PixelCato - Deriving Music from Art and Colour

Final Report for CS39440 Major Project

Author: Cai Rhys Jones (crj10@aber.ac.uk)
Supervisor: Prof. Reyer Zwiggelaar (rrz@aber.ac.uk)

3rd May 2017
Version: 1.0 (Release)

This report was submitted as partial fulfilment of a BEng degree in Software Engineering (inc Integrated Industrial and Professional Training) (G600)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name: Cai Rhys Jones

Date: 08/05/17

Consent to share this work

By including my name below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name: Cai Rhys Jones

Date: 08/05/17

Acknowledgements

I would like to thank the following people for their help and support during this project:

My supervisor Prof. Reyer Zwiggelaar for his invaluable guidance, advice and his organisation of weekly meetings which were helpful to reflect on current progress and discuss challenges I was facing.

My loving family, most notably my parents Andrea, Martyn, and my Grandmother Thelma, who have always supported me in my endeavours and have never stopped encouraging or believing in me.

My wonderful friends, Rob, Leon, Ben, Tom and Lewis, who have consistently managed to get me into and out of tight spots many times in life, I wouldn't change them for anything.

Finally to two exceptional teachers I've had in life:

Carol Mair Macy one of the first teachers I had who encouraged my writing and computing abilities at a young age, giving me the confidence to continue pursuing a career in the computing industry.

Elizabeth Roberts my first music teacher who started me off down the path of becoming a musician, she often told me to turn the guitar amplifier down so that I wouldn't develop Tinnitus, she truly helped me grow as both a musician and a person.

Abstract

Music and Art have been closely linked throughout history, there has often been the attempt to visually link music with colours and light.

In the last few decades the composition and creation of music has become accessible to those who have never received musical training through the use of technology such as Digital Audio Work Stations and MIDI devices.

There is a challenge in what is called Algorithmic Composition, which is the composition of melodies, chords and structure through the use of computer systems without the use of human intervention

In this project I will create software that will attempt to generate melodies, derived from images, art and photographs, to explore the link between music and colours.

Glossary of Common Musical Terms Used

- Atonal

  Without key.

- Chord

  A collection of three or more notes played together in unison.

- Chromatic

  Use of all 12 notes.

- Harmony

  The sound of notes played together in a way that sounds pleasant.

- Interval

  The distance in pitch between two notes.

- Key

  System of pitches based around the key note.

- Melody

  A sequence of single notes that are musical.

- Motif

  A recurring theme in a composition.

- Note

  A musical pitch.

- Pizzicato

  A technique in which a bowed instrument such as a violin is plucked.

- Root

  The start of a scale or the pitch of a key.

- Scale

  Notes of a key that ascend or descend.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Background

1.1 Introduction

Music is one of the most influential mediums of art, it is often characterised by its fundamental elements:

- Pitch
- Timbre
- Texture
- Dynamics
- Duration
- Tempo

It is able to evoke a range of emotions, making you feel happy, humorous, apathetic, or depressed, it can give you the urge to dance, send tingles through your body, make you laugh and make you cry, there is psychology behind what is known as musical expectation, certain musical queues can identify Tragedy, Suspense, Surprise and other such emotional responses [2].

Over the past 10-20 years there has been a vast evolution in the use of music in technology, software in the form of digital audio workstations have become the standard for creating professional quality music ranging from amateur artists using a laptop in a bedroom to professional sound producers in a recording studio. The creation of MIDI (Musical Instrument Digital Interface) introduced the ability for multiple instruments to be played from a single controller by producing events to describe the pitch, velocity and notes played. The MIDI standard is useful to musicians as it allows them to directly edit elements such as notes, timing and instrumentation with a level of relative ease, it is a compact format allowing a song to be coded in a small amount of information resulting in a small file size in the kilobyte regions [1]. Throughout history, there

has been research into the relation of colours to musical notes, Isaac Newton wrote about this in the book *Opticks: Or, a Treatise of the Reflexions, Refractions, Inflexions and Colours of Light.* At a high level, he took a prism and shined light through it, revealing the seven colours associated with a rainbow, from this he derived a musical colour wheel based on the Dorian musical mode [3].

1.2 Aim

The aim of the project is to produce a software solution titled ”PixelCato” that will take images, photographs or artistic pieces such as paintings and compose a melody based on the principle colours featured in the image using Newtonian colour theory.

1.3 Objectives

- Produce a solution that can support multiple image types.
- Produce an Algorithmic composition of a simple melody.
- Playback functionality of the generated melody.
- Allow the exportation of melodies, for editing and playback in external applications.

There was thought put into the computational composition aspect, the composition of chords to accompany the melody was considered but ultimately determined to be a time consuming process for the final objective.

1.4 Deliverables

Deliverables for this project include:

- The source code of the application ”PixelCato”, that demonstrates the ability to take images and produce melodies from the colour composition.
- This project report that discusses the solution, methodology and provides a critical evaluation.

1.5 Methodology

The methodology used in this project has been predominantly Scrum with some aspects of Extreme Programming (XP). The approach was iterative meaning the project went through multiple sprints each improving upon previous work and incrementally adding further functionality through the development process. XP processes used include simple design, where possible functionality has been developed with simplicity in mind, this is apparent in the minimalistic aesthetic of the interface. The use of regular Unit Testing has ensured consistent regression, quality and feedback from the system, which results in a robust solution.

1.5.1 Version Control

The project uses version control for the software development, in addition to this the report used a cloud backup solution with the web based LaTeX editor "Overleaf" [4]. The software version control method was Git, using a private GitHub repository created for use in this project, this allows for regular backups and code management, which can aid in a scenario where the new code implementation breaks the solution. The code is hosted externally on GitHub so in the event of local storage failure the code would be recoverable.

1.6 Degree Relevance

The project will utilise skills, techniques and standards acquired from modules undertaken as part of my degree such as CS12420 Software Development, CS21120 Program Design, Data Structures And Algorithms and CS22120 The Software Development Life-cycle. I will use experience gained from my industrial year in agile SCRUM methodology to ensure that best practices are applied during my iterative development cycles. The degree has prepared me to demonstrate my problem solving ability and will allow for the success of the project.

Chapter 2

Research

2.1 Introduction

This chapter discusses an overview of the problem, relevant research into subjects prior to development with a look at similar systems and an evaluation of potential appropriate technologies for the project.

2.2 Problem Overview

Music is a linear medium of art, it has a start, a middle and an end, when you listen to a piece of music there is a natural progression through these states, when you look at an image or a piece of a art such as a sculpture you process all the information at once, it's a static experience. how do you take the static experience of an image, and create something linear like music? Music has features like tempo, rhythm and harmony, that are not present in static art, what processes can we use to derive?

In an image you can get a sense of the image based on the most prominent colours in the palette, if there is a method of taking these principle colours and relating them to sound or music, then the static experience of an image can be derived and become linear.

There is another problem with this approach however, if the image is devoid of colour such as a greyscale image, how would you derive music from the colour?. The background of colour theory and the relation to music was explored to better understand the nature of composition by colour, subjects such as serialism are researched in order to determine a method of composing when devoid of colour.

Tonality must also be considered, if the colour is used to determine a musical note how do you ensure that the next note will remain in the same key?, in order to create harmony the notes must be derived from scale intervals and how do you relate the colours together to form this harmony.

2.3 Music and Colour Theory

Sound and Light are similar in that they both travel in a wave, however unlike light, sound travels slower and does not have the same electrical or magnetic fields [5]. The most common frequency range of sound is between 20Hz and 20000Hz and the pitch frequency for the musical note A located above the Middle C as standardised by the ISO(International Standards Organisation) is 440Hz [6]. Sound and light differ in another way which is interesting, if you take the note A at 440Hz and play another note of similar frequency slightly increasing or decreasing around 440Hz you get dissonance and the note sounds strange or unpleasant, however with light the eye would notice these subtle changes such as a red deepening or lightening and it would not have the same dissonant effect . There are some note frequencies that do not produce this dissonance and this forms the basis of musical harmony [5].

There have been previous attempts to correlate a relationship between Colour and Music. The famous Pythagoras believed that the planets produced sound as they moved much like strings produce sound when they vibrate, this theory become known as the Music of the Spheres, the theory describes the distance between the planets as tones, with 8 steps to the ”highest skies”(summum caelum).Plato along with Aristotle built on this idea, with Aristotle creating his own colour spectrum with black and white at either end, he then assigned musical notes to the spectrum according to how the tones blended together, this was considered to be purely speculative with very little evidence of an actual relationship. [5]

2.3.1 Newtonian Colour Theory

Sir Isaac Newton proved that light was made of particles refraction by shining light through two prisms, when light passed through the first prism it was split into a plethora of colours, Newton decided to divide the rainbow into the distinct colours, In his publication *Opticks* Newton suggests that the number seven was chosen based on the ancient greek belief that seven was a mystical number, he then assigned tonalities to a colour wheel based on the Dorian musical mode (all the white notes on a piano starting on the note ”D”) [3].

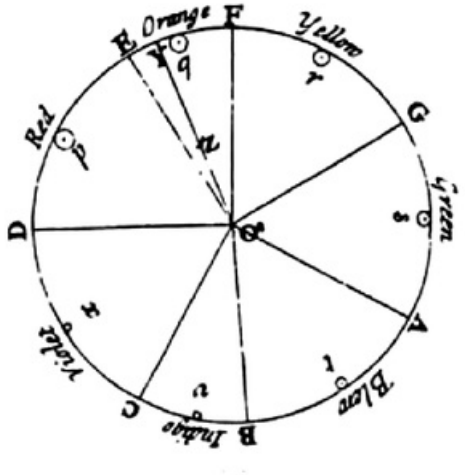

Figure 2.1: Newton Colour Wheel, taken from Opticks: Or, a Treatise of the Reflexions, Refractions, Inflexions and Colours of Light. sourced from Wikimedia Commons.

2.3.2 Melody Composition

In western music, there are seven notes in a diatonic scale, this is the base for most contemporary music with pieces generally falling into two categories Major or Minor. The scales are constructed with degree intervals starting with the root of a scale and ascending relative to the root note, chords are built out of playing different scale degrees together, a major chord is created when you take the root, third and fifth degrees of the scale and play them together, this forms the basis of most chord structures, with a minor chord following a similar pattern with a root, flattened third and fifth structure instead. A melody is composed from the arrangement of notes in a scale played at varying speeds, and building tension by manipulating the relationship between certain notes like the root and the fifth degree which is referred to as the dominant note. Melodies are described by what is referred to as melodic motion, this means that the intervals between the pitches played create tension, cadence, continuity and release.

2.3.3 Impressionism and Serialism

in the early 20th century, there was a western movement amongst composers who wanted to focus their efforts in conveying emotion rather than tonal picture with nice perfect harmony that they had come to know and accept, this was known as Impressionism. One such composer was Alexander Scriabin, Scriabin was influenced by the condition synaesthesia where people experience sensory input with visual effects, which meant that some people could see colours when hearing music [7], his pieces could be described as atonal which means without a particular musical key or tonal centre. Scriabin was inspired by Isaac Newton's work in Opticks, for his symphony titled *"Prometheus: The Poem of Fire"* He created a colour-coded circle of fifths wheel and created the instrument known as the "clavier a lumieres" which was an organ that projected coloured light on screen in a concert hall [8]. Serialism was another western movement most known for it's use of a 12 tone row in which the music was composed and structured using an arrangement of the

12 chromatic (black and white keys on a piano) notes, this technique was primarily developed by Arnold Schoenberg [9].

These movements are similar in that they both promote the use of chromaticism and are designed to ignore the traditional harmony and structure of western classical composition [9]

2.4 Similar Systems

2.4.1 RGB Music Lab

RGB Music Lab is a piece of software written by Kenji Kojima, that takes images, photos and 3D visuals and converts them into music by looking at the intensity of the pixel RGB(Red Green Blue pixel colour format) values and creating harmony based on three different values. the software composes the score from the image directly using computational composition it is meant to mimic the audio and visual link sometimes known as synaesthesia [10].

