Abstract

Psychologists and educators have long recognised the impact of music tutoring on children’s intellectual and emotional development. Piano tutoring in specific can improve a student’s dexterity, coordination, self-esteem, abstract reasoning, creative abilities and personal expression. However, the cost of music tutoring can be substantial, which denies a huge number of children these benefits.

Technology based education is used successfully in various academic subjects. Online tutoring can reach children anywhere in the world regardless of household income. However, piano theory must be put into practice. The ideal e-Learning solution would involve an engaging environment, which would be interacted with using a tangible input device such as an electronic keyboard. This project aims to create an online application, which will encourage and teach children the fundamentals of piano playing and music theory. The child will not only be able to interact with the application using a virtual on-screen piano, but also an external USB Musical Instrument Device Interface (MIDI) keyboard. This will provide the user with a physical simulation of a genuine piano.

Acknowledgments

I would like to express my most sincere gratitude to my girlfriend and family for their support and encouragement whilst writing this dissertation report. I would also like to most heartedly thank my dissertation supervisor Barry Dean whose scholarly guidance contributed to the overall outcome of this project. Last but not least, I would like to express my appreciation to Abumarkub for his expert assistance with the project development.
# Table of Contents

1. **Introduction** .................................................................................................................. 5  
   1.1 Problem Overview ........................................................................................................ 5  
   1.2 Aims ................................................................................................................................. 7  
   1.3 Objectives ...................................................................................................................... 8  
   1.4 Project Scope ................................................................................................................ 8  
      1.4.1 Stakeholders ........................................................................................................... 9  
      1.4.2 Application Context ............................................................................................. 9  
   1.5 Research Methods ....................................................................................................... 9  

2. **Research** .......................................................................................................................... 10  
   2.1 Existing Applications ................................................................................................... 10  
      2.1.1 Online Applications .......................................................................................... 10  
      2.1.2 Downloadable Applications .......................................................................... 12  
      2.1.3 CD-ROM Applications .................................................................................. 13  
   2.2 KS2 Learning Characteristics ..................................................................................... 14  
      2.2.1 Ages 7 to 8 ......................................................................................................... 14  
      2.2.2 Ages 9 to 11 ....................................................................................................... 15  
   2.3 Piano Tutoring ............................................................................................................. 16  
      2.3.1 Fundamental Piano Concepts ........................................................................... 16  
      2.3.2 Further Piano Concepts ................................................................................... 17  
      2.3.3 Sight Reading Teaching Methods ...................................................................... 18  
   2.4 Electronic Learning ...................................................................................................... 19  
   2.5 MIDI Connectivity ...................................................................................................... 20  
      2.5.1 MIDI Fundamentals ......................................................................................... 20  
      2.5.2 MIDI to Flash Rationale .................................................................................. 21  
      2.5.3 Possible Connectivity Methods ....................................................................... 22  
   2.6 Development Lifecycle ............................................................................................... 24  

3. **Requirements Analysis** ............................................................................................... 25  
   3.1 Functional Requirements ......................................................................................... 25  
   3.2 Non-Functional Requirements ................................................................................ 27  
   3.3 MoSCoW Method ....................................................................................................... 28  

4. **Design** ............................................................................................................................ 29  
   4.1 Interface Design ......................................................................................................... 29  
      4.1.1 Characters ......................................................................................................... 30  
      4.1.2 Instructions ....................................................................................................... 30  
      4.1.3 Help & Tips ....................................................................................................... 31  
      4.1.4 Input Configuration ......................................................................................... 32  
      4.1.5 Virtual Keyboard ............................................................................................ 33  
   4.2 Game Design .............................................................................................................. 34  
   4.3 Storyboards ................................................................................................................ 38  

5. **Implementation & Testing** .......................................................................................... 42  
   5.1 Receiving MIDI Input .............................................................................................. 42
1. Introduction

1.1 Problem Overview

Psychologists and educators have long recognised the impact of early experiences on a child's intellectual and emotional development. Studies show that music tutoring can improve a student's self-discipline, dexterity, coordination, self-esteem, thinking skills, listening skills, creative abilities and personal expression.

Countless studies have been carried out which address the effect of music on academic development. Research conducted by University of Wisconsin Professor Dr. Frances Rauscher, published in 1997 in the Scientific Journal ‘Neurological Research’ concluded that children involved with piano tutoring at an early age show significantly enhanced abstract reasoning abilities, which are critical to success in science and maths. In a study released in 2000, students aged 7-8 from a low income school in Los Angeles were given eight months of piano training, as well as the use of newly designed music software. These students, taking the Stanford 9 Maths Test, went from scoring in the 30th to the 65th percentile (Gordon, 1999).

Music education from an early age clearly has a great impact on a child’s ability to learn and express themselves. However, the cost of piano lessons can be far too high for low-income families to afford: typically ranging from £20 and upwards for one hour. The aim of this project is to provide all children with an equal opportunity to learn the basics of the piano though the medium of electronic learning (e-Learning). Technology-based education allows for On-demand availability, Self-pacing and Interactivity. These aspects engage the user, pushing them rather than pulling them through education. Piano tutoring applications are available to purchase as CD-ROM format and as software pre-built into electric piano hardware, the cost of which can be substantial. This project aims to create an online application, which will enable children from any economic background, whether at home or in school, to grasp the fundamentals of piano and gain an interest in music via an engaging interactive experience.
The child will not only be able to interact with the application using a virtual on-screen piano, but also an external USB MIDI (Musical Instrument Device Interface) keyboard. This will provide the user with a tangible simulation of a genuine piano. MIDI keyboards are widely available in schools and can be purchased at a low cost. MIDI is an industry-standard protocol that enables electronic musical instruments such as keyboards to send event messages to a computer. MIDI devices range from compact two octave keyboards (Figure 1) to 49 key semi-weighted keyboards (Figure 2) to full size 88 key hammer-weighted keyboards. There are currently no web-based interactive piano learning applications with MIDI connectivity available online.
Gavin Meredith, founder of the website musicteachers.co.uk is one of many piano teachers who use the Internet to enhance music tuition. In 1998 he helped to set up this online resource, which predominantly caters for tutors. However, the team have found that children respond very well to a particular section of the site. Meredith explains ‘a couple of months ago we commissioned an online game called Name That Note. It's very simple, but it is intended to put learning notes in a fun context.’ (Guardian 2008). His pupils loved the game and they now intend to invest in more of the same. Meredith continues; ‘you can't always rouse a natural musical curiosity in them, but if you go through a medium that they are relaxed with and that they like, such as little flash games on the Internet, then that's a good route to get them interested.’

Adobe Flash is the industry-leading authoring environment for creating engaging interactive experiences. Flash offers developers a sophisticated programming environment, particularly since the release of ActionScript 3.0, an Object Oriented Programming language used within Flash. Flash is optimised for the display of graphical information and which is tailored for operation over the Internet. In terms of deployment, Flash content is accessible to almost all Internet users; Flash Player is available to 99% of users (Flash Player Census, Millward Brown 2008). With these factors considered, it can be concluded that Flash will be the most appropriate tool for this project. However, Flash does not currently support MIDI. Therefore, the focus of this project will be divided into two sections:

1. Establishing a connection between MIDI and Adobe Flash (Developing a Flash application which will receive MIDI messages)

2. The design and development of an online interactive piano learning environment for Key Stage 2 children (ages 7 to 11)

1.2 Aims

This project will focus on the research, design and implementation of a solution, which will connect MIDI with Flash in order to create a piano based e-Learning environment for children (Flash Music). The project will attempt to reveal an efficient method of receiving MIDI messages within an Adobe Flash application.
Conclusions will be made regarding the compatibility of Flash and MIDI. Once the MIDI connection has been established the project will be extended via the development of an online piano-tutoring interface.

The online application will feature fundamental piano training in the form of mini-games. The user will have the ability to play the games and monitor their progress. For the scope of this project, the aim is to develop at least one finalised training game for demonstration purposes. The number of games developed may be increased depending on the time constraints of the project. Users will interact with the application using a simple USB MIDI keyboard. For accessibility purposes, a QWERTY keyboard input alternative will be provided.

1.3 Objectives

- To develop a method of receiving control messages from a MIDI keyboard within a Flash application.
- To identify and evaluate existing piano tutoring software.
- To research the most effective methods of teaching children piano basics.
- To research the most effective methods of engaging a child within a multimedia learning environment.
- To design and develop an online interactive piano tutoring application with MIDI connectivity.
- To design and develop at least one fully functional piano training game utilising a MIDI keyboard as an input device.
- To develop a QWERTY keyboard input alternative for users with no access to a MIDI keyboard.
- To evaluate the compatibility and usefulness of linking MIDI and Flash.

1.4 Project Scope

This project does not aspire to produce a finalised piano tutoring program. The developed application will demonstrate the desired features stated within the project aims. Providing the aims are achieved, the application can be developed further following the completion of the project.
The scope of this document is to provide information to enthusiasts who are interested in e-Learning systems (specifically music oriented) and MIDI to Flash connectivity. As a result, further developments may be achieved in this particular area using the conclusions drawn.

1.4.1 Stakeholders
This project will benefit Schools (Music Departments), Parents and End Users (KS2 Children, Ages 7-11). The project report and source code will interest MIDI, Flash and Multimedia enthusiasts.

1.4.2 Application Context
The application will be developed for online use. Therefore, file size is an important factor to be monitored during the development stages. The application should be compatible with the most commonly used Internet browsers: Internet Explorer and Firefox (W3C 2009). Children in schools and at home will use the virtual piano tutor; the application will be used on various systems, therefore cross compatibility with Microsoft Windows and Apple OS X will be considered. However, for demonstration purposes the application will be developed for use with Microsoft Windows XP.

1.5 Research Methods
In terms of design and development, research methods will retrieve both qualitative and quantitative findings. Creating a persona will be the initial stage of the research process, it will provide an early foundation for design concepts and also allow a closer relationship with potential users to be obtained. Interviews and questionnaires will be conducted in order to gain a greater understanding of user requirements. Understanding the user’s tasks is of vital importance; especially considering the application is designed for children aged 7-11, who are therefore novice users. A list of potential tasks will be devised. A Hierarchical Task Analysis will then be devised for each action; this analysis will return an ordered step-by-step illustration of the tasks. Key areas to research will include: existing applications, Key Stage 2 learning characteristics, piano-tutoring methods, e-Learning techniques and MIDI connectivity. The research will then be used to construct system requirements. A list of requirements will be devised using the five main requirement categories:
Functional Requirements, Data Requirements, Environmental Requirements (context of use), User Requirements and Usability Requirements (Preece et al. 1994). This list will be useful to refer to when designing lo-fi prototypes, as it will provide a clearly defined understanding of what is required for a successful piano tutoring application.

