.windowResized = () => {
p.resizeCanvas(p.windowWidth -500, p.windowHeight - 350);
};

p.draw = () => {
var sc = 128 + p.sin(p.frameCount/30) * 128;
var x = p.width/2 + p.cos(p.frameCount/30) * 120;
var y = p.height/2 + p.sin(p.frameCount/30) * 120;
```

```
p.rect(0,0, p.width, p.height);
p.stroke(sc, 100);

p.push();
for (var i = 0; i < p.width; i += 10) {
p.line(i, 0, x, y);
}
for (i = 0; i < p.width; i += 10) {
p.line(i, p.height, x, y);
}
for (i = 0; i < p.height; i += 10) {
p.line(0, i, x, y);
}
for (i = 0; i <= p.height; i += 10) {
p.line(p.width, i, x, y);
}
p.pop();
};

p.myCustomRedrawAccordingToNewPropsHandler = (newProps) => {
//Make sure the canvas has been created
if (canvas) {
p.fill(newProps.p5Colour);
}
};
}
```

A.3 sketch2.js

```
import "./../style.css";

export default function sketch2 (p) {
let canvas;

let numDashers = 400;
let dashers = [];
let pts = [];

let n;
let rate;
```

```
let invrt;
let bins;
let offstsIndex;
let offsetPercentage;
let distC;


let a;
let x;
let y;
let z;
let d;
let sw;

p.setup = () => {
p.remove();
p.loop();
canvas = p.createCanvas(p.windowWidth-500, p.windowHeight - 350);
p.frameRate(60);
p.colorMode(p.HSB);

p.stroke(0);

for (n = 0; n < numDashers; n++) {
let ang = p.map(n, 0, numDashers, 0, p.TAU);
let angX = 1200/2 + 50 * p.cos(ang);
let angY = 400/4 * 3  + 50 * p.sin(ang);

let offsts = p.map(n, 0, numDashers, -100, 100);
let randomness = p.random(-18, 18);
if(n < numDashers / 4){
  offsts= -120 + randomness;
  rate = 0.01;
  invrt = 1;
} else if(n < numDashers / 2){
  offsts = -45 + randomness;
  rate = 0.015;
  invrt = 0;
}else if(n < numDashers / 4 * 3){
  offsts = 45 + randomness;
  rate = 0.02;
  invrt = 1;
```

```
}else{
  offsts = 120 + randomness;
  rate = 0.025;
  invrt = 0;
}

bins = [-170, -110, -45, 34, 110];
offstsIndex = p.floor(p.map(n, 0, numDashers, 0, 5));
pts.push([angX, angY]);
dashers.push(new makeDasher(1350 / 2, 600 / 2, rate, 200, bins[offstsIndex] + randomness,
p.PI-1 + 0.57 * p.cos(ang),
p.TAU + 1-0.57 * p.cos(ang), invrt));
}
};
function makeDasher(cx,cy,rate,rad,offst,bg,eg, invert) {
this.invert = invert;

this.cx = cx;
this.cy = cy;

this.cp = p.random(p.TAU);
this.rate = rate;

this.rad = rad;
this.offset = offst;

this.segLength = p.random(0.01, 0.2);

this.beginAngle = bg;
this.endAngle = eg;


if(this.invert){
  this.beginAngle = eg;
  this.endAngle = bg;
}
this.midWayPoint = (this.beginAngle + this.endAngle)/2;

this.minD = p.dist(this.rad * p.cos(this.beginAngle), this.rad * p.sin(this.beginAngle), 0,
this.maxD = p.dist(this.rad * p.cos(this.midWayPoint), this.rad * p.sin(this.midWayPoint), 0
```

```
this.move = function () {
p.push();
p.translate(this.cx, this.cy);

  this.cp += this.rate;
  if (this.cp > p.TAU) {
this.cp = 0;
  }

  a = p.map(this.cp, 0, p.TAU, this.beginAngle, this.endAngle);
  x = this.rad * p.cos(a);
  y = this.rad * p.sin(a);

  d = p.dist(x, y, 0, 0 + this.rad);


  sw = p.map(d, this.minD, this.maxD, 0, 8);
  p.strokeWeight(sw);

  offsetPercentage = p.map(d, this.minD, this.maxD, 0, 1);

  x = (this.rad * 2 + this.offset) * p.cos(a);
  y = (this.rad * 2 + this.offset) * p.sin(a);

  distC = p.dist(x, y, 0, 0) / 50 + a - this.cp * 3 + p.frameCount / 25;

  p.noFill();

  p.arc(
0,
0,
this.rad * 2 + this.offset + 10 * p.sin(x / 20 + p.frameCount/25),
this.rad * 2 + this.offset + 10 * p.sin(x / 20 + p.frameCount/25),
a - this.segLength * offsetPercentage,
a
  );
  p.pop();
};
  }

p.windowResized = () => {
p.resizeCanvas(p.windowWidth -500, p.windowHeight - 350);
```

```
};

p.draw = () => {
p.blendMode(p.BLEND);
p.background(255, 20);


for (n=0; n < numDashers; n++) {
dashers[n].move();
}

};


p.myCustomRedrawAccordingToNewPropsHandler = (newProps) => {
//Make sure the canvas has been created
if (canvas) {
p.stroke(newProps.p5Colour);

}
};
}
```

A.4 sketch3.js

```
import "./../style.css";

export default function sketch3 (p) {
let canvas;

    let rrS = 0;
    let bbS = 0;
    let xnoiseMax = 1.5;
    let ynoiseMax = 1.5;
    let xrot = 0.5;
    let yrot = 0.5;
    let transX = 0;
    let transY = 0;
    let walkMin = -2;
    let walkMax = 2;
    let footStepX = 0;
```

```
    let footStepY = 0;
    let outside = false;

p.setup = () => {
p.remove();
p.loop();
canvas = p.createCanvas(p.windowWidth - 500, p.windowHeight - 350);
p.frameRate(40);
p.colorMode(p.HSB);
};

p.windowResized = () => {
p.resizeCanvas(p.windowWidth -500, p.windowHeight - 350);
};

p.draw = () => {
        rrS = p.map(p.width, 0, p.windowWidth, 0, 255);
        bbS = p.random(0,100);
        footStepX = footStepX + p.random(walkMin,walkMax);
        footStepY = footStepY + p.random(walkMin,walkMax);

        outside = footStepX >= p.width || footStepX <= -p.width || footStepY >= p.height ||

        if (outside) {
            footStepX = 0;
            footStepY = 0;
        }
        transX = p.width/2 + footStepX;
        transY = p.height/2 + footStepY;

        p.translate(transX,transY);
        p.stroke(rrS, 0, bbS, 50 + footStepY / footStepX);
        p.strokeWeight(0.5);
        p.background(255, 0.1);

        p.beginShape();
            for (let a = 0; a < p.TWO_PI; a+=0.2) {
            let xoff = p.map(p.cos(a+xrot)+ 1, -1, 1, 0, xnoiseMax);
            let yoff = p.map(p.sin(a+yrot)+ 1, -1, 1, 0, ynoiseMax);
            let r = p.map(p.noise(xoff,yoff), 0, 1, p.random(10, 50), p.random (100,400));
            let x = r * p.cos(a);
            let y = r * p.sin(a);
```

```
            p.vertex(x,y);
            }
        p.endShape(p.CLOSE);
        xrot += 0.01;
        yrot += 0.01;

};

p.myCustomRedrawAccordingToNewPropsHandler = (newProps) => {
if (canvas) {
p.fill(newProps.p5Colour);
}
};
}
```