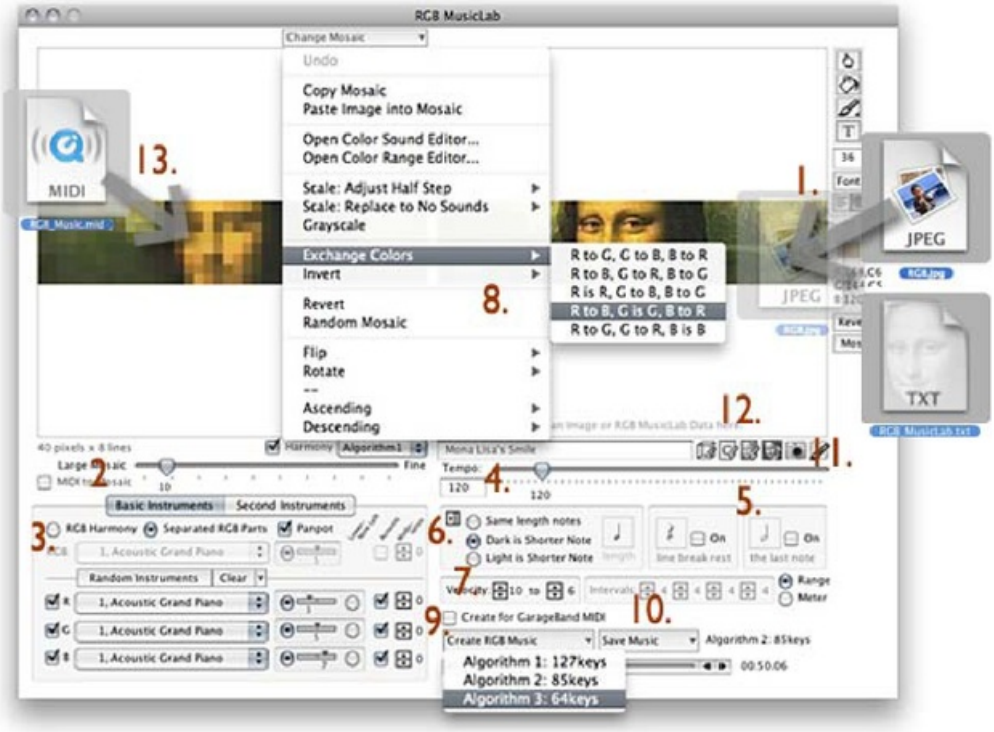

Figure 2.2: RGB Music Lab User Interface sourced from the author's website

2.4.2 Photosounder

Photosounder is an audio editor/synthesiser which is designed to create sounds from images, it is described on its website as "the ultimate bridge between the graphical world and the audio world, bringing the full power of image editing to the service of creating and transforming sounds." [11]. Its main features are the ability to derive sounds from fractals, photographs and images and provide powerful image manipulation tools, it is a commercial product being sold for $79 on the website. From the examples shown on their website it is clear the emphasis is on hearing an image as opposed to creating music with it however there are some examples of creating melodies drawn in Adobe Photoshop and playing them back. [11]

Figure 2.3: Photosounder screenshot sourced from the author's website

2.5 Computational Composition

The process of algorithmically composing music is a difficult task, and the fundamental musical elements need to be considered. Previous software applications have had success in this field most notably the Iamus system which utilises a computational system known as Melomics, this system creates pieces of contemporary classical music that have even been performed by the London Symphony Orchestra [12].

Melody composition looks at the following main musical elements.

2.5.1 Pitch

In composition pitch refers to the frequency of a musical note, knowing how to assign particular notes in a melody to sound pleasant or portray a particular feeling is a talent and skill in itself. Determining pitch is one of the first factors to consider when composing a melody. Traditionally a composer would need to consider the tonal range of the instruments they are working with, particularly in the case of composing for a vocalist, all of whom have various ranges and limits. Different instruments also have different ranges for example a 24 fret guitar covers 5 octaves, in contrast a violin has a limit of four and a half, although some of this is determined by instrumentalist ability.

2.5.2 Timbre

Timbre is also known as tonal colour, it is the physical characteristics of the sound used distinguishes different instrumentation, for example if you play middle C on a guitar and a piano at the same time, the notes will resonate and sound like the same pitch, however their makeup will be different and produce very distinct sounds.

For composition timbre is important when choosing instruments, it will allow you to make appropriate choices for Melody, Accompaniment and Percussion. Playing techniques should also be considered, for example different bassists can produce different distinct sounds that derive from their own playing techniques such as slapped, fingered, or plucked bass.

2.5.3 Rhythm

Rhythm in music refers to the timing of the notes within the timing, the most common Time Signature is 4/4, which means that there are 4 quarter notes or beats in a bar of music. Rhythm is important for composition as it controls the flow of the music.

2.5.4 Dynamics

Dynamics in music refers to how loud or quietly a note should be played, in composition different sections or passages could have different dynamics, a piece may start off with a very soft quiet instrumental intro noted as *pianissimo* which would then erupt in a crescendo (to go from quiet to loud) to a very loud section notated as fortissimo , dynamics are important in creating tonal variety in a piece, and are also used in scoring for film, theatre and screen to match the content of what is being shown at the time.

2.6 Relevant Technology

2.6.1 MIDI

MIDI is a technology standard that aids musicians and amateurs in the creation of music, many different devices, operating systems, digital instruments use and support the format, it works in a similar style to sheet music, there are messages that describe what note is played and how to play it. MIDI does not provide audio, much like looking at sheet music, it is however extremely flexible, it can contain performance instructions such as how hard to play the note (referred to as Velocity), length of note, and has features such as pitch bending to imitate effects like tremolo. [13]

There are three main parts to the MIDI standard:

- Messages
- Transport(Connection)
- The File Format

2.6.1.1 Messages

The MIDI Protocol is made up of different messages that describe the composition, there are velocity messages which tell the device how loud or how hard to play the note, there are instrument messages which can control which type of instrument plays the notes, there are note messages that tell the device to play or stop playing that particular note and length messages which describe the length and brightness of a note. [13]

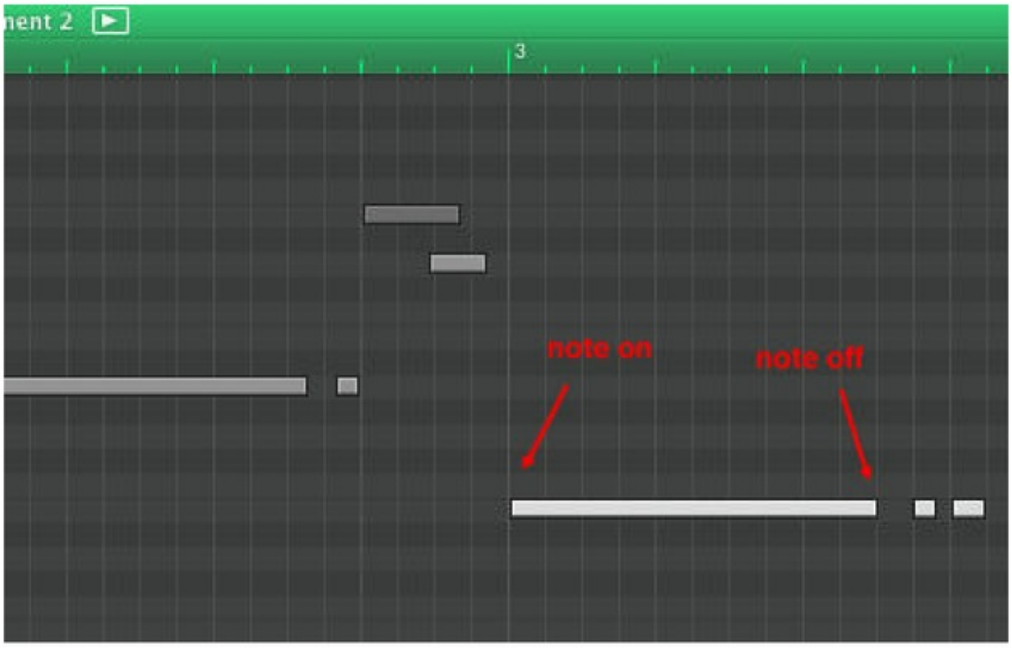

Figure 2.4: Demonstration of a Note On and a Note Off message [1]

2.6.1.2 Transport

MIDI was originally a physical standard, intended for use with MIDI cables to connect multiple digital instruments together, the messages they send are now used in numerous devices to create music and can be transported along different standards such as Firewire or USB. [13]

2.6.1.3 MIDI Files

MIDI files are very lightweight as they can contain a full symphony in a couple of hundred lines of code, the standard allows them to be playable over hundreds of different devices and platforms and there are multiple free and commercial products that allow for the creation and manipulation of MIDI files. How a MIDI file sounds is dependent on the synthesiser you play it back with, leading to it gaining a slight stigma, for sounding robotic and unpleasant. [13]

2.6.2 .NET Framework

The .NET Framework is Microsofts' principle development framework, it runs primarily in a Windows environment however there is cross platform support for Linux and Unix based systems through the use of the Mono Framework [14].

The .NET Framework is composed of two core elements the Framework Class Library(FCL) and the Common Language Runtime(CLR). The FCL provides libraries for GUI development using frameworks such as Silverlight, Windows Presentation Foundation(WPF) and legacy Windows Forms. It offers data connectivity with multiple database formats such as Oracle DB and SQL Server, there is even web development support through the use of ASP.NET(Active Server Page) [14].

The FCL provides each .NET language the ability to use code written in other .NET languages for example a VB.NET application written by an amateur developer, could be extended and contributed to by a C#.NET Developer.

Applications developed in .NET run through a virtual machine named the Common Language Runtime(CLR) this is similar to the Java Virtual Machine(JVM) and in fact the C#.NET Language is often described as Microsofts' version of Java. This CLR provides in built services such as Garbage Collection for memory management, built in security services and internal exception handling.

2.6.2.1 WPF

WPF is a .NET graphical user interface framework used to render interfaces for windows applications, WPF uses DirectX and is built on an XML based language known as XAML, this is used to define and link different components of the interface. libraries to support WPF interfaces have been included with most versions of Windows since Vista [15].

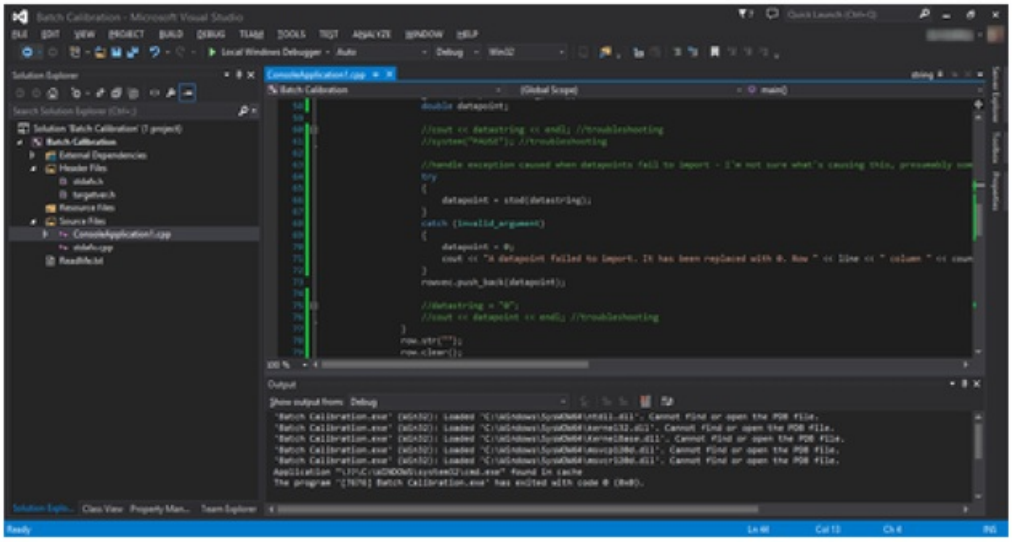

Figure 2.5: Screenshot of Visual Studio showing C++ Code, sourced from wikimedia commons

2.6.2.2 Visual Studio

.NET has an Integrated Development Environment known as Visual Studio, it is used to develop multiple different applications from web services, mobile apps, web sites and enterprise resource planning application it supports source control with multiple different version control systems such as Git, SVN or Microsofts' Team Services. it has a built in XAML editor for building WPF interfaces, and a class designer used to author object orientated classes using UML [16].