In order to manage a project it is crucial to establish a clear set of prioritised requirements (Haughey PMP, 2008). The MoSCoW method was developed by Dai Clegg of Oracle UK in 1994 and is used to prioritise requirements. The capital letters in MoSCoW stand for:

- **M** Must have this.
- **S** Should have this if at all possible.
- **C** Could have this if it does not affect anything else.
- **W** Won’t have this time but Would like in the future.

MoSCoW is used to determine which requirements must be implemented first and which must come later or will not be implemented at all.

2. Research

2.1 Existing Applications

There are currently no online piano-tutoring applications with MIDI connectivity available. Therefore, this report will research similar programs, which aim to teach or encourage an interest in playing piano. The focus of the research will be on piano tutoring for children, however the research will also address other e-Learning applications with regards to interaction and interface elements.

2.1.1 Online Applications

There is a large selection of music orientated games on the web: younger children can learn to play nursery rhymes on a virtual rainbow keyboard, whilst those looking to progress to notation can learn about quavers as they fire missiles and defend from alien attacks (Figure 3). These scenarios are beginning to be taken seriously as a method of capturing a child’s attention and provoking a desire to keep learning. Even the prestigious Association Board of the Royal Schools of Music, has a section of its student site dedicated to online games.
Name That Note

The Music Teacher UK website hosts a game which is geared towards teaching children note names and positions on a musical staff. The game uses bright colours and sound effects to engage the user. When the game is initially loaded, the child is prompted to choose a clef to learn on (Figure 4).
The user is then presented with a staff and a random note. They must identify the
note before the time runs out (Figure 5). At the end of the game the child has the
opportunity to upload their score to a public scoreboard.

![Figure 5 - Name That Note Game Play](image)

### 2.1.2 Downloadable Applications

A large selection of downloadable piano tutoring applications are available online,
most of which are simple free play applications. These applications require
downloading and installing and most are executable files that only work on Windows
systems. However, Synthesia (Figure 6) is an advanced application built in C++,
which teaches the user to play the piano using falling notes in a similar way to the
popular game, *Guitar Hero*. The main features include:

- MIDI keyboard connectivity.
- User progress tracking with per-song scoreboards.
- Slow, fast-forward, and rewind songs while learning new parts.
- Ability to practice left and right hand parts together or separately.
- PC and Mac compatible.
- Ability to learn to play any MIDI file (available from various websites).
- No sheet music reading is required.

Synthesia is generally targeted at more advanced piano players and seems to be
grounded towards entertainment rather than education. The application teaches the user
to play songs in an unconventional manner; the method used does not actually teach piano theory, as the grand staff is replaced by colour-coded squares.

![Figure 6 - Synthesia Interface](image)

### 2.1.3 CD-ROM Applications

There is an abundant supply of piano tutoring programs available for purchase. These applications are advanced software packages with a large selection of features. For example, Adventus Piano Suite Premier is a popular program consisting of features such as: *Theory Thinker* - a narrated music theory course with lessons and practice exercises for the keyboard. *Piano Player* - a variety of learning methods aimed at developing sight reading strategies and playing skills. The software makes use of interactive feedback, which is geared towards motivating success. *Personal Profile* - provides detailed evaluation reports which track the child’s progress. Reports can be logged for any number of users. *Composers Corner* - the user can create and print their own simple multitrack arrangements; they can also load in MIDI files to learn to play. *History Happens* - includes biographies of over 350 composers and performers. *Games* - the software includes various music games geared towards reinforcing music theory and improving dexterity. However, this software package currently retails at £67.48 excluding the price of a MIDI keyboard, which retail from £30 and upwards.
2.2 KS2 Learning Characteristics

This section of the report will delve into the ways in which children actually learn; which is vital in regards to constructing a thorough set of user requirements.

2.2.1 Ages 7 to 8

Members of this age group have short attention spans, typically between 5 and 10 minutes. It is best to divide activities into small pieces or steps (North Carolina Cooperative Extension Service 2002). Therefore, piano-tutoring games for children must be short and lucid in order to hold the child’s attention. Reading skills vary in this age group. Activities need to be read to the child, however verbal and visual instructions should also be provided. Instructional elements of the application will be broken down into steps using a clear walk-through dialogue method including text, animation and audio. Children of this age are very concrete thinkers and benefit from hands-on learning and activities where they both do and see things. The MIDI interface combined with on-screen visuals and auditory feedback will be used to cater for this learning characteristic.

Children of this age require praise and encouragement for even small successes. They have difficulty separating themselves from the activity; ‘they may view any evaluation as a reflection on them as a person’ (Indiana 4-H 2002). The application must provide positive feedback to the child for any successful interaction. When the child is less successful they will be notified in a sensitive manner and encouraged to keep trying.

The Youth Organisation, 4-H ‘Helpers Guide To Youth Learning’ explains that competition or activities, which reward a single winner or ‘best’ person, should be avoided; ‘Instead of comparing youth to each other, compare present to past performances of each child’. This age group is more concerned with actually doing a project than completing it. These children should be presented with activities, which they can complete successfully. These are important factors to consider during the development of the training games.
2.2.2 Ages 9 to 11

Activities need to encourage physical involvement because boys and girls of this age are extremely active. Hands-on involvement using objects is helpful. Youth this age are still fairly concrete thinkers and will give more attention if they are seeing and doing things (Indiana 4-H 2002). With the developed application the child will have the opportunity to play the keyboard and follow instructions on-screen. These two elements combined will be particularly helpful for concrete thinkers.

Children of this age are just beginning to think logically and symbolically and are starting to understand abstract ideas. As they consider an idea, they tend to think it is right or wrong, fun or boring. This is where it is key to ensure the training games are fun to play, so the child feels they are playing a game as oppose to just learning a piano scale, for example. Just like younger children, many prefer individual evaluation to group competition. Comparison with the success of others is difficult for these children because it can erode self-confidence. Instead of comparing children with one another, positive self-concepts should be built by comparing the individual’s present and past performances (CyferNet 2006). A high-score system could be developed as a feature of the e-learning application. However, an open high-score system would cause the child to feel that they must compete with their peers. With these factors considered, it would be beneficial for the child to be able to monitor their own personal progress. Children of this age are generally concerned with immediate rewards. The satisfaction of completing a project often comes from pleasing the helper, rather than from the value of the activity itself. The helper in the case of this project will be an on-screen character. With the use of animations and dialogue, the character can be made to respond to the child’s actions. The better the child does at the task; the more responsive the virtual tutor becomes.

Children in this age group have a strong need to feel accepted and worthwhile. School and other pressures are demanding. A continued emphasis on successes is needed. Youth want to know how much they have improved and what they could do better next time (Indiana 4-H 2002). Therefore, positive feedback and encouragement will be of vital importance within the developed application.
2.3 Piano Tutoring

This section of the report will focus on piano playing concepts, which a child must understand in order to further their playing skills.

2.3.1 Fundamental Piano Concepts

- **The Keyboard**
  The piano consists of alternating white and black keys. The white keys are grouped into sets of eight keys called *octaves*. The black keys are grouped into sets of twos and threes. Beginning at C (the white key directly to the left of each set of two black keys), the white notes are named as follows: C, D, E, F, G, A and B (Appendix 5).

- **Reading Music**
  Reading music is actually simpler than it appears at first glance. The seemingly random combination of lines and spaces is quite orderly and quickly mastered. The *grand staff* consists of two *clefs*, the *treble* and the *bass*. Each staff is represented by a set of five lines and four spaces (Figure 7).

- **The Treble Clef**
  An extra line below the treble clef represents Middle C. The first line is E. The second line is G. The third line is B; the fourth, D; and the fifth, F. These can be remembered easily with the commonly used acronym *Every Good Boy Does Fine*. The spaces, beginning at the bottommost are F, A, C, and E, represented by the acronym *FACE* (Appendix 6).

- **The Bass Clef**
  Once again, beginning at middle C, represented by a line above the top line in the clef. The pinnacle line is A and the second line down is F. The centre line is D, the second line from the bottom is B, and the bottommost line is G. The acronym *Great Big Ducks Fly Away* can help children remember which note is represented by which line. The spaces, beginning at the bottom, are A, C, E, and G: *All Cows Eat Grass* (Adams 2008).

- **Rhythm**
  The basic unit of rhythm is a measure. The simplest rhythm is 4/4 time. This means that there are four quarter notes in each measure. A half note is the
equivalent of two quarter notes. A dotted-quarter note is equivalent to three quarter notes and a whole note is equivalent to four quarter notes (Appendix 4).

- **Sharps & Flats**

A black note can be referred to as the *sharp (#)* of the note on its left. A black note can also be referred to as the *flat (b)* of the note on its right. The set of two black keys may be called C# and D# or Db and Eb. The set of three black keys are F#, G#, and A# or Gb, Ab, and Bb.

![The Grand Staff](image)

**Figure 7 - The Grand Staff**

### 2.3.2 Further Piano Concepts

Further piano concepts are usually presented to the child after they have grasped the fundamentals. If the application were to be expanded, these concepts should be incorporated into the learning environment:

- **Tone & Pitch**
- **Scales & Arpeggios**
- **Chords & Harmony**
- **Fingering**
- **Expression**
2.3.3 Sight Reading Teaching Methods

This section of the report will describe the research findings in terms of teaching children the relationship between key positions and note names. The research is also geared towards methods of teaching children the basics of sight-reading; recognising notes on the Grand Staff.

Rainbow Piano Technique

Piano tutor and graduate in Piano Teaching Annie Wang recently published her book entitled Rainbow Piano Technique (Wang 2008). Wang and Heiko Hoffmann, a researcher at the University of Southern California, developed the technique. Each note on the grand staff is coloured according to its pitch (Figure 8), corresponding coloured markers are provided with the book. These markers can be cut out and attached to a piano keyboard (Figure 9). The child can then build up a note association with each colour. A recent news article on the Los Angeles Times website reports of a 3 year old boy who successfully uses the rainbow technique; ‘Instead of seeing notes like C or F sharp, Richard sees purple and dark green, and knows their corresponding piano keys.’ (Wagner 2008). Within three months the boy was able to play simple songs using both hands and he was weened off the colour stickers, one by oneuch like learning to ride a bicycle using stabilisers.