2.6.2.3 Benefits and Drawbacks

There are a number of benefits to utilising a framework such as .NET, through the Microsoft Development Network (MSDN) there is an extensive amount of documentation, development standards and interface guidelines. When using the .NET framework on different windows versions you can ensure that if the appropriate .NET version is installed on the users machine it will behave the same way as the development environment and should produce the same results. .NET comes with an extensive collection of class components allowing developers to include different features and capabilities quickly and with relative ease. There are some drawbacks to .NET for a project like this however such as the user having to install and run the appropriate .NET framework if they don't have it, this could cause some users confusion. also with the widespread of HTML 5 and different web based technologies such as Node.JS becoming commonplace there are questions as to why people would write "Thick" Applications.

2.6.3 Web Application Technologies

A web based solution would be able to provide for a wide audience, it has become normal for developers to convert their applications that users would once install locally into a web application that can be experienced instantly by the end user. In terms of interface design, HTML 5 is very flexible and there is scope to create fully functional graphical interfaces at minimal cost to the client, processing would be done entirely on the server side and any feedback gained from the process would be displayed to the user at client side. Benefits of this approach include performance standardisation, Multiple clients would have a very similar experience in processing times due to the bulk of processing being performed on the server.

2.6.3.1 Node.js

Node.js is a JavaScript run-time environment designed to utilise JavaScript as a server-side language. Primarily JavaScript is used to write scripts embedded in HTML which would then run client-side in the web browser, with the use of Node,js scripts written in JavaScript can be run on the server to produce content and feedback before the web page is sent to the client. this has now become a core element of a paradigm known as "JavaScript everywhere" [17].

2.6.3.2 PHP7

PHP is a server-side scripting language designed for web development, it is usually embedded in the HTML markup, it is a flexible language and has been a staple as part of the W/LAMP

(Windows/Linux Apache MySQL and PHP) server architecture. PHP supports object-orientated programming(OOP), and supports many OOP features such as abstract classes, final methods and constructors.

2.6.3.3 ASP.NET

ASP.NET is a server side framework designed for building dynamic web pages and web applications using the .NET Framework, it builds upon Microsofts' previous web technology Active Server Pages, and uses the same Common Language Runtime as the .NET core.

2.6.3.4 Benefits and Drawbacks

In a web application you have a very wide audience, if you use the technologies mentioned above a typical user does not need to install or run any additional software to use your program, this can be attractive to users who do not wish to deal with program installation. One consideration however is that with a web focus, a users image would need to be uploaded and stored on the server for processing even if it is only temporary this would require a storage solution architecture, with a locally installed application all data is stored on the clients machine and the user has full control over their own data. Web applications are also platform independent, although there might be some differences between web browsers most operating systems and devices with web capability would potentially be able to use your application.

Chapter 3

Design

3.1 Introduction

This chapter discusses the design and architecture of the application, including choices of development environment, programming language and data structures.

3.2 Overall Architecture

The software solution is titled ”PixelCato”, the name is a play on words of the Violin technique known as *”Pizzicato”* in which the violinist plucks the strings as opposed to bowing them. PixelCato will perform the following processes in order to derive melodies from Images, Art and Photo’s:

- Take user input.

  PixelCato will allow the user to open most image file types stored on their computer, for processing.

- Analyse the image.

  PixelCato will look at the colour palette of an image, choosing the most common and most appropriate colours for use in the melody.

- Perform colour comparison.

  The software will compare colours in the palette in order to determine which Newtonian colours they are closest to.

- Generate Melody.

  Using the information gained during comparison PixelCato will generate a melody.

- Allow user output.

  The user will be able to playback the melody PixelCato generated, and save the melody at their leisure.

3.3 Development Choices

3.3.1 Framework

From the information gained by the research in Chapter 2, the decision was made to build the software solution using a combination of the .NET technology with a UI developed with WPF (Windows Presentation Foundation). .NET was favoured due to its advantages in being extremely versatile in it's application development frameworks, the plethora of developers and the eco system surrounding it for windows users of which I am.

3.3.2 Programming Language

There are multiple programming language choices with .NET, and they all have interoperability with each other meaning they can work together in the same application.

The language should be based around a design that is modular, therefore it should be object orientated, in addition to this security should be considered, as the language I choose will be based on the .NET framework and Common Language Runtime, there is little to none risk of client compromise.

The choice was made to use C# as the development language, this is for a number of reasons. In C# I have prior experience gained from my industrial placement where I worked on a Manufacturing Execution System, in addition to this throughout University we have studied the Java programming language of which C#'s syntax is based on, therefore the languages are similar to each other and offer developers of either discipline a sense of familiarity.

3.3.3 Development Environment

The project will be developed using the Visual Studio IDE, Visual Studio is the main development environment for .NET applications and provides many useful tools such as UML class diagrams, XML Doc style commenting tools and in built source control management using Git, SVN or Team Foundation.

3.4 Initial Class Design

The initial Class Design of the application contains five main classes,

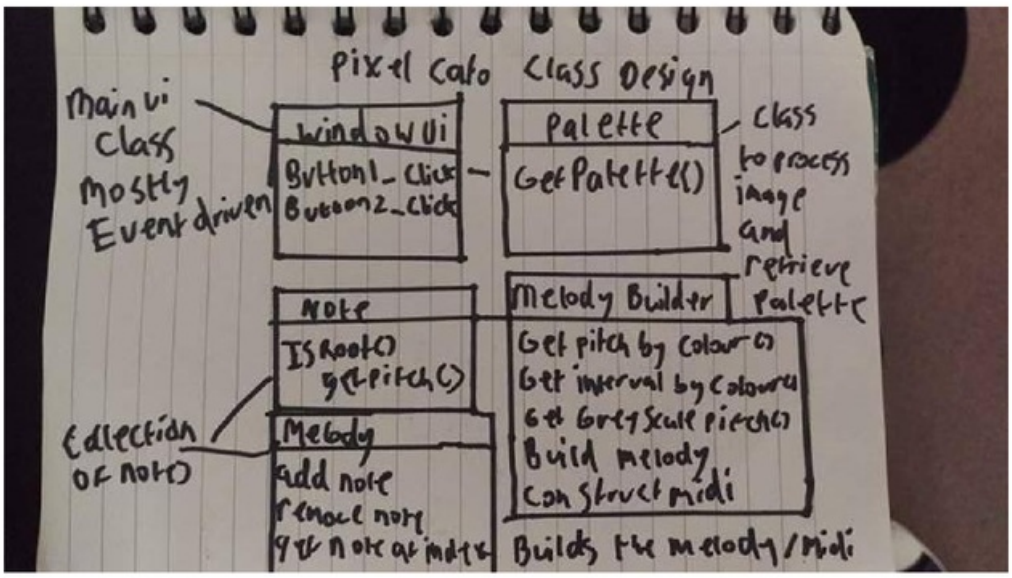

Figure 3.1: Rough hand drawn annotated initial class diagram, does not contain any properties

3.4.1 WindowUI

This class handles all user interaction with the GUI, its methods will be event driven, when a user clicks a UI element such as a menu or a button an event will be fired and run the associated event handler.

3.4.1.1 Properties

in the initial design the UI has no properties.

3.4.1.2 Methods

There will be an event driven method for every button so based on the initial user interface design discussed further in this chapter there would be at least three methods for button click events.

3.4.2 Palette

This class will handle the processing of the users' image, it will have a method to retrieve the colour palette of an image, inside this method a histogram of the colours contained in the image will be created in order to determine the most common colours. This method will also use a

colour comparison algorithm to filter out very similar colours and provide more tonal variety in the potential palette.

3.4.2.1 Properties

Palette will have a single ColourPalette property which is an array.

3.4.2.2 Methods

- GetPalette(image) This method will take an image and analyse it, creating a histogram of the colours which will then be used to appropriately filter similar colours until there is an appropriate representation.

3.4.3 MelodyBuilder

The MelodyBuilder class builds the melody based on the palette provided, its constructor will process the palette and determine which Newtonian colour the image's most common colour is closest to, this will serve as the root of the scale, it will then look at the following colours and perform the same process however this time it will determine the scale degree of the note. For greyscale images the melodies created will be considered atonal with the note's determined by how far away the colour is from black.

3.4.3.1 Properties

The main property of this class will be an array of pitches, which will hold a string based representative of each one of the 12 musical notes.

3.4.3.2 Methods

- Constructor Method(Array palette)

  The constructor method will use the palette to create the notes used in the melody, it will do this by using the methods outlines below.

- GetPitchByColour(String Colour)

  This method will take the colour and return the note associated with that colour by colour theory

- GetIntervalByColour(String Colour, Note root)

  This method will take the colour and the root note of the scale and use colour theory to determine the scale interval

- GetGreyscalePitch(float Delta)

  This method will return a pitch based on how far away the colour was from the colour black and will not adhere to any key structure

- CreateSequence(Melody)

  This method will take a melody and construct a MIDI sequence of notes for playback and exportation.

3.4.4 Note

This is a simple class to represent a musical note, it has properties for pitch and whether the note is a root note, it's methods are simple getter and setter methods.

3.4.4.1 Properties

A Note has two properties a boolean IsRoot which determines whether this note is the root of the scale and a string Pitch which determines the pitch of the note.

3.4.4.2 Methods

The Note class is very simple and contains setters and getters for the above properties.

3.4.5 Melody

I decided I needed a data structure to store a melody, a melody is a series of notes so I thought I would write my own collection for use with the Note class, therefore the Melody class is a collection of notes with methods to add, remove or retrieve notes.

3.4.5.1 Properties

The Melody class will have a root property of type Note, this will hold the root of the scale.

3.4.5.2 Methods

- Add(Note)

  This method will add a note into the melody.

- Remove(Note)

  This method will remove a note from the melody.

- Item(index)

  This method will retrieve a note at the specified index.

3.5 Initial User Interface Design

After doing a sketch of what the design may look like, I designed the initial user interface with the Windows user experience guidelines in mind, this was to ensure a high-quality consistently

styled application. I made sure not to include any stylistic features that would interfere with performance or user interaction. I wanted it to be very minimalistic, often developers can make their GUI's stand out in a bad way and draw unnecessary attention to themselves. In the Windows UX guidelines they list the types of programs and the appropriate usage of WPF to design the best UI for that programs main user type [18].

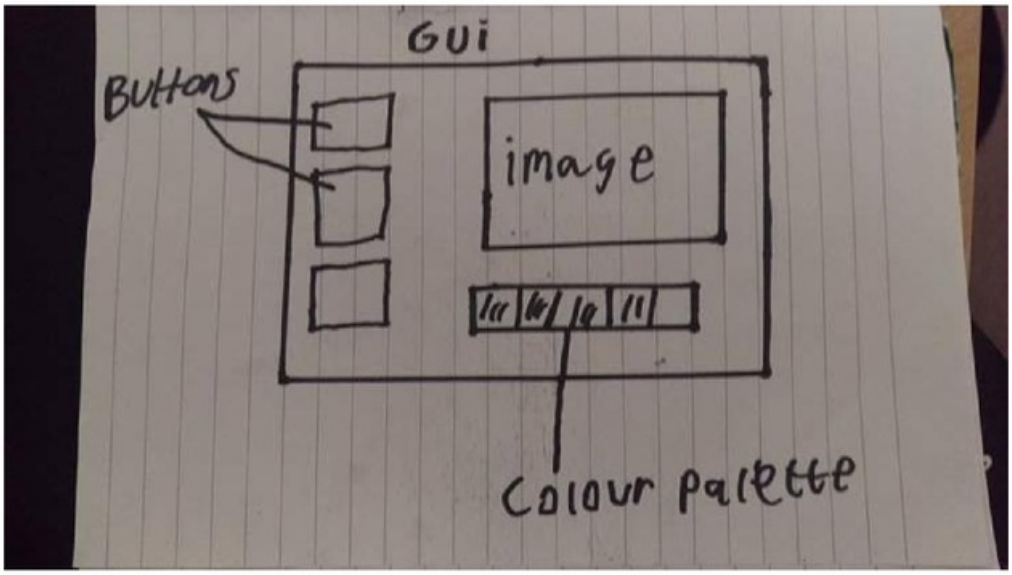

Figure 3.2: Rough hand drawn annotated initial UI mockup

- Productivity Applications.

  These types of user are using your application for long periods of time, and expect familiarity, consistency, and immediate productivity. When designing for this type of user the goal should always be to help the user finish their work.

- Consumer Applications.

  The type of user using consumer applications wish to perform a set of tasks, and their usage will be regular for short periods of time. This user expects an enjoyable experience that can perform the tasks required. When designing for consumers there should be some use of branding and ensure the users can intuitively perform their task efficiently.

- IT Professional Applications.

  The type of user using these applications will generally be technically savvy, their main goal is to perform a task or meet a goal as quickly as possible, they will use the application for a short period of time and get the job done. The main design goal for this user is to get the job done.

PixelCato falls under the umbrella of a Consumer Application, the design goal for the user interface is to make the experience enjoyable for users and perform the required tasks, the interface should also be intuitive it should be fairly clear how to use the program just from looking at it.

The initial WPF design evolved from the hand sketch by moving the palette to the left hand side of the application, leaving room for playback features underneath the image.

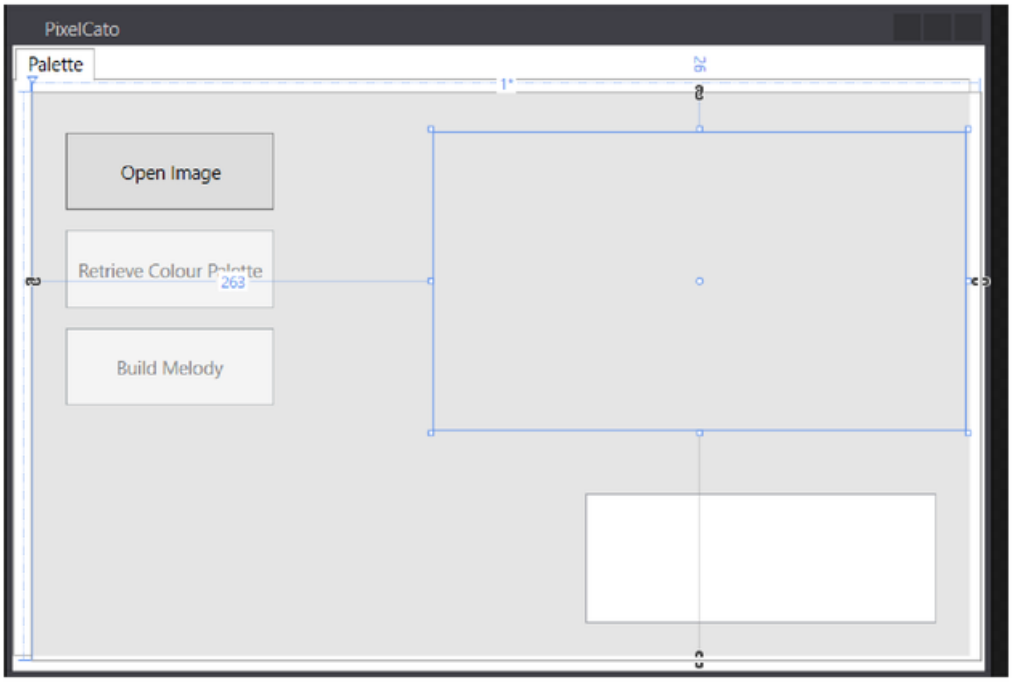

Figure 3.3: Initial WPF Design

3.6 Final Class Design

The final class design has one additional class titled ”CatoLog” to handle logging of information, in addition to this there have been some changes to methods and properties.

3.6.1 MainWindow

The WindowUI class was renamed to MainWindow, to fit in with the WPF convention, as you can see in Figure 3.4 the properties ”mediaPlayer” and ”tempFileName” were added in order to facilitate the ease of interaction with the Media Player as the WPF implementation does not support loading of a byte stream the file must first be temporarily saved. It also has an additional property ”generatedSequence” to handle the ability to save the generated MIDI sequence.

3.6.2 Palette

The Palette class now contains the ColourPalette property to hold the colour retrieved from the image. The method GetPalette(Image) was renamed to RetrieveColourPalette(Image), to differentiate it from a standard get method as it performs additional processing than would be expected

normally.

3.6.3 MelodyBuilder

The two methods GetPitchByColour(Colour) and GetIntervalByColour(Colour, Note), I redesigned into overload methods with the same name, GetPitchFromColour(), if you pass just a colour, it will determine and return the pitch according to the colour. If you pass the colour and the root note, it will determine which scale interval to make the next note.

The method GetGreyScaleByDelta() I initially wanted to include as a third overload for the above method, I couldn't get this to work however so I left it as it's own method, during this process it was renamed to getPitchFromDelta() to offer some sort of consistency.

3.6.4 Note

The Note class remains unchanged from initial design.

3.6.5 Melody

The Melody class now inherits from a .NET CollectionBase class which provides it with common collection methods such as Count() and the ability to use LINQ queries to sort or search the collection.

3.6.6 CatoLog

CatoLog is a class added to handle logging of events, errors and other useful information to a text file, this is for use in case of an error where more tech savvy users can check the log and try to determine the root cause, be it user error or system.

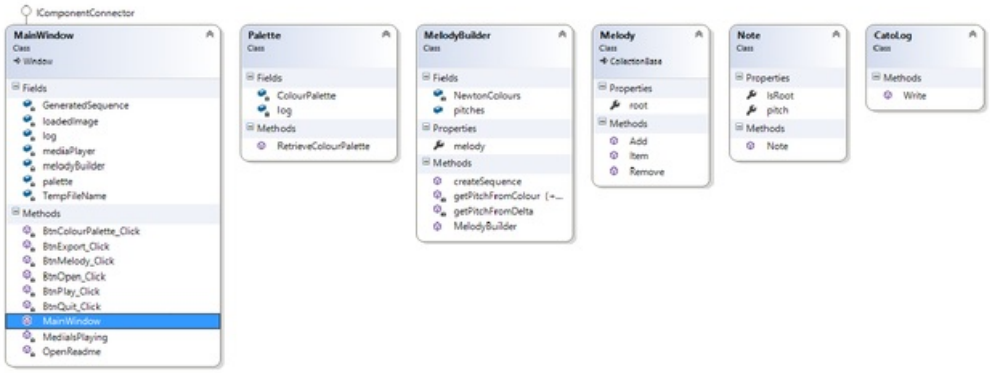

Figure 3.4: Screenshot of Visual Studio class diagram used in the project

3.7 Final User Interface Design

For the final design I made a number of changes, firstly when the application loads it displays a splash screen for 3 seconds, this is to establish branding and try to give off a sense of quality. the splash logo (Figure 3.5) and the icon for PixelCato (Figure 3.6) were designed around the concept of sheet music written for the Treble Clef, the splash logo displays the staff followed by a quavers and demiquavers.

Figure 3.5: Splash Logo

Figure 3.6: PixelCato Icon

In the MainWindow GUI I included a menu bar in order to impose a sense of application familiarity with PixelCato, many desktop applications use a menu bar for features such as opening files or saving.

The menu has two tabs, File and help. File has options to Open the image, retrieve the palette, build the melody and save the melody, the Help tab allows the user to view a readme text file explaining the process of using the software.

In addition to the menu bar I added an additional button for saving the melody and made the design decision to create a panel on the left side of the screen to group the buttons independently

of the image display and playback button on the right I decided to colour this with a blue/white gradient.

Finally I added a loading bar, this will display to the user any time the application is processing something.

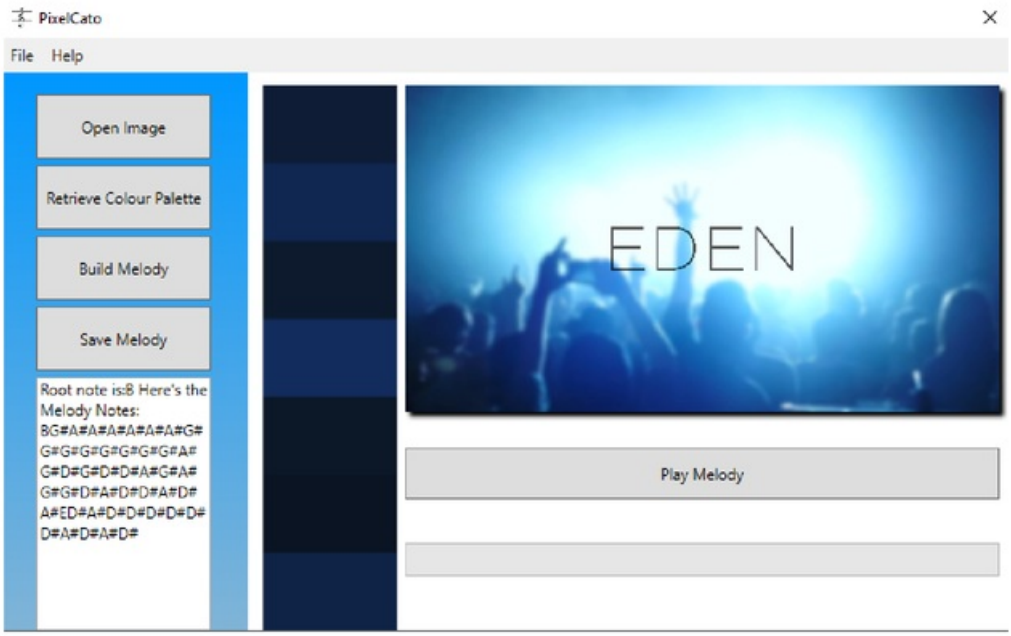

Figure 3.7: Final GUI

3.7.1 UI Sound Design

In addition to the graphical aspect, I composed three different simple sound motiffs for use in the application, there is a splash melody, this is played at the start of the application, and consists of an arpeggio(a chord broken up into single notes) of plucked strings.

There is a finished melody that plays after the application has finished processing something, this is to let the user know that it is finished and awaiting user interaction.

There is also a problem melody that plays a tri-tone in the case of an error, this is to let the user know that something went wrong.

Chapter 4

Implementation

4.1 Introduction

This chapter covers development of the software over the course of this project with a review of sprints, additional development and an overview of algorithms developed.

4.2 Sprints

4.2.1 Sprint 1, 27th Feb-3rd March

In the first sprint I planned to give myself a good foundation of which to develop the rest of the solution, the focus was to setup version control using Git, setup the Visual Studio Development environment and project, develop a basic WPF UI, and write the methods for file handling of images.

To start out I installed Visual Studio Community edition, which is the free development environment Microsoft provide for the .NET framework. Once I had this setup I started a new WPF project and generated the required XAML and .CS files. I then created a private GitHub repository and pushed the initial files, following this I started to develop the MainWindow UI class.