![Figure 8 - Grand Staff - Rainbow Piano Technique](image)

![Figure 9 - Colour Coded Piano Keyboard](image)
This technique could be applied to the development of this project. Colours can simply be applied to the on-screen virtual piano keyboard and corresponding staff. This method may be useful to apply to a hint feature of the application. For example, if the child is having difficulty matching a note on the staff to a key, they can click a button, which will display the rainbow colours for a limited time period. The child will then be able to identify the note using the rainbow piano technique.

**Piano by Number**

Assigning a number to each piano key is a popular technique for teaching young children piano. Children's music educator and Emmy Award winning composer John Aschenbrenner created a simple technique ‘Piano by Number’ (Piano Is Easy 2008). The child is supplied with a collection of stickers numbered one to twelve, the stickers are placed on each white key (two octaves). The child can then play along with numbered musical notation. When the child is comfortable with a song the stickers are removed much like the rainbow technique. Both Piano by Number and the Rainbow Piano Technique have been proven to be beneficial to early learners. However, if both techniques were used simultaneously this would cause confusion amongst the children. Therefore, a decision must be made on which technique to use. Research shows that children respond rapidly to colours, even from a young age. Children love bright colours and respond immediately; if you are to communicate with children, colour should become your ally (Sewani 2007). With these piano tutoring techniques and child learning concepts grasped, it is important to consider them in the subject area of this project; e-Learning.

**2.4 Electronic Learning**

Like no other training form, e-Learning promises to provide a single experience that accommodates the three distinct learning styles: auditory learners, visual learners, and kinesthetic learners (Kruse 2004). Benefits of e-Learning to the learner include:

- **On-demand availability** enables students to complete training conveniently at off-hours or from home.
- **Self-pacing** for slow or quick learners; reduces stress and increases satisfaction.
• **Interactivity** engages users, pushing them rather than pulling them through training.

• **Confidence** that quick reference materials are available reduces stress and pressure.

While many debate over whether or not e-Learning satisfies the actual educational learning experience as far as the student-teacher relationship goes, it is difficult to deny the wealth of benefits this type of teaching method provides. Even with the youngest of school age children, who are often the most open to the innovations of technology, there are hardly any educational institutions left in the world that do not use e-Learning in some capacity. As seen throughout the history of e-Learning, the benefits have only continued to grow. The cost effectiveness has proven highly beneficial even for the least economically disadvantaged venues (Higher Education Online 2007).

### 2.5 MIDI Connectivity

#### 2.5.1 MIDI Fundamentals

MIDI is an industry-standard protocol that enables electronic musical instruments such as keyboard controllers, computers, and other electronic equipment to communicate, control, and synchronise. As an electronic protocol, it is notable for its widespread adoption throughout the industry. MIDI does not transmit an audio signal; it transmits *event messages* such as Pitch and Velocity (volume), also control signals for parameters such as Vibrato and Panning. Note On, Note Off and Velocity messages are the main focus of this project. Each note has a MIDI value, middle C has a note value of 60 (Figure 10). For a full list of MIDI note values see Appendix 1 and Appendix 2.

![Figure 10 - MIDI Note Numbers](image-url)
2.5.2 MIDI to Flash Rationale

This section of the report will describe research findings in regards to potential benefits and methods of receiving MIDI input messages within an Adobe Flash environment. Connecting MIDI and Flash is a widely unexplored concept. Although largely uncharted, MIDI support in Flash is a much-desired model. So much so that various online petitions have been established with the intention to encourage Adobe to incorporate MIDI support within Flash (Adobe Flash Player Bugs and Management System, *MIDI Sound*, 2008). These petitions have thus far been in vain. For example, the 2007 petition for USB MIDI support in Flash has recently been closed (Adobe Flex Bugs and Management System, *USB Midi Device support*, 2007). These petitions seem to have huge support from the development community but little response from the Adobe team.

MIDI support in Flash would open a realm of development possibilities. With the release of the Flash Player 10 (FP10) Sound Application Programming Interface (API), developers are already beginning to create audio-based applications for online use. For example, *Hobnox Audiotool* is a state of the art rich Internet application, which aims to become a professional audio environment written in Flash. It features dynamic audio flow with cable wiring and an infinite desktop for arranging virtual audio devices. Hobnox originally streamed the sound bytes via Java to the user’s soundcard, the audio is now streamed with Flash Player using the newly released FP10 Sound API. Flash Player guarantees a much more robust audio stream and lower latency (sound delay). If users were able to interact with the application using MIDI controllers and keyboards, it could be used as a serious audio manipulation and production tool such as, *Propellerhead Reason, Ableton Live* and *Steinberg Cubase*. The benefit of using Flash as the development tool is that the application can be tailored for online use. Musicians would then be able to compose, mix, perform and record music on any online system anywhere in the world. This illustrates once more that Flash is the most suitable tool for this project, as Flash is being developed in an increasingly powerful multimedia direction. It also illustrates that Adobe are listening to the community, as the onSoundComplete Event used within the *Hobnox Audiotool* is Adobe’s answer to the *Adobe Make Some Noise* campaign (www.make-some-noise.info). Further to this release, Adobe Systems have recently issued a pre-release
version of project *Alchemy*, a small tool that compiles C and C++ code for programs running on ActionScript Virtual Machine (AVM2). The idea is to expand the capabilities of Web applications running on FP10 and Adobe AIR 1.5. The *Alchemy* project shows that Adobe understand how important it is to add more processing power to the Flash Player. Complex audio manipulations can now be achieved within Flash Player using C or C++.

MIDI control messages can be used not only to trigger audio samples but also visuals. This is an integral part of this development project. In order to fully engage a child, the application will react to the user’s input by providing instant visual and auditory feedback. The child’s interactions will trigger a variety of images and animations accompanied by sounds (music, sound FX, narrative). If the child is making progress they will receive positive feedback. If the child is not making progress they will receive reinforcing help and encouragement. A MIDI keyboard will also provide the child with an opportunity to enhance their dexterity and experiment directly with a genuine keyboard. Julia Deats, a piano tutor who specialises in teaching young children, believes that allowing children to experiment and have fun before giving them formal tuition is key; ‘It encourages creativity, they'll enjoy doing it and it should foster a kind of a natural musicianship in them.’ (Guardian 2008).

2.5.3 Possible Connectivity Methods

**FlashMidi**

In Paris, 2005, Alexis Isaac developed an API to control MIDI hardware in Flash. This API was published under the General Public License of the Free Software Foundation and is available to download including full documentation. The MIDI connection is established using an executable application created using the Microsoft Windows development environment, Delphi. The Delphi application embeds a Flash ActiveX. The Flash application then forwards MIDI commands to the Delphi application using fscommand (a Flash function which is used to pass messages to programs that host ActiveX controls).
Features

- Sends note on and note off events with velocity parameter on any of the user’s 16 MIDI ports.
- Can automate the note on / note off events by passing a duration for the note.
- Can change the volume of a channel.
- Can change the patch of a channel.
- Can load and play MIDI files.
- Receives MIDI input.

Limitations

- FlashMidi is only available on Windows.
- The user must download and install a plug-in each time the API is used.
- The Flash classes are written in ActionScript 2.0 (AS2). This project aims to establish the connection using the latest Object Oriented version of ActionScript (AS3).
- A pre-Alpha Mac plug-in for the API was created in July 2005. However, this was the final update; the API was not developed any further.

Abumarkub Midi-to-Flash

In Holland, July 2008 a Flash developer, Abumarkub published a proof of concept Flash movie online, which was able to receive MIDI data from a MIDI device. This was accomplished by using a Java Applet. The Java Applet receives the MIDI data and passes it on to JavaScript (embedded in an HTML page); which then sends it to Flash via the ExternalInterface (an API that enables communication between ActionScript and the Flash Player container).

Features

- Receives MIDI input.
- Receives not only note on / note off messages but also control messages such as velocity (volume) and pitch bend.
- The Flash classes are written in AS3.
- No plug-in is required to run the application; MIDI messages are received using a Java Applet, which runs in the background.
Limitations

- No source code or documentation available.
- Only compatible with Windows XP (not compatible with Windows Vista or Mac OS X).

Abumarkub’s solution was superior to Alexis Isaac’s earlier work in two key areas; it was written in AS3 and did not require a plug-in. The obvious hindering factor was that the code was not open-source or publicly available. However, due to demand from eager developers (such as the author of this report), in October of 2008 Abumarkub published the code for open use. The solution was therefore chosen as the foundation tool for this project.

2.6 Development Lifecycle

Due to the timescale and scope of this project, a Rapid Application Development (RAD) cycle will be used. RAD was initially a response to restrictive processes developed in the 1970s, such as the Waterfall model. The problem with previous methodologies was that applications took such a long time to build, that requirements had changed before the system was complete, often resulting in unusable systems (Developers.net 2005). Starting with the ideas of Barry Boehm and Scott Shultz, James Martin developed the Rapid Application Development approach during the 1980s at IBM and finally formalised it by publishing a book in 1991. The process involves three key areas: Requirements Gathering and Analysis, Prototyping and Iteration. RAD was created for use with projects with restricted timescales and stern deadlines. The benefits of an iterative development process are that potential bugs and usability issues are recognised during the development process rather than when the project is finished. Therefore, the issues can be addressed and resolved before it is too late. However, the RAD approach may involve compromises in functionality and performance due to the rapid nature of the process.

In terms of developing the application interface, two key design tools will be used, Storyboarding and Lo-Fi Prototyping. Storyboarding is a common technique in Human Computer Interaction and design for demonstrating system interfaces and contexts of use. Storyboards will provide design focus during the development stages. Paper prototyping is regarded as an extremely effective method of
prototyping. An article on the *User Interface Engineering* website exclaims, ‘a paper prototype is a design tool, not only a testing tool’ (Scanlon 1998). Various rough sketches will be created in order to initiate the design concepts, after each prototype has been finished an evaluation will be conducted in which each of the following will be stated: The Design Issue, Severity/Impact, User Interface Aspect, Violated Heuristic (Nielsen and Molich, 1990; Nielsen 1994) and Recommendations. These evaluations will provide a basis to progress with the development phase of the project.