I developed it's basic functionality to open an image and display it, as well as writing the event handlers for user input from the various buttons. In this sprint I didn't encounter any major issues besides a few problems with my syntax being rusty after the break following my industrial year.

4.2.2 Sprint 2, 6th-9th March

This sprint was planned in order to prepare myself for my Mid Project Demonstration, I decided that I would develop the algorithm to detect the palette contained in the image, and display it in the UI, I could then use this as a talking point to discuss with my second marker how I was going to generate melodies based on them.

It was here I developed the Palette class, the palette class's RetrievePalette() method is used to get the most common colours in the image and qualify them by comparing them to filter out similar colours, I used an external library called ColorMine (see Appendix A) in order to easily compare RGB colours to each other this provides a Delta value used to determine how similar a colour is to other colours in the palette. I faced an issue where I was unable to qualify colours very well this led to many different shades of the image taking the entire palette, there was very little variety and the load times could be exceptionally long with large images, as a method of handling this I simplified the colour palette by taking the image and converting it into a 256 colour image, this gave me the variety I required.

4.2.3 Sprint 3, 20th-22nd March

I received feedback during my mid project demonstration from my second marker Chuan Lu regarding the 256 colour palette, she suggested that it would limit the colour possibilities and that I should look at maintaining the original palette but finding a way of better qualifying them.

With this in mind I set out to develop a better solution, I used ColorMine to compare the colour in a pixel to the colours currently contained in the qualified palette, I then disqualified it if it was the same as an already existing colour, this was very efficient in qualifying images, reducing load times and providing a diverse palette that accurately represented the image. When developing this I did have a case of some drastic load times, this was due to a bug in which I wasn't correctly breaking the loop when a colour was compared, meaning it would loop through the entire qualified palette if it found one, causing it to be highly inefficient.

4.2.4 Sprint 4, 3rd-7th April

For this sprint, the plan was to implement the generation of melodies from the image used, in order to do this the Melody, Note and MelodyBuilder classes were developed. The Note class was developed first as it made sense to start with the basic class and then continue from there, this is an incredibly simple class.

The melody class was developed by inheriting from an existing .NET collection interface known as CollectionBase, I used this and wrote methods for adding, removing and retrieving items of the Class type Note, I could then use a melody as a collection of notes which would be extremely useful for melody generation.

Finally MelodyBuilder was implemented with methods to determine the pitch of the notes from the palette, one of the tasks that took me the longest was working out how to make sure that the notes generated remained in the key of the root note, I was able to use the pattern of constructing a major scale in order to achieve this.

I also made a modification to the Palette class, in which I changed the colour comparison al-

gorithm that the ColorMine library was using, this was after finding out that it was identifying colours such as blue, closer to indigo, and reds closer to orange, this was causing some weird melody anomalies. I did however keep the original algorithm as it would better suited for comparison for greyscale images.

I did not have time during this sprint to implement the method for generating a MIDI sequence from the melody, this then rolled over to the next sprint.

4.2.5 Sprint 5, 17th-21st April

This sprint's focus was to take the melodies generated by the MelodyBuilder Constructor and create a MIDI sequence of the notes,the secondary focus was to include atonal melody generation for greyscale images. I looked into a few solutions but ultimately used a library called the C# Sanford MIDI Toolkit (See Appendix A) this allowed me to create midi sequences with relative ease and gave me full control over the tracks, resolution and timing of the events. I ran into a few problems with working out the tick resolution for the MIDI, I wanted the timing to be one note per beat, therefore I had to lower the resolution of the MIDI Sequences to 24 ticks per quarter note and place the note events every 24 ticks. I was unable to get Greyscale melody generation implemented so this was pushed to the next sprint.

4.2.6 Sprint 6, 24th-29th April

The main purpose of this final sprint was to perform final bug fixing, polish the UI, include greyscale melody generation,implement UI threads for performing heavy tasks such as palette retrieval and melody generation and finally provide playback functionality for the generated MIDI sequences.

In the UI I sorted out the XAML grids to ensure there was no wasted markup, I included the menu bar to offer a sense of application familiarity as previously pointed out in the design chapter, this offers the user the ability to use core application functionality from a simple menu, I also included a help option for users to view a quick start guide in how to use the software, the final UI change was the inclusion of the splash startup screen.

In addition to aesthetic changes I implemented threading in the UI I thought that this would be a lot harder than it was, I found threads difficult to understand previously in Java based programming, but in .NET using the BackgroundWorker class, it was very simple and easy to understand.

In the MelodyBuilder class I implemented the GetPitchByDelta() method, and adjusted the architecture of the constructor to fully support atonal melody generation for greyscale images.

The final part of this sprint was implementing playback, I wanted to include the option for users to export the melodies to audio files such as WAV and Ogg Vorbis, however I was unable to develop this. I was able to implement the playback of MIDI files through the use of the WPF media player, an issue with this though is that it could not play files through a byte stream, therefore I have to create temporary midi files for the playback functionality to work as intended.

4.3 Algorithms

I have developed except where clearly specified the following algorithms, copies of the source code for each section can be located in Appendix C.

4.3.1 Colour Palette Retrieval

The Palette class contains a method called RetrieveColourPalette() this takes a parameter of an image byte stream and returns an array list filled with a qualified colour palette. In order to do this the method creates a dictionary which holds what is essentially a histogram of colours.

The algorithm loops through every pixel in the image and checks the colour, if the colour does not exist in the dictionary, it adds it to the dictionary. if the colour exists, it implements the count of the image by one, this creates a histogram of colours which can then be sorted to determine the most common colours.

The Histogram is then sorted by order of most common colour to least common, then for every colour in the histogram the colour is compared to the previous colour, if the colour is not diverse enough (a delta value of under 60) it is not added into the qualified palette, this repeats until all the colours have been checked and the qualified palette has been built.

4.3.1.1 Colour Comparison Algorithms

I use two colour comparison algorithms in the Palette and MelodyBuilder Classes, both of these algorithms are contained in the external library ColorMine, and both of them are designed to retrieve the delta-e (colour difference delta) of a colour I did not write either algorithm.

The first algorithm used is the CIE 1976 algorithm, this algorithm was designed to be used with the LAB colour space, but can be used in RGB and CMYK spaces with the same principles, this algorithm was dropped from use in the coloured melody generation but is retained for greyscale as it does a good job at identifying how far a colour is from the colour black.

The second algorithm is the CMC 1984 algorithm , this algorithm has two parameters, Lightness and Chroma, which allows the user to weight the difference based on a ratio of lightness to chroma, the default is 2:1 and is a commonly used value. I use this algorithm for comparison of coloured images, this is because of a problem I encountered where the previous algorithm I used identified blue as indigo, and red as orange.

4.3.2 Melody Generation

Melody generation is handled by the MelodyBuilder class, there are two different methods of melody generation depending on the type of image, whether it is a colour image or a Greyscale image.

4.3.2.1 Colour

If the image is coloured, the algorithm starts looping through the palette passed to the MelodyBuilder Constructor, for every colour in the palette, it will compare that colour to each of the seven Newtonian colours, the colour that it is most similar to will be identified and used to determine either pitch, or scale degree. If this is the first colour in the palette, this colour determines the root note of the scale creates a note with that pitch and adds it to the melody, every colour after this root note, will be compared and used to determine the scale degree relative to that root note, for example if the colour is closest to Violet the root pitch will be the note "C", then if the next colour is closest to Red, the next pitch will have a scale degree of two, so the second note in the C major scale will be chosen which is the note "D"

4.3.2.2 Greyscale

For a Greyscale image the process is different, every colour in the palette is compared against the colour black, depending on how far away the colour is from black determines what note is added to the melody, this has no restrictions on tonality or scale, and is used to create serialism 12 tone row inspired melodies.

4.3.2.3 MIDI

The MIDI generation utilises the Sanford MIDI Toolkit, the CreateSequence() method creates a new MIDI track and for every note in the melody created by the MelodyBuilder constructor it will add a MIDI Note on message and a Note off message, it will then move 24 ticks ahead in the track ready for the next note.

Chapter 5

Testing

5.1 Introduction

This chapter discusses the testing strategy adopted for use in testing the software, it will look at Unit Testing, Manual Testing, and testing the melody generation of different images.

5.2 Strategy

For this project I have adopted a testing strategy based around the testing of the standard working behaviour, through a combination of unit and manual testing. User testing was initially planned however due to time constraints was decided against.

5.3 Unit Tests

Unit testing is a method of testing software in which you test different modules of the application with example data.

To ensure that my application kept performing correctly after each development cycle unit tests were utilised to make sure that core functionality remained the same, the tests used are contained in the source code in the PixelCatoTests folder.

5.3.1 MelodyBuilder

The unit tests for this class test the constructor method, and the createSequence() method in order to ensure that a melody is built when providing a palette.

5.3.2 Melody

The testing for this class tests the core collection functionality such as retrieving a specified index, adding a note, and removing a note.

5.3.3 Note

The tests for this class test instantiation, and the assignment of pitches and root status.

5.3.4 Palette

The test for this class's retrieve palette method checks the palette retrieved using a test image to determine that the right amount of colours are detected.

5.4 Manual Testing

The majority of the testing of this application was done manually, from testing the UI, to the melody generation.

5.4.1 User Interface

To test the user interface I performed a number of manual tests using the test table below.

Table 5.1: User Interface Testing Table

Test Number	Test Case	Expected Result	Actual Result	Figure
1	Click the "Open Image" Button.	Dialog pop up allowing choice of file.	Dialog pop up allowing choice of file.	5.9
2	Load an image file.	Image displays in the application on the right hand side.	Image displays in the application on the right hand side.	5.10
3	Load an unsupported file.	Error dialog informing user of a problem with the file type.	Error dialog informing user of a problem with the file type.	5.11
4	Click the "Retrieve Colour Palette Button"	Loading bar should activate while processing, upon finishing a sound will play and the palette will appear just to the right of the buttons, left of the image.	Activated loading bar, upon finishing sound plays and palette is displayed.	5.12, 5.13
5	Click the "Build Melody" Button.	Melody notes should appear in the text box and the melody should be ready to play.	Melody notes appear and melody is ready for playback.	5.14
6	Click the "Save Melody" Button.	Dialog should appear allowing the user to save a midi file to a place of their choosing.	Dialog appears correctly.	5.15
7	Click the "Play" Button	The generated melody should start to play, the caption on the button will also become stop, in order to be able to stop the melody.	Generated melody plays.	5.16
8	Click the"Stop" Button	If the melody is playing the play button should become "Stop" and will be able to stop the melody playing	Melody stops.	NA

5.4.2 Melody Generation

To test melody generation, I used a number of different images, including photos, digital images and paintings. I have included a few select images and I will make an effort to try to demonstrate the melody, I will use a tool to generate sheet music from the midi files generated and attempt to comment on the melodies generated, based on what I hear and what is present in the sheet music.

5.4.2.1 Colour Pieces

The first coloured piece is a digital drawing by artist Massimo Di Leo [19].

Figure 5.1: Drawing of a red haired girl

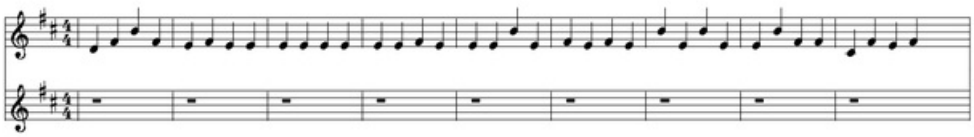

Figure 5.2: Sheet music for the Red Haired Girl

This melody is rather long so I have only included the first nine bars in figure 5.2, the melody starts off ascending and descending from the root, then there are a couple of repeated notes in the third bar, when you reach the 7th-9th bar there are a number of alternating pedal notes. This is

one of the more interesting melodies generated as even though it is long there are a few interesting sections.

This next piece is one of the most famous pieces of art in history, The Mona Lisa, by Leonardo Da Vinci [**?**].

Figure 5.3: The Mona Lisa

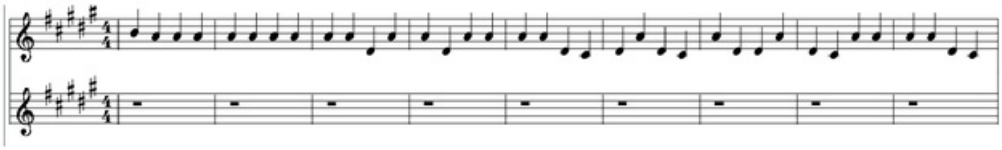

Figure 5.4: Sheet music for the Mona Lisa

The melody (Figure 5.4) generated for this piece is again a long piece , what's interesting about this melody is that it starts off very repetitive however it starts to produce lots of tonal variation after a few bars, jumping from low scale intervals to higher ones.

5.4.2.2 Greyscale Pieces

The first Greyscale piece is a drawing of Princess Mononoke from the Studio Ghibli film "Princess Mononoke" [20].

Figure 5.5: Image of Princess Mononoke used to generate melody

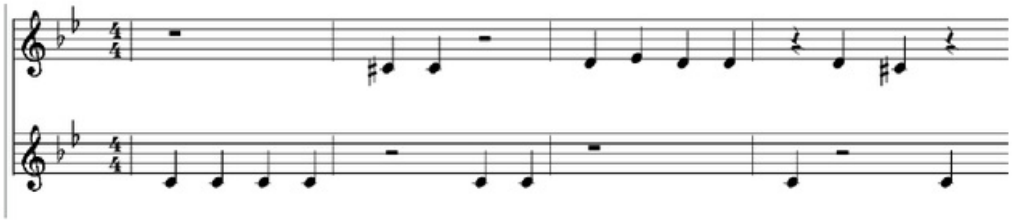

Figure 5.6: Sheet music for the princess mononoke image

As you may be able to see in the sheet music in Figure 5.6 The melody starts with four repeating root notes, it then gains a bit of variety as it moves up by a semitone, returns to the root note and then proceeds to ascend and descend between three semitones.

The next grayscale piece is a drawing of the Eiffel tower surrounded by water [21].

Figure 5.7: Drawing of Eiffel tower surrounded by water

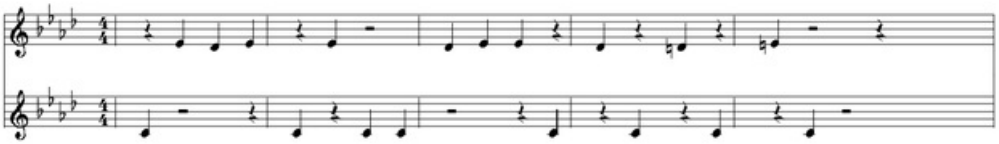

Figure 5.8: Sheet music for the Eiffel Tower drawing

This melody, shown in figure 5.8, starts off with a cell of alternating semitones, the melody then continues to chromatically ascend while occasionally returning to the root note.