### 3. Requirements Analysis

The research section of this report provided a strong overview of the user’s needs. These requirements need to be organised, rationalised and prioritised. The initial part of this section will list Functional Requirements of the application; technical aspects such as inputs, behaviours and outputs will be described. The second stage of the requirements analysis will list Non-Functional Requirements, for example: execution and evolution qualities, which will address aspects such as usability, testability and scalability. The final part of this section will prioritise each requirement according to the MoSCoW framework.

#### 3.1 Functional Requirements

Table 1- Functional Requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Functional Requirement Description</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR1</td>
<td>Allow the user to select a MIDI input.</td>
<td>The user must be able to choose which MIDI input to use in case they have more than one connected.</td>
</tr>
<tr>
<td>FR2</td>
<td>Automatically connect to a MIDI input if only one device is connected.</td>
<td>The technical aspects of configuring a MIDI device may seem confusing to a child so it will be addressed automatically.</td>
</tr>
<tr>
<td>FR3</td>
<td>Provide walkthrough e-learning tutorials.</td>
<td>The user requires the option of learning about the piano before, after or during the training programs.</td>
</tr>
<tr>
<td>FR4</td>
<td>Display coherent instructions.</td>
<td>The child requires clear, simple instructions for all games.</td>
</tr>
<tr>
<td>FR5</td>
<td>Receive MIDI input from a USB keyboard.</td>
<td>MIDI interaction is an integral part of this project. The user must be given the opportunity to interact with a tangible interface.</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FR6</td>
<td>Play audio samples (piano notes).</td>
<td>Piano samples must be triggered when a MIDI message is received or a key is pressed on the virtual keyboard.</td>
</tr>
<tr>
<td>FR7</td>
<td>Audio samples should be velocity sensitive and react to Note Off commands.</td>
<td>The samples must vary in volume and duration depending on the users interactions in order to emulate a real piano (expression sensitive).</td>
</tr>
<tr>
<td>FR8</td>
<td>Record a user’s progress and display training game results graphically.</td>
<td>The child should be given the opportunity to compare their present performance with past performances.</td>
</tr>
<tr>
<td>FR9</td>
<td>QWERTY keyboard input alternative / mouse input alternative.</td>
<td>In order to not exclude users who do not own a MIDI keyboard.</td>
</tr>
<tr>
<td>FR10</td>
<td>Feature one fully functional piano training game, which will help the child learn musical notes on a staff.</td>
<td>Learning note names, positioning and pitch, both on the keyboard and staff is a fundamental step in being able to play the piano.</td>
</tr>
<tr>
<td>FR11</td>
<td>Feature five further piano training games, which will be unlocked as the child progresses.</td>
<td>Games will teach fundamentals such as notation, timing, scales, expression and fingering.</td>
</tr>
<tr>
<td>FR12</td>
<td>Support full screen functionality.</td>
<td>Providing a full screen option will prevent the child from being distracted by the Operating System and Internet browser interfaces. This will also aid users with visual disabilities.</td>
</tr>
<tr>
<td>FR13</td>
<td>Play alternative audio samples; Organ, Xylophone, Harpsichord and various orchestral sounds.</td>
<td>Alternative sounds will provide an enriched experience and allow the child to explore other musical instruments.</td>
</tr>
</tbody>
</table>
## 3.2 Non-Functional Requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Non Functional Requirement Description</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFR1</td>
<td>Walkthrough tutorials must engage the child.</td>
<td>Children’s attention spans are limited so must be engaged using a combination of graphics, animation and audio. A linear digital story walkthrough will help to win the child’s attention and prevent errors.</td>
</tr>
<tr>
<td>NFR2</td>
<td>Provide both textual and graphical instructions.</td>
<td>This will ensure the application is accessible to users with visual impairments and will aid users with low reading skills.</td>
</tr>
<tr>
<td>NFR3</td>
<td>The application must be fully documented.</td>
<td>This will support help documentation and allow for easy maintenance of the application. It will also aid further development, which may be carried out by not only the code author but also other developers.</td>
</tr>
<tr>
<td>NFR4</td>
<td>The application must have a rapid response time and performance must be high.</td>
<td>Code and resources must be optimised to ensure a small file size and fast response time. A child will not wait for parts of the application to load over a long period of time due to their short attention spans.</td>
</tr>
<tr>
<td>NFR5</td>
<td>The published application must function within Internet Explorer, Mozilla Firefox and Safari.</td>
<td>This will ensure accessibility to as many users as possible.</td>
</tr>
<tr>
<td>NFR6</td>
<td>The application should be cross-platform compatible.</td>
<td>This will also ensure accessibility. However, due to the scope of this project it will be developed for Microsoft Windows. Mac support will be a further development recommendation.</td>
</tr>
<tr>
<td>NFR7</td>
<td>The Virtual Piano Tutor must be robust; it must continue to function despite abnormalities in input.</td>
<td>Children are likely to experiment with the application ruthlessly. Therefore, stress testing must be carried out frequently during development.</td>
</tr>
<tr>
<td>NFR8</td>
<td>The level of supportability must be adequate in terms of help documentation.</td>
<td>It is likely that schoolteachers or parents may wish to read help relating to setting up a MIDI keyboard. Help for the child will be integrated within the application using a coherent friendly approach.</td>
</tr>
<tr>
<td>NFR9</td>
<td>The Learnability of the interface must be carefully designed and First time users must not encounter difficulties achieving tasks.</td>
<td></td>
</tr>
</tbody>
</table>

27
must be carefully designed and evaluated.

<table>
<thead>
<tr>
<th>NFR10</th>
<th>The Memorability of the interface must be adequate.</th>
<th>Users must be able to re-establish proficiency after a period of not using the application.</th>
</tr>
</thead>
</table>

NFR11 Error Prevention and Recovery must be carefully integrated into the application design. Users should be prevented from making errors; e.g. the use of walkthrough wizards will control the user’s input. However, the application must be able to recover from any possible user errors.

NFR12 The application must be Satisfying to use. Which may be considered to be subjective, however several design standards has been constructed regarding satisfaction. Such as the End-User Computing Satisfaction instrument (Doll & Torkzadeh 1988) which consists of Content, Accuracy, Format, Easy of Use and Timeliness.

3.3 MoSCoW Method

Due to the importance of requirements prioritisation, this sector of the report will prioritise the user requirements using the MoSCoW method. The requirements are divided into, Must, Should, Could and Won’t.

**Must**

- **FR1** Allow the user to select a MIDI input.
- **FR2** Automatically connect to a MIDI input if only one device is connected.
- **FR5** Receive MIDI input from a USB keyboard.
- **FR6** Play audio samples (piano notes).
- **FR9** QWERTY keyboard input alternative / mouse input alternative.
- **FR10** Feature one fully functional piano training game, which will help the child learn musical notes on a staff.
- **NFR2** Provide both textual and graphical instructions.

**Should**

- **FR3** Provide walkthrough e-learning tutorials.
- **FR4** Display coherent instructions.
- **FR7** Audio samples should be velocity sensitive and react to Note Off commands.
- **FR12** Support full screen functionality.
• **NFR1** Walkthrough tutorials must engage the child.
• **NFR4** The application must have a rapid response time and performance must be high.
• **NFR5** The published application must function within Internet Explorer, Mozilla Firefox and Safari.
• **NFR7** The Virtual Piano Tutor must be robust; it must continue to function despite abnormalities in input.
• **NFR9** The *Learnability* of the interface must be carefully designed and evaluated.
• **NFR10** The *memorability* of the interface must be adequate.
• **NFR12** The application must be *satisfying* to use.

**Could**

• **NFR3** The application must be fully documented.
• **NFR8** The level of supportability must be adequate in terms of help documentation.
• **NFR11** Error Prevention and Recovery must be carefully integrated into the application design.

**Won’t (Future Developments)**

• **FR8** Record a user’s progress and display training game results graphically.
• **FR11** Feature five further piano training games, which will be unlocked as the child progresses.
• **FR13** Play alternative audio samples; Organ, Xylophone, Harpsichord and various orchestral sounds.
• **NFR6** The application should be cross-platform compatible.

4. **Design**

With a thorough list of requirements obtained, the next stage of the development process was to actually design the application. This involved sketching, wire framing and prototyping with constant referral to the requirements analysis.

4.1 **Interface Design**

This section of the report will describe the main design elements used within the application interface. Alternative ideas and design solutions will also be illustrated.
4.1.1 Characters
Children relate and respond to cartoon characters extremely well; they encounter them when they watch television, read books and play computer games. Two characters were created for use with the e-Learning application; these animated tutors provide instruction, help and tips throughout Flash Music. They also provide feedback and help to the child during the training games. The virtual tutors are based on two well-known composers: Ludwig Van Beethoven and Wolfgang Amadeus Mozart (Figure 11). Beethoven remains on the screen constantly and reacts depending on the child’s progress with the games. His mood can vary from bored, to happy, angry, disapproving or ecstatic. Mozart appears when the child opens a walkthrough tutorial; he explains fundamental concepts of piano playing in preparation for the training games.

![Beethoven & Mozart](image)

Figure 11 - Beethoven & Mozart

4.1.2 Instructions
Beethoven communicates to the child using a speech bubble. The user can read through the instructions in a linear fashion. As discussed in the KS2 Learning Characteristics section of this report: breaking the information into small pieces will
help the child maintain a firm focus. The child simply reads the first speech bubble and when they are ready, they click the more button to advance (Figure 12). Narration is presented in a friendly, informal manner, with close attention being paid to the reading skills of the younger users.

4.1.3 Help & Tips

A popup is used to provide the child with further instructions and tips relating to each game. The child is presented with a tool-tip style popup when they roll over a button (Figure 13). The Walkthrough Tutorial (Figure 14) and Progress (Figure 15) icons are also explained using the tool-tip.
4.1.4 Input Configuration

The MIDI Keyboard Setup screen allows the user to select an input if more than one device is connected to their computer (Figure 16). However, if the user only has one device connected it will be automatically selected, represented by a green tick. If the user chooses to use their character keyboard, they simply click the ‘QWERTY’ button displayed on the main interface. They are then presented with the QWERTY Keyboard Setup screen, which describes the character key mapping. The character keys are mapped to represent a piano keyboard as closely as possible (Figure 17).
4.1.5 Virtual Keyboard

The on-screen keyboard (Figure 18) responds to the user’s MIDI input. The keyboard copies the precise key presses of the user (Figure 19). The virtual keyboard can also be played using the mouse as an alternative to a MIDI keyboard. However, it is impossible to play chords using the mouse, therefore QWERTY input is also supported. When the child selects ‘Use QWERTY keyboard’ the character keys, which map to each note are displayed on the keyboard (Figure 20).
4.2 Game Design

Various training game concepts were devised. However, due to the scope of this project, only one game idea was chosen. The project research was mainly focussed on teaching children the fundamentals of a piano keyboard, note positions and sight-reading. Therefore, a game design was developed which focused on teaching children these fundamental piano playing concepts. The proposed game was entitled Keys to Score. When the user selects the game from the welcome screen they are presented with a three second countdown, which mimics existing popular e-Learning games available for the Nintendo DS console. Keys to Score involves a treble staff that
displays random notes, the child must then identify the note and play the correct piano key. If the user is stuck on a particular note, they have the option to use a colour hint that opens a colour coded keyboard and treble staff. The feature is based on the Rainbow Piano Technique developed by Annie Wang. A colour hint can be used up to three times during the game. Beethoven provides feedback depending on the child’s progress. If the user’s answer is incorrect they are given three chances before Beethoven forces them to move on. When the user answers incorrectly, Beethoven will provide various tips and clues. However, his animated state will become increasingly disapproving if the child repeatedly answers incorrectly. When the child succeeds, Beethoven becomes happier depending on the magnitude of their success. For example, the character responds by waving, jumping and even igniting fireworks. Throughout the game, the user receives auditory feedback; applause, sighs and various sound effects.

Figure 21 - Keys to Score Interface Storyboard
Figure 22 - Colour Hint Storyboard

Figure 23 - Wrong Answer Storyboard
Figure 24 - Calculating Score Storyboard

Your final score is:

93%

Figure 25 - Final Score Storyboard
4.3 Storyboards

In order to gain an understanding of the various tasks, which the user will undergo, a set of storyboards were created. The Welcome screen instructions were broken down into simple sentences (Figure 26). These instructions are displayed in a linear fashion, allowing the user to follow at their own pace.

The initial stages of the design process involved creating prototype storyboards, which illustrated various design concepts for the interface and games. Figure 29 depicts a concept for a game, which involves notes moving along a staff that require identifying by the user. The initial designs were based on an ancient manuscript to match the Beethoven and Mozart theme. On reflection, it was decided that the designs were too cluttered and dull. Due to research findings which suggested children respond well to bright colours, a new design was created which is much more
bold and vibrant (Figure 30). The final Flash Music interface design consists of a Game Panel to the left and a Help Panel to the right (Figure 30). The Help Panel is where Beethoven provides help and feedback to the child. The child can also configure the input type and choose to view the application in full-screen. The Game Panel is where the action takes place: each game or tutorial loads into the panel with the Virtual Keyboard situated at the bottom of the screen.

Figure 27 - Initial Flash Music Interface Storyboard
Figure 28 – Initial Flash Music Interface Storyboard 2

Figure 29 - Initial Flash Music Game Storyboard
Figure 30 - Flash Music Proposed Interface Design
5. Implementation & Testing

5.1 Receiving MIDI Input

5.1.1 Midi-to-Flash Version 1

In October 2008 Abumarkub published his midi-to-flash source on Google Code (http://code.google.com/p/miditoflash). It was published under the new Berkeley Software Distribution (BSD) license. The BSD license contains three major clauses allowing unlimited redistribution for any purpose as long as its copyright notices and the license disclaimers of warranty are maintained. The final clause states that neither the name of the organisation nor the names of its contributors may be used to endorse or promote products derived from the software without specific prior written permission (Nelson 2006). The code was set up to be compiled using the Flex Software Development Kit. This package release included a Java applet that received MIDI data using the Java Sound API. Therefore, the user requires the Java Plug-in, which is not quite as ubiquitous as the Flash Player. However, the Java Plug-in is free to download and relatively small in file size. The Java applet is set up to pass on the data to a JavaScript file via the AppletContext, which allows the applet to query and affect its environment. JavaScript then sends the data to Flash via an ExternalInterface call. All of the Flash classes are written in the Object Oriented AS3 format. For the purposes of this project, the code was used to create a proof of concept Flash application, which received MIDI input. The code was initially compiled using Eclipse IDE with Flex SDK installed to create a simple prototype, which listed incoming MIDI messages and triggered visuals (Appendix 11). However, due to the multimedia nature of this project the code was reconfigured so that it could be compiled using the Flash IDE (Appendix 3). The proof of concept received MIDI messages successfully, however the Abumarkub code organisation was not flexible enough to efficiently build a full application around it. After some correspondence with the source developer, Abumarkub decided to redevelop the code so that developers could use it more flexibly. The developer also intended to resolve the cross-compatibility issues.
5.1.2 Midi-to-Flash Version 2

In early February 2009, Abumarkub published a new, much more versatile version of the midi-to-flash code. The update was now compatible with Microsoft Windows XP, Vista and Linux. However the OS X compatibility issue was still in development. The MIDI Bridge is comprised of three parts:

1. A Java applet that receives the MIDI data from the user’s system.
2. JavaScript code that enables 2-way communication between Java and Flash.
3. A small Flash application that broadcasts the MIDI data to other Flash applications using the Adobe Flash class; LocalConnection.

The bridge ID can be set and the desired MIDI device can be selected. These parameters can also be set as flashvars in the containing HTML file. Multiple instances of the MIDI Bridge can be run simultaneously, each instance can be connected to a different device. Because a midi client receives midi data through a LocalConnection it can be any type of Flash application: a SWF running in a browser or in a standalone player, a projector or an AIR application. In order for a MIDI client to receive MIDI data from the MIDI Bridge, an instance of the MidiClient class must be created and passed a MIDI bridge ID. If a bridge with the same bridge ID is running, the client will automatically connect to the desired bridge at runtime:

```javascript
var bridgeId:String =
    stage.loaderInfo.parameters["midiBridgeId"];
bridgeId = bridgeId == null ? "_device1" : bridgeId;
_midiClient = new MidiClient(bridgeId,false);
_midiClient.addEventListener(MidiDataEvent.CONNECTED, handleMidiDataEvent);
_midiClient.addEventListener(MidiDataEvent.MIDI_DATA, handleMidiDataEvent);
```

Subsequently event listeners must be implemented which actually receive the MIDI data. The messages can then be used flexibly within a Flash application. See Appendix 12 for full MIDI Bridge scheme.

5.2 Flash Music

The Flash Music application was written in AS3 and compiled using the Adobe Flash IDE (Appendix 3). In order to maintain an Object Oriented Programming (OOP)
style the application is divided into a carefully organised collection of classes and packages (Figure 31). For the package structure of Flash Music see Appendix 7 and for the Package structure of the MIDI Bridge see Appendix 8. For the Flash Music online Package structure see Appendix 9. This approach allows for a structured project that can be easily maintained and developed by the author or other developers. The application is multi-tiered; a Java applet detects MIDI messages, which are received by JavaScript embedded in an HTML page. These messages are then called via the ActionScript.

5.2.1 Class Hierarchy Chart

![Flash Music Class Structure Diagram](image)

Figure 31 - Flash Music Class Structure

5.2.2 Basic Implementation (Must-Have Features)

**FR1** Allow the user to select a MIDI input.

The user is presented with a MIDI configuration pane when they click the *MIDI* interface button (Figure 16). The configuration pane displays a list of detected input
devices, the user simply clicks the desired device and they are connected. Three classes control the configuration pane:

1. **MidiConfiguration**

   MidiConfiguration is called as soon as the application is loaded, via the main Classpath ActionScript file: FlashMusic. The pane consists of a skin, a **Confirmation** button, a **Refresh** button and a list of devices. The class controls the visibility of the configuration pane skin using a true / false Boolean:
   
   ```
   _isShowing = false;
   _skin.visible = _isShowing;
   ```

   Devices are detected from the imported MIDI Bridge Abumarkub class:
   
   ```
   import net.abumarkub.bridge.MidiClient;
   ```

   If no devices are detected the user is presented with a message:
   
   ```
   _content.htmlText = "<font color='#CC0000'>No available Midi Keyboard detected!</font> Connect a Midi keyboard and press 'refresh'. Otherwise use the mouse to play or select QWERTY to use the QWERTY keyboard.";
   ```

   If one or more devices are detected, the message reads: ‘These are the MIDI devices connected to your computer.’ followed by a list of input devices, other devices such as outputs, sequencers and synthesisers are filtered out within the Java application. If the list of devices was longer than two, the pane exceeded the length of the background skin and the **Confirm** and **Refresh** buttons were obscured by the list. Positioning the buttons and scaling the skin relative to the list solved this problem:
   
   ```
   _bg.height = _pane.height + _scrollPane.y + 40;
   _scrollPane.height = _pane.height;
   _confirmBtn.y = _refreshBtn.y = _pane.height + _scrollPane.y + 10;
   ```

2. **MidiConfigurationPane**

   This class receives the MIDI device data and creates an items object that handles the data. In this function the device data is received from the MidiDeviceData class. The data ID is then added to the items object and the vertical position is defined:
   
   ```
   public function createItem(data:MidiDeviceData):void
   {
   ```
var item:MidiConfigurationPanelItem = new MidiConfigurationPanelItem(data);
item.y = _numItems * (item.height + _margin);
addChild(item);
_items[data.id] = item;
_numItems++;
}

3. MidiConfigurationPanelItem

This class handles the device list items. Each list item is assigned an Event Listener that detects Mouse Events.

    _skin.addEventListener(MouseEvent.MOUSE_OVER,handleOver);
    _skin.addEventListener(MouseEvent.MOUSE_OUT,handleOut);
    _skin.addEventListener(MouseEvent.MOUSE_DOWN,handleDown);

A ColorTransform is used to highlight an active list item:

    public function handleOver(e:MouseEvent = null):void {
      var ctr:ColorTransform = _skin.bg.transform.colorTransform;
      ctr.color = 0x84EB84;
      _skin.bg.transform.colorTransform = ctr;
    }

When the user selects an item and a successful connection is established, a tick is placed next to the device.

FR2 Automatically connect to a MIDI input if only one device is connected.
This feature was added to the Abumarkub MIDI Bridge update. The foremost-detected MIDI input device is always connected as soon as the application loads. Therefore, if only one device is detected, it will be connected automatically.