5.5 Testing Screenshots

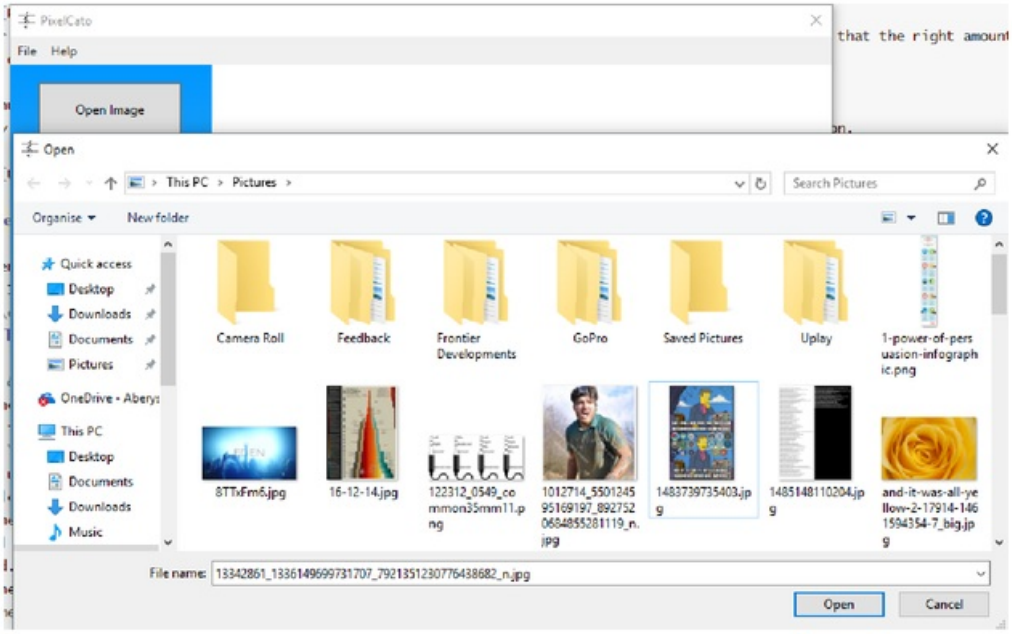

Figure 5.9: UI Test 1

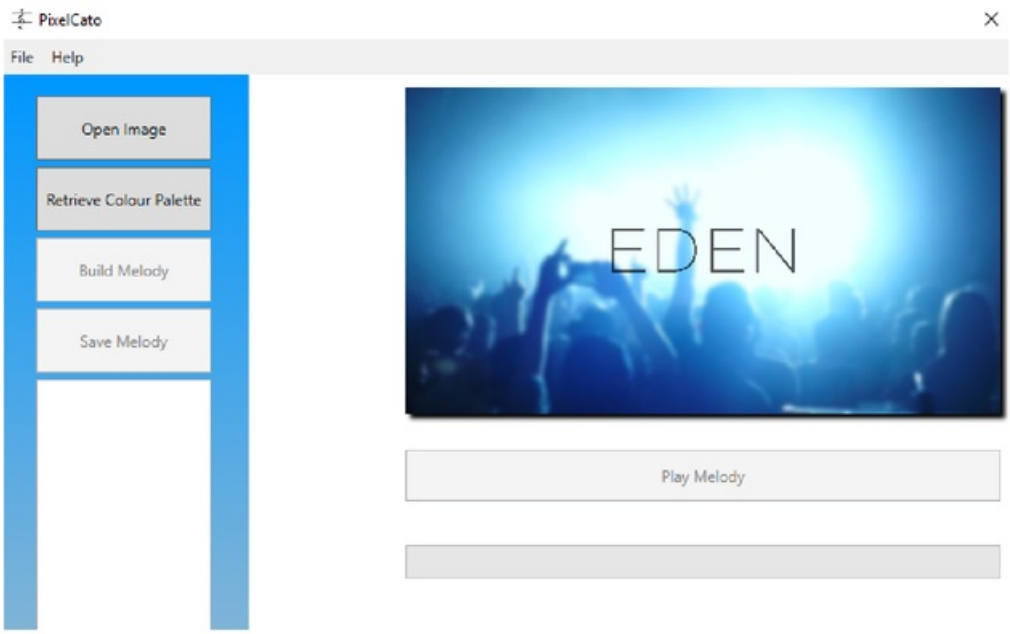

Figure 5.10: UI Test 2

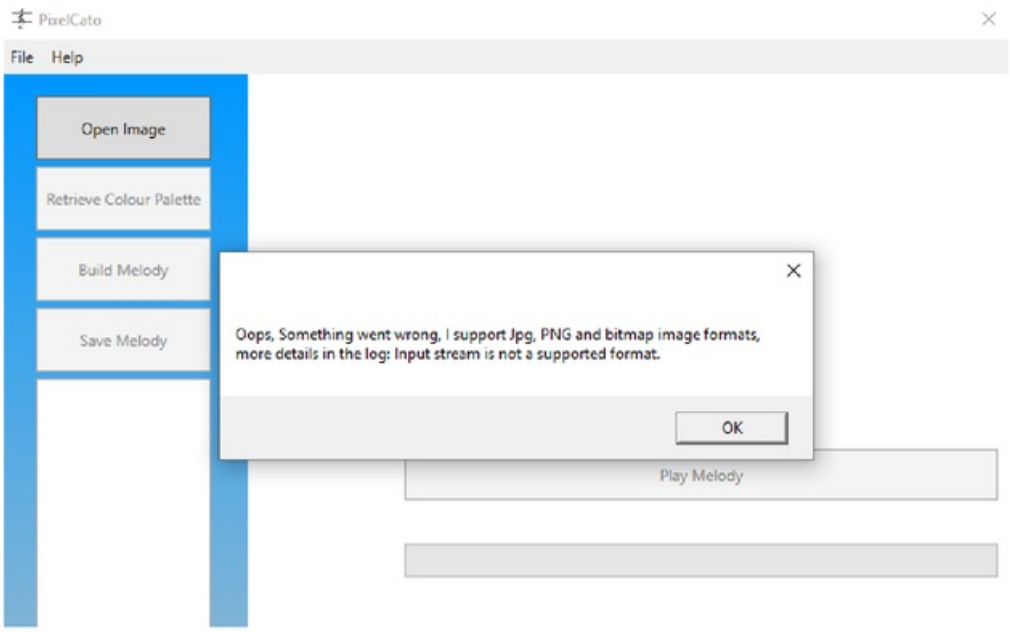

Figure 5.11: UI Test 3

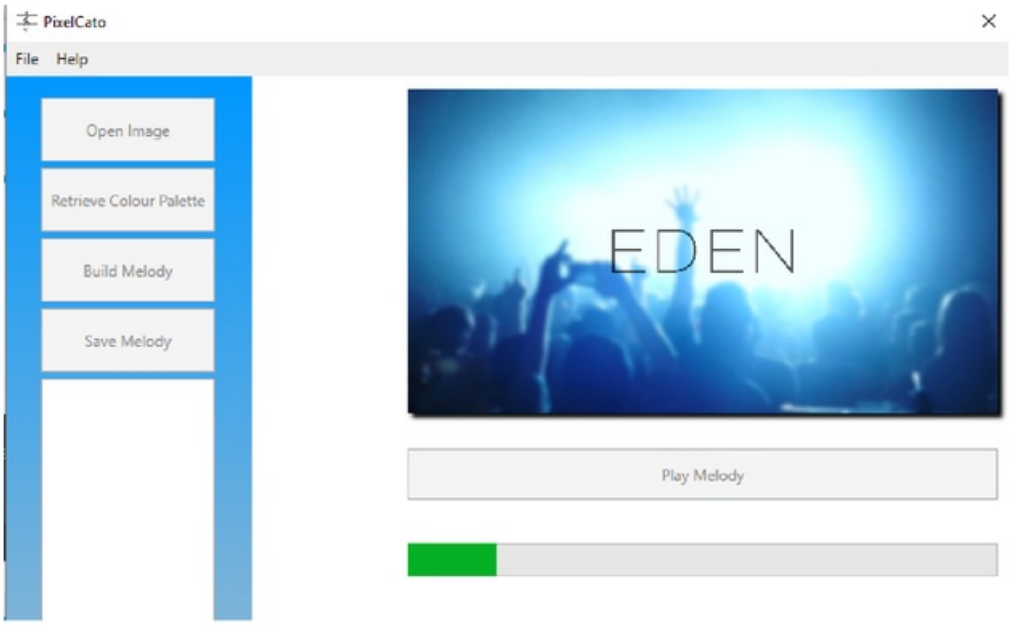

Figure 5.12: UI Test 4

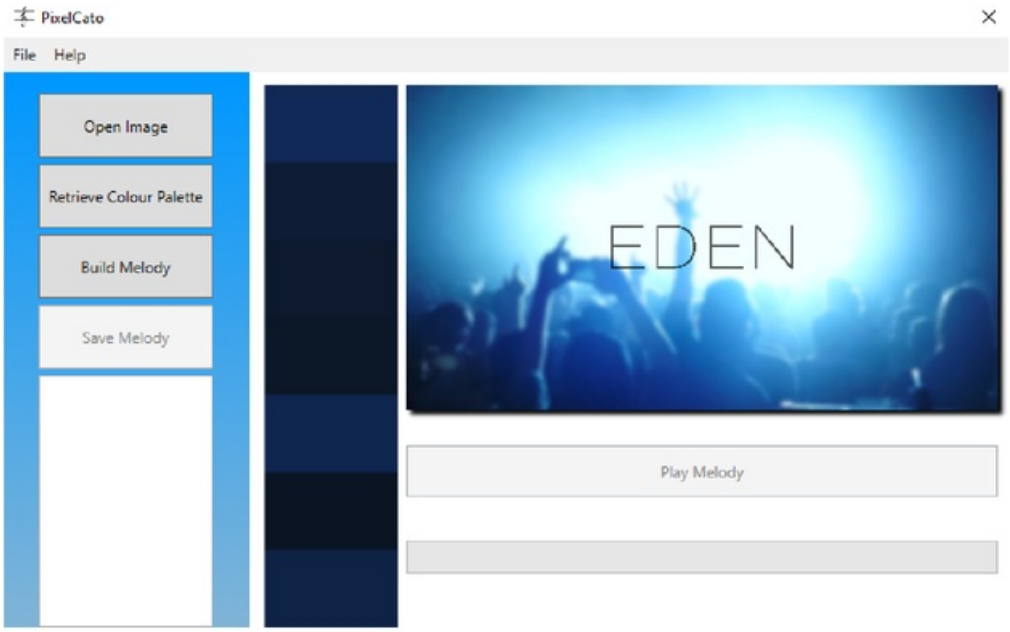

Figure 5.13: UI Test 4 Screenshot 2

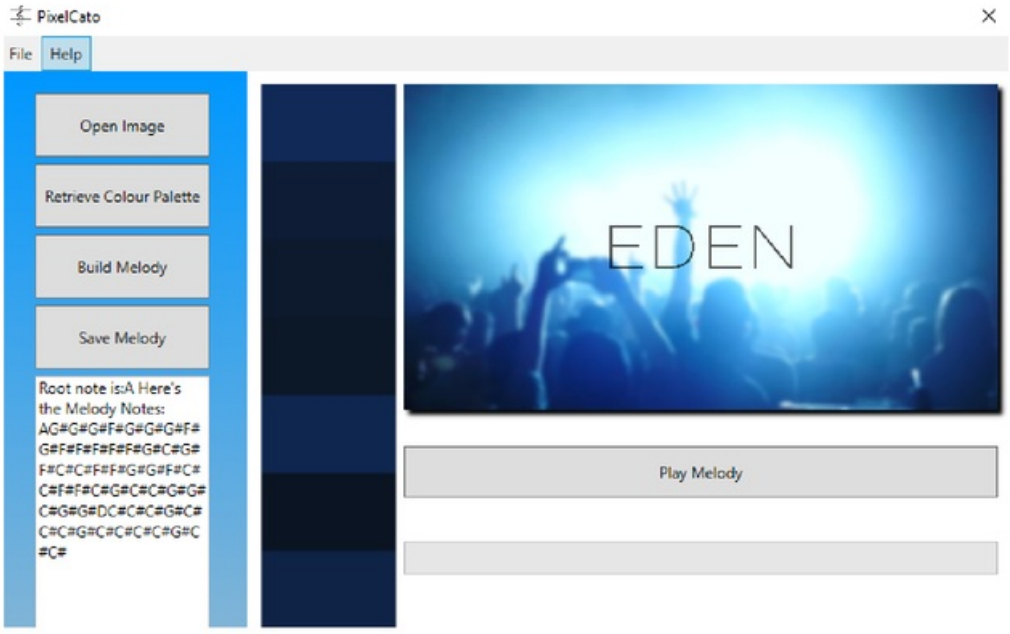

Figure 5.14: UI Test 5

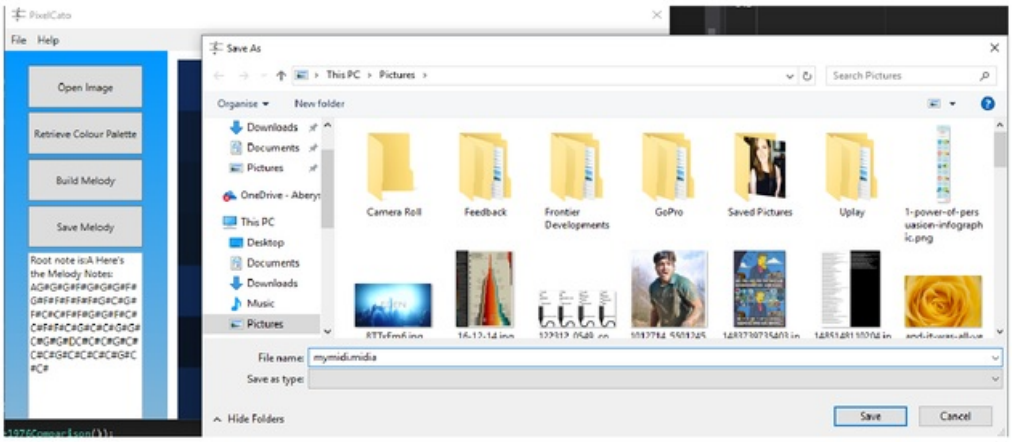

Figure 5.15: Ui Test 6

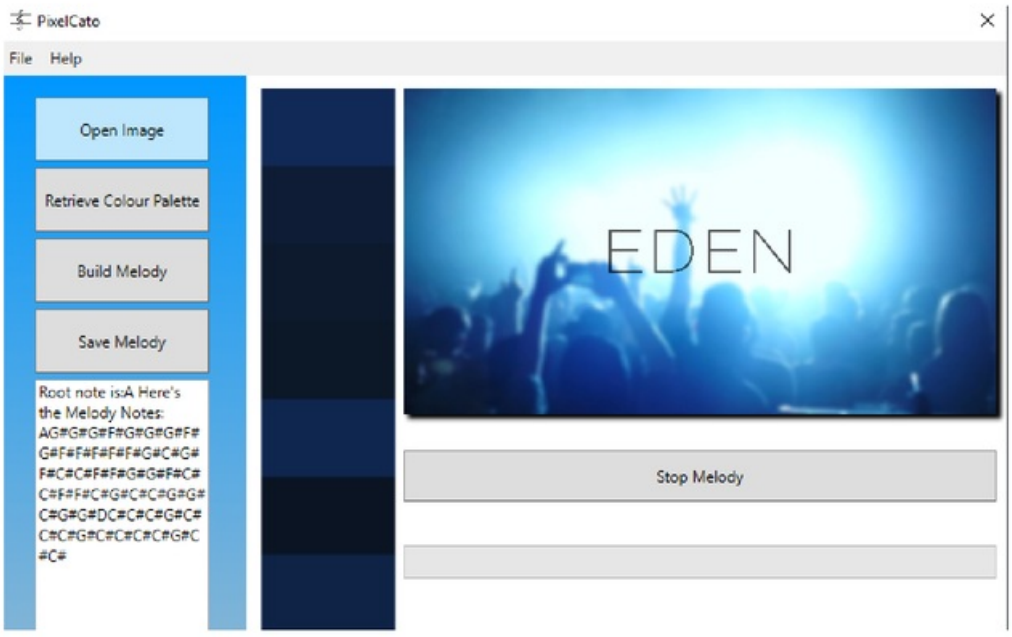

Figure 5.16: Ui Test 7

Chapter 6

Critical Evaluation

6.1 Introduction

This chapter will review the Aim and Objectives from chapter one and evaluate how well they were met, I will give an evaluation on the final design, my personal work flow and how the project could have been improved if repeated.

6.2 Aim

The aim of this project was to produce a piece of software that could take different images and compose melodies based on the colours contained in the image using a method of linking colours to music developed by Isaac Newton.

The aim of this project has been achieved, the software is capable of taking images and composing melodies based on their colours, there are however things to consider. I have found that the process of composing algorithmically is challenging, for a melody to sound pleasing to the ear it has to build tension, it has to use rhythm effectively and should attempt to appeal to the audience's emotion.

The melodies produced by PixelCato often prove to be repetitive, the software is only capable of producing melodies in the 4/4 time signature at a rate of one note per beat, at the tempo of 120 beats per minute, In some pictures where there ends up being a lot of different opposing colours there can be a lot of tonal variety and make the melody more interesting, however it is a common case that most of the melodies can sound soulless, robotic or even random.

In the case of greyscale images, ironically some of the melodies produced can be more diverse in tone, this is due to the variety in delta from the colour black to most colours in a greyscale image.

I would have liked to implement a form of edge detection in order to try to shape the melodies in some way, I thought about this multiple times throughout the project, however I could not come up with a method of shaping the melody without taking away from composition by colour which was a staple of my solution, this would have been a good addition for grayscale images in which there was no colour.

It would be interesting to pair this type of program with Artificial Intelligence algorithms and see whether they can learn from the melodies created and existing famous melodies in order to produce better results from pictures.

6.3 Objectives

.

- Produce a solution that can support multiple image types.

  The software has been tested with and fully supports the following formats: Jpeg, PNG, Bitmap and Gif. As I am utilising the ImageFactory Library, most formats are supported, one notable exception is the .tga format, which neither the library nor the .NET Bitmap Image type supports.

- Produce an algorithmic composition of a simple melody.

  The software is capable of algorithmically building a simple melody based on the colours contained in the image. As previously discussed in the aim section, I would have liked there to be more tonal variety, use of rhythm and less repetition. Currently the software only builds melodies in a Major key for coloured images and atonally(without a key) for greyscale images, I would have really liked to include melodies that would be built around different scale and key types for example Minor (sad), Pentatonic (5 note), due to time constraints and a bit of poor time management on my part I was unable to implement this.

- Playback functionality of the produced melody.

  The system is able to playback the generated melody, it does this by using the .NET Media Player which supports playback of MIDI files, using the Microsoft Midi sound font. There is a problem with this however as the .NET WPF media player does not support playing media from a byte stream, this caused issues as I had to then create temporary MIDI files in a temp folder in order to support playback, a decision of which I was not very happy with.

- Allow the exportation of melodies, for editing and use in external applications. This is the main objective I feel like the system doesn't meet very well but I don't think it fails, It definitely supports exportation of the MIDI format, a user can take the file and import it into most digital audio workstations to edit or a MIDI compatible media player to listen to. I wanted the ability to export to real sound files such as WAV or OGG Vorbis, the problem with this is that the conversion of MIDI to those formats is a difficult process, one which would have required either another external library, or for me to write my own MIDI Synthesiser, which could be a project in itself.

Overall in terms of objectives PixelCato does meet them, however on further reflection at the end of the project I consider that the objectives themselves may have been a bit shallow, I feel if I had defined better objectives then the system in it's current state would fall short, I do not consider this to be a failure, I consider this to be an important learning outcome, better software comes from better requirements just like a good foundation creates a better building.

6.4 Design

6.4.1 Choice of Tools and Platform

Early on in the project I opted to use the C# programming language on the .NET framework, the platform itself has not given me any critical issues or hurdles, and has been generally painless to develop in, I find the work flow of Visual Studio to be very efficient particularly when swapping from UI design to event driven or class methods. However on further reflection I do feel like the project could have been better suited as a web application, this would have given me a wider audience, and provided a good basis for the opportunity to perform some user testing. Web applications are very quickly becoming normality these days, with developers opting for web based solutions over desktop development, the main advantage being platform independence, virtually any device with a web browser has the potential to access your application and with the JavaScript Web Audio API there is real evidence that the web is powerful enough for complicated audio processing.