FR5 Receive MIDI input from a USB keyboard.
Flash Music is able to receive MIDI control messages; Note On, Note Off and Velocity using the MIDI Bridge.

FR6 Play audio samples (Piano notes).
The SoundGenerator and KeySound classes handle sample playback. The application Classpath, FlashMusic initiates the processMidiData function, which handles
incoming MIDI messages. Note On and Note Off messages are used to control the virtual keyboard, the current game and the piano key samples:

```javascript
private function processMidiData(data:IMidiData):void
{
    switch(data.command)
    {
    case MidiCommands.NOTE_ON:
    case MidiCommands.NOTE_OFF:
        if(_midiKeyboard.visible)
        {
            _midiKeyboard.pressKey(data);
        }
        _soundGenerator.processMidiData(data);
        if(_currentGame == null)
        {
            break;
        }
        _currentGame.processMidiData(data);
        break;
    }
}
```

Each note on the piano has been sampled and edited using audio editing software: Cakewalk Sonar and Adobe Audition. The Cakewalk Sonar *Virtual Studio Technology* (VST) was used to generate each note using a Grand Piano voice. These samples were then cropped and equalised using Audition. This technique allows for high-quality genuine grand piano sounds to be generated by the user. The sampled technique is also very versatile as any sound samples can be triggered; from piano sounds to animal sounds, which can be useful when developing engaging games for children. The limitation of this technique is that the more sound samples are used, the larger the file size. Another option would be to have the application connect to the user’s internal General MIDI (GM) sound generator, which has a built in collection of piano synthesised sounds. This could have been implemented using Java. The advantage of this technique is that it eliminates the need for bandwidth-intensive samples. The limitation is the quantity and quality of the sounds available. Although there are 127 voices to choose from, the voices cannot be edited and no more can be added. The sounds are synthesised and therefore do not match the authentic tones of a piano.
- **SoundGenerator**
  The SoundGenerator class controls the playback of the piano samples and maps each sample to the corresponding MIDI note number. The two octaves used within Flash Music range from C4 to B5 (Appendix 2). The MIDI number 48 represents the C4 note and as the notes progress, so do the MIDI numbers. Therefore, the MIDI number 48 is assigned the C4 sample; the MIDI number 49 is assigned to the note D flat 4 and so on (Figure 10):

  ```javascript
  public function SoundGenerator()
  {
    _allNotes = {"48":new C4(),"49":new Db4(),"50":new D4(),"51":new Eb4(),"52":new E4(),"53":new F4(),"54":new Fs4(),"55":new G4(),"56":new Gs4(),"57":new A4(),"58":new Bb4(),"59":new B4(),"60":new C5(),"61":new Db5(),"62":new D5(),"63":new Eb5(),"64":new E5(),"65":new F5(),"66":new Fs5(),"67":new G5(),"68":new Gs5(),"69":new A5(),"70":new Bb5(),"71":new B5()};
    _allSounds = new Object();
  }
  ```

**Fr9** QWERTY keyboard input alternative / Mouse input alternative.
Mouse input is handled using EvenListeners, when the user is in MIDI mode they can click each key to trigger a sound or interact with a game. However, they cannot play more than one note using the mouse, therefore chords are impossible. For this reason, a QWERTY keyboard input alternative was implemented. As with the SoundGenerator class, the MIDI note numbers are mapped to the relevant character key codes within the QwertyToMidiMapping class, for example:

  ```javascript
  _mapping[KeyMappings.KEY_TAB] = 48;
  _mapping[KeyMappings.KEY_1] = 49;
  ```

Essentially, when a character key is pressed, the key code is used to emulate a MIDI event. The initial approach to key mapping did not represent the layout of a real piano keyboard (Figure 32). It was difficult to play chords and would potentially confuse the user. Therefore, the mapping was re-designed to resemble a genuine keyboard (Figure 33).
FR10 Feature one fully functional piano training game, which will help the child learn musical notes on a staff.

The Keys to Score game was a major part of the implementation of this project. The game is based around the treble staff. A MovieClip was created which contains 14 frames, each with the note displayed in a different position on the staff. The keyframes are mapped to the corresponding MIDI notes:

```ActionScript
private const _midiMapping:Object = {
```

A random note is displayed when the game initially loads and each time an answer is given. A random number is generated between 1 and 14, the treble staff MovieClip then goes to and stops at a random frame:

```ActionScript
_trebleStaff.gotoAndStop(Math.floor(Math.random()*14+1));
```

The user then plays the key, which they think corresponds to the note displayed on the staff. The answer is then checked; by comparing the current MovieClip frame to the received MIDI number and a Boolean is set to true or false:

```ActionScript
var correct:Boolean =
(midiMapping[_trebleStaff.currentFrame] == _midiNote);
```

If the answer is correct, an applause sound effect is played, Beethoven gives approval and reveals the note name and the correct answers total is calculated and displayed.
on-screen. The player’s score determines Beethoven’s ‘mood’, which triggers a different animated state. The score count increments and the graphical note representation become animated. A short timer is then triggered, which is followed by a new note:

```
if(correct)
{
    _applauseChannel = _applauseSound.play();
    _message = "Correct answer! The note is " +
               _trebleStaff.currentLabel + "!";
    _messageType = FeedbackIds.CORRECT_ANSWER;
    _correctText.text = "Correct " + ++_correct;
    switch(true)
    {
        case _correct >= 10:
            _mood = MoodIds.HAPPY_3;
            break;
        case _correct >= 5:
            _mood = MoodIds.HAPPY_2;
            break;
        case _correct < 5:
            _mood = MoodIds.STAND;
            break;
    }
    _total++;  
    _trebleStaff.note.gotoAndPlay(81);
    _nextQuestionTimer.start();
}
```

During the development, a problem with the scoring system occurred; if the user answered correctly they could simply keep pressing the key and achieve fantastic scores. Also, both Note On and Note Off messages were being detected which incremented or decremented the score by two each time. Creating a processMidiData function, which only received Note On messages and blocked incoming messages whilst the answer was being processed, solved this problem:

```
public function processMidiData(data:IMidiData):void
{
    if(_processingAnswer || data.data2 == 0){return;}
    _processingAnswer = true;
    _midiNote = data.data1;
    checkAnswer();
}
```
When the user has identified 15 notes, Beethoven ‘calculates’ their percentage score during a drum roll:

```actionscript
private function calculateEnd():void
{
    _message = "GAME OVER!\nCalculating your score.. ";
    _endChannel = _endSound.play();
    _mood = MoodIds.STAND;
    _messageType = FeedbackIds.CORRECT_ANSWER;
    dispatchEvent(new Event(EventIds.MSG_CHANGE));
    dispatchEvent(new Event(EventIds.MOOD_CHANGE));
    _calculateTimer.start();
}
```

After the calculation timer has finished the child’s score is displayed and Beethoven reacts / provides feedback depending on the user’s performance:

```actionscript
_percentScore=Math.floor(_correct/_maxNumQuestions*100);
```

### 5.2.3 Extensions

**FR7** Audio samples should be velocity sensitive and react to Note Off commands.

In order for Flash Music to match the sounds of a genuine piano as closely as possible, it was important for the application to recognise not only when a user presses a key, but also when they release the key. It is also important to detect the pressure (velocity) at which the key was pressed. Without these two features it would be impossible for the child to play with expression.

- **KeySound**
  
  The KeySound class receives the velocity as a MIDI message and plays the audio sample with a relative volume:

  ```actionscript
  public function KeySound(data:IMidiData,sound:Sound)
  {
      var velo:Number=Math.round(data.data2/127*100)/100;
      velo = velo > 1 ? 1 : velo;
      _soundTransform = new SoundTransform(velo);
      _soundChannel = sound.play(0,0,_soundTransform);
  }
  ```

  This function receives the key velocity (data2). Velocity ranges from 0 to 127, whereas the sample volume ranges from 0.0 to 1.0. Therefore, 127 is multiplied by 100 and rounded up. This number is then divided by 100, so that it is relative to the sample volume scale. A soundTransform is then used
to control the volume of the sample. Note Off messages are used to stop the sample playback. However, when a sample is stopped it produces an audible distorted click sound. This is a common sample playback issue within Flash. In order to solve this issue, a Tween was used to fade the sample out rather than an immediate stop, which is also how a real piano functions:

``` ActionScript
Tween(this,"fadeSound",None.easeOut,_soundTransform.volume,0.05,_fadeTime,true);
```

This line of code produces a short fade out when a Note Off message is received.

**FR12 Support full screen functionality.**

Due to high demand from websites such as *YouTube*, which make use of Flash Video, the ability to view a Flash Movie in full screen mode was provided by Adobe with the release of Flash Player 9. The Flash Player display state now has two modes:

``` ActionScript
stage.displayState = "fullScreen";
stage.displayState = "normal";
```

One problem encountered with the full screen feature is that keyboard input is disabled when full screen mode is active. Therefore, the QWERTY input mode is disabled when full screen is active. This is a deliberate security feature implemented by Adobe: ‘All keyboard input and key-related ActionScript is disabled while in full-screen mode, with the exception of the keyboard shortcuts that take the viewer out of full-screen mode.’ (Stampfli 2006). Unfortunately there are no work-arounds available at present. However, there is evidence to suggest that this issue will be resolved in the future, Tracy Stampfli of Adobe states that; ‘we hope to address these limitations in future versions of Flash Player as we continue to develop this feature.’

### 5.3 Heuristic Evaluation

In order to obtain an expert-based evaluation of the Piano Tutor application, a Heuristic evaluation was carried out. Heuristic evaluation is a usability engineering method, which reveals negative aspects of usability within a user interface design (Nielsen and Molich, 1990; Nielsen 1994). A range of user tasks were performed in order to identify potential usability problems. The interface was then examined several times, and various aspects of the system were analysed in relation to relevant Heuristics and Guidelines. Each problem was discussed in detail; the severity and impact on the user was then described followed by recommendations for future
development. The evaluation made use of Jakob Nielson’s well-established Ten Heuristic Evaluation guidelines. However, other expert based Heuristic guidelines are available, including: Donald Norman’s Guidelines, Ben Shneiderman’s 8 Golden Rules and Tognazzini’s Principles. Nielson’s Usability Problem Severity Rating method was used to measure and prioritise the issues:

0 - Not a usability problem at all
1 - Cosmetic problem only: need not be fixed unless extra time is available on project
2 - Minor usability problem: fixing this should be given low priority
3 - Major usability problem: important to fix, so should be given high priority
4 - Usability catastrophe: imperative to fix this before product can be released

Table 3- Heuristic Evaluation

<table>
<thead>
<tr>
<th>Issue</th>
<th>Heuristic Violated</th>
<th>Severity &amp; Impact</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>The full-screen icon is not clearly displayed</td>
<td>Visibility</td>
<td>2 The user may not know the feature exists</td>
<td>Clearer labelling, or a tool-tip would solve this problem</td>
</tr>
<tr>
<td>No back button included with the application instructions (speech bubble)</td>
<td>Flexibility and efficiency of use</td>
<td>3 The user may need to revert back to a previous instruction</td>
<td>Include a back button</td>
</tr>
<tr>
<td>The MIDI configuration panel obscures the QWERTY panel when it is open.</td>
<td>Visibility / Consistency and standards</td>
<td>4 May confuse the user</td>
<td>Force the application to only display one panel at a time</td>
</tr>
<tr>
<td>The QWERTY panel is obscured by the Keys to Score game treble staff when the game is active</td>
<td>Visibility / Consistency and standards</td>
<td>4 QWERTY instructions are obscured</td>
<td>Force the QWERTY panel to always be on the foremost layer</td>
</tr>
<tr>
<td>The Colour Hint button is not clearly displayed</td>
<td>Visibility</td>
<td>3 The feature can be easily missed</td>
<td>The button should be more prominent and colourful</td>
</tr>
<tr>
<td>The welcome page may be distracting from the instructions.</td>
<td>Consistency and standards / Help and documentation</td>
<td>2 The user may not notice the instructions</td>
<td>The initial instructions should consume the whole screen</td>
</tr>
<tr>
<td>The MIDI configuration panel does not refresh automatically when a device is removed or</td>
<td>Visibility of system status</td>
<td>3 The user may assume the device is not recognised</td>
<td>The application should refresh the list automatically</td>
</tr>
<tr>
<td>Issue</td>
<td>Heuristic Violated</td>
<td>Severity &amp; Impact</td>
<td>Recommendation</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>--------------------</td>
<td>-------------------</td>
<td>----------------------------------------------------</td>
</tr>
<tr>
<td>The user cannot skip a question they do not know within the Keys to Score game</td>
<td>User control and freedom</td>
<td>2 May cause frustration when practising</td>
<td>Create a practice mode for the game with a ‘Skip Note’ option</td>
</tr>
<tr>
<td>The note name is not displayed prominently within the Keys to Score game</td>
<td>Visibility of system status</td>
<td>3 The user may not notice when the note name is revealed</td>
<td>Display the note name in large font on the main game screen</td>
</tr>
<tr>
<td>No note naming help is available during game play</td>
<td>User control and freedom / Help and documentation</td>
<td>4 The user cannot access help with identifying notes whilst playing a game</td>
<td>Note position memory aids should be provided if the user requires extra help</td>
</tr>
<tr>
<td>The Beethoven character causes distraction</td>
<td>Error prevention / Consistency and standards</td>
<td>2 May distract the user from the training game</td>
<td>Stop or curb the character animation between each answer</td>
</tr>
<tr>
<td>The Keys to Score game countdown introduction gives the impression of the game being timed</td>
<td>Match between system and the real world</td>
<td>4 May cause the user to rush through the training</td>
<td>Provide a timed game mode and a non timed practice mode</td>
</tr>
</tbody>
</table>

### 5.5 User Testing

The HCI Handbook states; ‘there simply is no substitute for testing proposed solutions with real users’ (*The Human-computer Interaction Handbook*, Julie A. Jacko, Andrew Sears, 2003). End-user testing is of vital importance when conducting a usability evaluation. The great advantage of empirical end-user testing is that the results are incontrovertible. Unlike Heuristic Evaluation, where HCI experts speculate as to what may cause users difficulties, an end-user test highlights where users actually do have difficulties (Tscheligi 2000). It remains up to the tester, however, to interpret the results and determine what caused the problems. In carrying out the user testing of the application there were several key factors to take into consideration:

- The manner of questions to be asked during and after the test sessions must be carefully planned, in order to understand exactly what the user is going through.
The level of interaction between the user and evaluator must be monitored during the testing; the user should be prompted when necessary or it is thought to be beneficial to the evaluation.

Body language and general behaviour should not be intimidating; the user must feel relaxed and comfortable. The users on screen actions, body language, facial expressions and tone of voice will be monitored during the test sessions. This will provide a greater understanding of the user’s perceived negative and positive aspects of the application.

The Keys to Score game was tested on two participants: one male user from the final stages of KS2 and one female from the lowest age bracket. The users were given three attempts at the game. Intervention was kept to a minimum in order to relieve any pressure on the children. Infrequent prompting also minimises the Hawthorne Effect, which causes users to alter their behaviour when they feel they are being monitored closely.

5.5.1 User Test Results (User 1)

Date: 28th Feb 2009

Participant Name: Lucas
Gender: Male
Participant Age: 11

Table 4 - Test Results (User 1)

<table>
<thead>
<tr>
<th>Attempt</th>
<th>Correct /15</th>
<th>Incorrect /15</th>
<th>Score (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14</td>
<td>1</td>
<td>93</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>2</td>
<td>86</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>0</td>
<td>100</td>
</tr>
</tbody>
</table>

Test Session Summary:
Lucas found the game fun, but a little confusing at first until he got used to the rules. He was initially puzzled by the higher and lower octaves. He rushed it a little due to overconfidence, hence the mistakes. Lucas was very keen to keep having more attempts in order to try and beat his high score. I felt that he would need more of a challenge to keep him interested for longer. For example, a timed game where he could try and beat his best time, and something that played a familiar tune or melody.
Lucas actually thought at one point that if he answered each question fast enough, the game would play a song, which caused him to rush some of the answers.

![Image of a child using a computer for a music game](img)

Figure 34 - Test Session One

5.5.2 User Test Results (User 2)

**Date:** 28<sup>th</sup> Feb 2009  
**Participant Name:** Stephanie  
**Gender:** Female  
**Participant Age:** 7

<table>
<thead>
<tr>
<th>Attempt</th>
<th>Correct /15</th>
<th>Incorrect /15</th>
<th>Score (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>3</td>
<td>80</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>2</td>
<td>86</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1</td>
<td>93</td>
</tr>
</tbody>
</table>

**Test Session Summary:**

Stephanie took a lot longer to complete the game than Lucas; she was less confident and got very confused. She requested that I wrote out FACE and EGBDF on a piece of paper for her to refer to visually (Figure 36), which she relied heavily on for help (she could not have completed the task without it, and she may have become bored because it was too difficult; possible screen aid option). She needed a lot of extra
help, but began to get the hang of it in the end. For example, she did not discover the *colour hint* until her final attempt. Stephanie had to count up the keys alphabetically each time by herself, but began to remember them, and started to insist on no help after her second attempt. She also became confused about which A or C was on the left octave or right octave. She was slightly distracted by the character jumping in the help panel and asked why he did not have a mouth, but she noted that his celebrations were very nice. Stephanie got bored and tired of the game much faster than Lucas.

![Figure 35 - Test Session Two](image1)

![Figure 36 - Aid Sheet For The Younger User](image2)

### 5.5.3 Feedback Summary

The initial countdown led the children to believe that the game had a time limit, or that they were being timed, perhaps this would be a good option for older or more able and experienced children. The younger user struggled slightly, a simple opening tutorial could help as an option for those completely new to reading music. Overall,
the children described the application as ‘really fun’ and after their second attempt, they began to remember the keys and note positions quite intuitively.

7. Conclusion & Future Work

7.1 Conclusion

7.1.1 Flash Music

The Flash Music application acts as a demonstration of what can be achieved by combining a Flash environment, with a tangible input device such as a MIDI keyboard. Although applications exist which are geared towards teaching a child to play the piano, the instrument cannot be learned without using an actual keyboard. This is where Flash Music differs from existing applications; it is a stimulating educational environment, which allows the user to experiment on a genuine keyboard and receive feedback from the application. The user can use low-cost portable devices (Figure 37) and eventually transfer the skills they learn onto a grand piano.

![Accessible MIDI Keyboards](image)

Figure 37 - Accessible MIDI Keyboards

The user may even interact with Flash Music using an authentic digital grand piano (Figure 38). The application is however limited in functionality. Flash Music requires substantial further development in order for it to be an effective learning tool. The developed application teaches basic keyboard and note position concepts.
Although these learning models are important, they only represent a small part of learning to play the piano. Therefore, the objective: to design and develop an online interactive piano tutoring application with MIDI connectivity, has not been fully met. However, the fundamentals are firmly in place for future development of the application.

Figure 38 - Suzuki HG-425e Digital Grand Piano

7.1.2 MIDI in Flash

An integral objective of this project was to provide the user with the possibility to use a piano keyboard as an input device. Thorough research and experimentation with MIDI to Flash connectivity revealed an elegant solution and the goal was accomplished. A MIDI keyboard can be connected via a USB cable, MIDI lead or MIDI adaptor and messages will be received within the Flash environment. The final objective of this report was to evaluate the compatibility and usefulness of linking MIDI and Flash. The value and the future of Flash MIDI support will now be concluded. Adobe Flash and the Flash Player are best known for three key features:

1. **Lightweight**: from the Flash Player itself to the SWF file format to the assets it is optimised to load, focus has always been placed on small file sizes.
2. **Standards**: the Flash Player supports many web and multimedia format standards in the industry, such as JPG, MP3 and XML.
3. **Interactivity**: Flash Players greatest strength is the dynamic behaviour through ActionScript 3.0 to allow user interactivity.

MIDI fits all of these categories perfectly; it is extremely small, being the vector of music. It is a standard music format - many will argue it is the only standard. MIDI is flexible and interactive, not only in terms of input but also output; soundcards or additional software can transform General MIDI into rich orchestrations. However, many developers and campaigners hold the opinion that Adobe will never implement MIDI support within the Flash Player. The reasoning behind this is that, where the Flash Player stands for ubiquity and common experience, MIDI is inconsistent between platform and device. John Dowdell of the San Francisco Adobe Developer Team states that ‘the problem on desktops has been when you get varying capabilities on various platforms, so that the same MIDI file can sound different on different machines’ (Who Wants MIDI In The Flash Player Blog 2005). These judgements mainly relate to MIDI output, whereas input has been the concern of this project. In terms of output, the Flash team are moving towards dynamic sound generation, which has been implemented within the latest Flash Player. Apart from the obvious educational benefits, MIDI input combined with dynamic sound generation will allow for rich Internet audio manipulation tools to be created using Flash. Designers, Programmers, Animators and Artists use Flash; having Musicians on board would definitely render Flash a universal tool.