6.4.2 Object Orientated Design

In terms of Object Orientation I feel that the decision to create the classes Melody and Note were good design decisions, in Object Orientation you should model the world around you, a Melody is a collection of notes, and a Note is a particular pitch. For the final UI Class design I had to make some global properties for the manipulation of different variables between multiple classes with the UI class acting as the middle man, this is bad practice, the UI class should have been only the UI and it's event methods, not a route to pass between classes, if I had followed a design pattern such as MVC, or MVVM, this would have been much better practice. The UI class does however implement threads, this is so when the application is processing something, the UI is unaffected and will still respond to the user displaying a loading bar to indicate that the application is currently working. The Palette class has one method but I believe it could have been renamed to refer to itself as an image process class as even though it's main purpose is to get the palette, it does make decisions regarding colour choice which could be used in a more general sense. An issue with the Palette class is depending on the amount of colours it can have poor performance, the method RetrieveColourPalette() contains nested foreach loops which check the colour in a pixel amongst the current built palette to see if it exists, which can be an expensive process however I was unable to come up with a better alternative. The Palette class could also have been the palette itself created by a Palette builder, I believe this may have been a better design decision.

I believe the MelodyBuilder class was designed well, it has good method choice, use of overload methods and has good performance. The main criticism I have for this class is that it also does colour comparison, between colours defined for the Newtonian colours and the colours in the palette in order to determine notes used, this process could possibly have been a separate class entirely.

6.4.3 User Interface Design

In terms of UI, I believe the interface is well suited for a consumer application, it's simple, intuitive and is aesthetically pleasing. I like how certain elements of the UI are disabled when appropriate, for example you can't build a melody if you haven't got a palette from the image, and you can't

get a palette if you haven't loaded an image, however the argument could be made that some users might get confused with this, being unaware why certain options are disabled to them.

6.5 Project Management

This project has been challenging to manage, I found it difficult to stick to specific sprint goals during sprints, my time management was poor, often leaving development cycles later than anticipated and then having to further push back other features due to lack of progression.

6.5.1 Scrum Values

Overall I believe I adhered to the scrum values where possible but have appropriately evaluated where this isn't the case.

- Commitment I tried to fully commit to sprints, however I found that sometimes I would find it hard to stay focused in longer iteration cycles so I reduced the length from five day to three day cycles. Additionally I did not do a retrospective or review at the end of a sprint, if I had done this perhaps some design and implementation decisions would have been improved, leaving me with a better foundation of which to develop further.
- Courage I always kept a positive attitude and knew whatever problems I was facing I would overcome them eventually.
- Focus For the most part I adhered to this principle however there were occasions where I would swap from one piece of work in a sprint to another, this is just a quirk of my work flow method and is something I generally do in life, I prefer to work on different sections and swap between them than hyper focus on one task.
- Openness I believe I was transparent in issues I faced during my Mid Project demonstration and my weekly supervisor meetings.
- Respect This value is more appropriate for teams utilising scrum.

6.6 Testing

The testing strategy, could definitely have been improved, although the unit tests that were developed performed adequately for basic testing of the system I would have liked to have written more tests which would test the content of melodies for certain images, this would have been useful in determining the effect of different comparison algorithms. I would have liked to have been able to perform some automated UI testing, however due to time constraints and my lack of familiarity with the process this was not done.

6.7 Further Development

There are a number of ways you could improve upon this project and further development. The melody generation could be improved, creating melodies with different time signatures, interesting rhythms and more tonal variety, one way you could do this is through the use of Artificial intelligence algorithms which would compare melodies generated with that of melodies written by competent musicians and improve them based on a learning algorithm. The melody could also be improved through the use of edge detection, I previously wanted to include this, but do to being unable to design an appropriate implementation, and time constraints was unable to develop. the edge could be used in some way to shape the timing, rhythm or pitch of the notes. Another improvement that has been discussed previously is the potential for a web based solution, and with the appropriate use of the Web Audio API, there is potential for powerful audio manipulation, it would also be platform independent and therefore able to reach a much wider audience of potential users. Finally the option to export to audio formats would be a vast improvement as users could then listen to compositions through multiple mediums such as Mp3 players or on their mobile device.

Appendices

Appendix A

Third-Party Code and Libraries

C# Midi Toolkit is licensed under The MIT License by author Leslie Sanford [22].This library was used without modification.

ImageProcessor is licensed under the Apache 2.0 License by author James M South [23]. This library was used without modification.

ColorMine is licensed under The MIT License by author Joe Zack. This library was used without modification [22].

Appendix B

Ethics Submission

Ethics assessment reference number: 6857

AU Status
Undergraduate or PG Taught

Your aber.ac.uk email address
crj10@aber.ac.uk

Full Name
Cai Rhys Jones

Please enter the name of the person responsible for reviewing your assessment.
Prof. Reyer Zwiggelaar

Please enter the aber.ac.uk email address of the person responsible for reviewing your assessment
rrz@aber.ac.uk

Supervisor or Institute Director of Research Department
cs

Module code (Only enter if you have been asked to do so)

Proposed Study Title
MMP-PixelCato: Deriving Music from Art and Colour

Proposed Start Date
30/01/2017

Proposed Completion Date
08/05/2017

Are you conducting a quantitative or qualitative research project?
Mixed Methods

Does your research require external ethical approval under the Health Research Authority?
No

Does your research involve animals?
No

Are you completing this form for your own research?
Yes

Does your research involve human participants?
No

Institute
IMPACS

Please provide a brief summary of your project (150 word max)
The project develops software to take images and convert them into melodies based on Newtonian colour theory.

Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material?
Not applicable

Note: Appendix C has been removed. It contained a lot of source code, which the author might have wanted to go and develop further after the project.

Annotated Bibliography

[1] A. Ghassaei, "What Is MIDI?" [Online]. Available: http://www.instructables.com/id/What-is-MIDI/

Webpage describing a brief overview of MIDI, with information about different event types such as NoteOn, Pitch, Velocity and advanced features such as pitch bending. Accessed March 2017

[2] D. B. Huron, *Sweet Anticipation: Music and the Psychology of Expectation*, pp. 2+. [Online]. Available: https://books.google.co.uk/books?id=uyI_Cb8olkMC&pg=PR5&redir_esc=y#v=onepage&q&f=false

This text describes the psychology behind musical expectation, the section describes musical queues that evoke different feelings or emotion.

[3] I. Newton, *Opticks:: Or, A Treatise of the Reflections, Refractions, Inflections and ... - Sir Isaac Newton - Google Books*, pp. 25–30. [Online]. Available: https://books.google.co.uk/books?id=GnAFAAAAQAAJ&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

Newton's book on refraction, reflection and colours of light, in the passages he describes the light he sees when he shines light from the sun through a prism

[4] "Overleaf: Real-time Collaborative Writing and Publishing Tools with Integrated PDF Preview." [Online]. Available: https://www.overleaf.com/

Overleaf is a LaTeX editor which allows for cloud based backup and collaboration, accessed April 2017

[5] J. Goldsmith, "essay - Color and Sound - a relational history in theory." [Online]. Available: http://www.people.vcu.edu/~{}djbromle/color-theory/color01/Relationship-color-sound-joe_goldsmith.html

Essay that discusses the apparent relationship between sound and colour

[6] "ISO 16:1975 - Acoustics – Standard tuning frequency (Standard musical pitch)," Tech. Rep. [Online]. Available: https://www.iso.org/standard/3601.html

ISO standard for the frequency of the first A pitch above middle C

[7] J. Harrison, *Synaesthesia: The Strangest Thing*, iSBN 0-19-263245-0.

Harrison discusses Scriabins influence from synaesthesia but disagrees that he himself had it

[8] H. C. Plummer, ""colour music-a new art created with the aid of science: The colour organ used in scriabine's symphony prometheus"."

"plummer discusses the design and technology used in the colour organ"

[9] A. Shoenberg, *Style and Idea: Selected Writings of Arnold Shoenberg - translated by Leo Black*.

"Shoenberg describes his style of 12 tone row composition"

[10] K. Kojima, *Kenji Kojima / RGB MusicLab*.

"Web page of the RGB Music Lab, discusses how it works, and the methodology behind it. Accessed April 2017"

[11] M. Rouzic, *Photosounder.com - Image-sound editor & synthesizer*. [Online]. Available: http://photosounder.com/

"Web page of Photosounder application, contains audio samples and main features of the application. Accessed April 2017"

[12] "Computer composer honours Turing's centenary — New Scientist." [Online]. Available: https://www.newscientist.com/article/mg21528724.300-computer-composer-honours-turings-centenary

Article discussing the performance of Iamus's musical pieces and subsequent recording by the London symphony orchestra, accessed April 2017

[13] T. M. Association, "About MIDI-Part 1:Overview -." [Online]. Available: https://www.midi.org/articles/about-midi-part-1-overview

"Article by the MIDI organisation as an introduction to the MIDI standard", accessed April 2017

[14] Microsoft, *Getting Started with the .NET Framework*. [Online]. Available: https://msdn.microsoft.com/library/hh425099(v=vs.110).aspx

Introductory webpage to the .NET framework on the MSDN website, contains a break down of the Common Language Runtime and the Framework Class library, accessed April 2017

[15] ——, *Introducing Windows Presentation Foundation*. [Online]. Available: https://msdn.microsoft.com/en-us/library/aa663364.aspx

MSDN page discussing the Windows Presentation Foundation, accessed April 2017

[16] ——, *Introducing Visual Studio*. [Online]. Available: https://msdn.microsoft.com/en-us/library/fx6bk1f4(v=vs.100).aspx

Archived article on the MSDN website, accessed April 2017, for a previous version of visual studio however the overview is still relevant

[17] Ibm, "JavaScript Everywhere and the Three Amigos (Into the wild BLUE yonder!)." [Online]. Available: https://www.ibm.com/developerworks/community/blogs/gcuomo/entry/javascript_everywhere_and_the_three_amigos?lang=en

IBM article discussing the use of Javascript everywhere from client side to server side with the term "JavaScript everywhere", accessed April 2017

[18] Microsoft, *Designing with Windows Presentation Foundation (Windows)*. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/dn742448(v=vs.85).aspx

Microsoft User Experience Guidelines, accessed April 2017

[19] [Online]. Available: http://dileomax.artstation.com/

Digital drawing of a girl with red hair

[20] [Online]. Available: https://forum.rpg.net/showthread.php?528316-Casting-Couch-thread-greyscale-and-black-and-white-art-needed/page9

Greyscale drawing of Princess Mononoke, accessed April 2017

[21] [Online]. Available: https://alpha.wallhaven.cc/wallpaper/231809

Greyscale drawing of the Eiffel tower, accessed April 2017

[22] MIT.

MIT License

[23] Apache Software Foundation, "Apache License, Version 2.0," http://www.apache.org/licenses/LICENSE-2.0, 2004.

Apache Licence 2.0

[24] J. C. Cranwell, "The Mysterious Correlation of Light and Sound." [Online]. Available: http://www.gootar.com/theory.htm

[25] "Music of the Spheres: Geometry in Art & Architecture Unit." [Online]. Available: https://math.dartmouth.edu/~{}matc/math5.geometry/unit3/unit3.html

"Discusses Pythagoras's idea of the Music of the Spheres", accessed April 2017

[26] "Computational intelligence in music composition: A survey."

Paper discussing computational composition, contains information on the Iamus system, and the methodology behind algorithmic composition

[27] [Online]. Available: http://www.artwallpaperhi.com/Digital/greyscale/grayscale_monochrome_artwork_2560x1600_wallpaper_4374,accessedApril2017

Greyscale photo of a cat, accessed April 2017