### 7.2 Future Work

Flash Music has the potential to be used as a successful educational tool, although at present it merely demonstrates the power of e-Learning combined with tangible interaction. The application requires development in the following key areas:

#### 7.2.1 Further Training Games

User testing highlighted some key areas, which require addressing: the existing *Keys to Score* game should incorporate timed challenges, further clefs (Bass, Alto, Tenor) and difficulty settings, due to the varying abilities of KS2 children. Timed challenges would be a simple implementation, but may have a high impact on the application. A child will often strive to beat their personal best and timed challenges will encourage the user to return to the application on a regular basis. In order to allow children to
progress with their musical knowledge and piano playing skills, further training games need to be developed. These games should address concepts discussed in section 2.3 Piano Tutoring, which include; The Keyboard, Reading Music, Rhythm, Sharps & Flats, The Treble Clef, The Bass Clef, Tone & Pitch, Scales & Arpeggios, Chords & Harmony, Fingering and Expression. Besides adding supplementary games to Flash Music, a platform could be created which would allow third parties to add their own games and training modules. This would create a rich experience, which would include the involvement of educators and educational institutions.

Following the additional developments, the application will require further testing and evaluation. The project is largely user orientated, therefore, additional user testing will be required. When developing additional training games, requirements gathering techniques used should be expanded. Creating a user archetype (Appendix 10) was the initial stage of the project design process, it was also one of the most crucial and frequently referred to elements of the project. It not only provided an initial foundation for design concepts but also allowed a closer relationship with potential users of the piano-trainer to be obtained. Creating a larger collection of personas may have helped to an even greater extent. However, persona expert, Kim Goodwin claims that ‘a large set of personas is problematic; the personas all blur together.’ (Goodwin 2001). So it seems that it is still important to keep the scope narrow and focused.

7.2.2 Walkthroughs
The Requirements Prioritisation section of this report stated that the system should provide walkthrough e-Learning tutorials. This requirement was not met to the fullest extent. Tips and hints are provided throughout the application, however the idea was for each game to feature an introductory walkthrough. The walkthrough would run in the style of a classroom tutorial presented by the Mozart character. User testing revealed the importance of pre-game walkthroughs; younger users required mnemonic devices to help with note positioning (Appendix 6). The application instructions and walkthrough tutorial modules should also be made available in additional languages in order to reach a wider audience.
7.2.3 User Progress

The Requirements section also noted that the system would benefit from the ability to record a user’s progress and display training game results graphically. Research into KS2 learning characteristics exposed the fact that children respond well to comparisons of current and previous progress. They do not, however, respond well to direct comparisons to their peers. The application should allow user’s to log-in before they begin, the child can then play each game and their progress (scores, completion times and accuracy) will be stored in a database. XML may be a suitable method of storing these results as Flash supports XML reading and writing comprehensively. The child will then be provided with the opportunity to view their scores graphically as simple charts and graphs. This cycle of train-reflect-train follows that of existing popular e-Learning games available, such as Dr Kawashima’s Brain Training available for the Nintendo DS console.

7.2.4 Additional Sound Generators

Functional Requirement 13 suggested as a future development the system could support playback of alternative audio samples. This would allow children to explore other instruments, which would be achieved by incorporating further sample sets into the application. However, this is likely to increase the application file size substantially. An approach would be to load the sounds externally when they are requested. Further development of the newly implemented Adobe Dynamic Sound Generation feature may even be used as a lightweight alternative in future.

7.2.5 OS X Support

Functional Requirement 14 states that the application should be platform independent. Without MIDI, Flash Music can be used on any system within any Internet browser. Throughout the development of Flash Music, MIDI input was supported on Windows XP, Windows Vista and Linux. However, Apple’s operating system (OS X) was not supported. Karen Haslam of Macworld UK reported in 2006, that Apple was the number one supplier to the Western European education market, pushing Dell into second place. The share of Mac’s in the US educational market is also substantial. Therefore, Mac support is a key issue. During the final weeks of this project, the MIDI Bridge code developer revealed an update to the Java applet. This update to the Java code fixes the OS X compatibility issue. However, since the release of Mac
OSX version 10.4.8 Apple no longer support the *com.apple.audio.midi* Java package, which is essential to the MIDI Bridge applet. This package allows Java applications to easily access the user’s *CoreMIDI* system. Mac OS X Universal Binary Java Midi Subsystem (mmj) is a free solution to this problem created by a collection of developers, Humatic Hypermedia. Mandolane Limited have also created a similar package. However, in order for mmj to function it must be downloaded and placed in the Java Extensions directory. Mandolane also requires downloading and installing. Because the Humatic applet is loading the mmj library from the user’s local system, the applet has to be signed and trusted before it can run. This creates obvious security issues and is likely to cause confusion considering the target audience of Flash Music. However, this is not an entirely uncommon procedure, for example the popular online gaming environment *Runescape* is a signed applet, which does not seem to be an obstacle in view of the game’s huge success. Mac support is an important feature to develop and these are the initial steps towards creating a full-bodied solution.

### 7.3 Summary

The goal of this project was to provide all children with an equal opportunity to learn the basics of the piano, though the medium of e-Learning. Flash Music is designed to not only teach piano, but also to help children gain an enthusiasm for music. Although many will argue that an e-Learning application is no compromise for a real teacher, Flash Music can certainly be used in conjunction or as a precursor to piano lessons. In future, Flash Music or similar applications with MIDI connectivity may even be used as a substitute to schooling from a piano instructor; relieving parents of the heavy burden of piano lesson fees. This concept may even be used to help children learn other instruments, with the use of alternative MIDI devices such as Guitars and Drum Pads.

THE FLASH MUSIC DEMONSTRATION APPLICATION IS AVAILABLE FROM:

www.flash-music.co.uk
8. References

8.1 Print

Books


Journal Articles


8.2 Electronic

Websites


Association Board of the Royal Schools of Music [online], Online Games, Available from: http://www.abrsm.org/?page=students/games [Accessed 20 January 2009]


Piano Is Easy, Piano by Number, Available at: http://pianoiseasy2.com [Accessed 10 January 2009]


9. Appendices

**Appendix List**

Appendix 1 - MIDI Values Chart (etechdocs.com)
Appendix 2 - MIDI Keyboard Values (J. Wolfe, UNSW)
Appendix 3 - Adobe Flash IDE
Appendix 4 - Note Break Down Diagram (quamut.com)
Appendix 5 - Notes on a Keyboard (quamut.com)
Appendix 6 - Mnemonic Device / Memory Aid (quamut.com)
Appendix 7 - Flash Music Package Structure
Appendix 8 - Abumarkub Package Structure
Appendix 9 - Flash Music Online Folder Structure
Appendix 10 - Persona
Appendix 11 - MIDI to Flash Proof of Concept
Appendix 12 - MIDI Bridge Scheme
<table>
<thead>
<tr>
<th>Note No.</th>
<th>Name</th>
<th>Note No.</th>
<th>Name</th>
<th>Note No.</th>
<th>Name</th>
<th>Note No.</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C-2</td>
<td>32</td>
<td>G#0</td>
<td>64</td>
<td>E-3</td>
<td>96</td>
<td>C-6</td>
</tr>
<tr>
<td>1</td>
<td>C#-2</td>
<td>33</td>
<td>A-0</td>
<td>65</td>
<td>F-3</td>
<td>97</td>
<td>G#-6</td>
</tr>
<tr>
<td>2</td>
<td>D-2</td>
<td>34</td>
<td>A#0</td>
<td>66</td>
<td>F#3</td>
<td>98</td>
<td>D-6</td>
</tr>
<tr>
<td>3</td>
<td>D#-2</td>
<td>35</td>
<td>B-0</td>
<td>67</td>
<td>G-3</td>
<td>99</td>
<td>D#-6</td>
</tr>
<tr>
<td>4</td>
<td>E-2</td>
<td>36</td>
<td>C-1</td>
<td>68</td>
<td>G#3</td>
<td>100</td>
<td>E-6</td>
</tr>
<tr>
<td>5</td>
<td>F-2</td>
<td>37</td>
<td>C#1</td>
<td>69</td>
<td>A-3</td>
<td>101</td>
<td>F-6</td>
</tr>
<tr>
<td>6</td>
<td>F#-2</td>
<td>38</td>
<td>D-1</td>
<td>70</td>
<td>A#3</td>
<td>102</td>
<td>F#-6</td>
</tr>
<tr>
<td>7</td>
<td>G-2</td>
<td>39</td>
<td>D#1</td>
<td>71</td>
<td>B-3</td>
<td>103</td>
<td>G-6</td>
</tr>
<tr>
<td>8</td>
<td>G#-2</td>
<td>40</td>
<td>E-1</td>
<td>72</td>
<td>C-4</td>
<td>104</td>
<td>G#-6</td>
</tr>
<tr>
<td>9</td>
<td>A-2</td>
<td>41</td>
<td>F-1</td>
<td>73</td>
<td>C#4</td>
<td>105</td>
<td>A-6</td>
</tr>
<tr>
<td>10</td>
<td>A#-2</td>
<td>42</td>
<td>F#1</td>
<td>74</td>
<td>D-4</td>
<td>106</td>
<td>A#-6</td>
</tr>
<tr>
<td>11</td>
<td>B-2</td>
<td>43</td>
<td>G-1</td>
<td>75</td>
<td>D#4</td>
<td>107</td>
<td>B-6</td>
</tr>
<tr>
<td>12</td>
<td>C-1</td>
<td>44</td>
<td>G#1</td>
<td>76</td>
<td>E-4</td>
<td>108</td>
<td>C-7</td>
</tr>
<tr>
<td>13</td>
<td>C#-1</td>
<td>45</td>
<td>A-1</td>
<td>77</td>
<td>F-4</td>
<td>109</td>
<td>C#-7</td>
</tr>
<tr>
<td>14</td>
<td>D-1</td>
<td>46</td>
<td>A#1</td>
<td>78</td>
<td>F#4</td>
<td>110</td>
<td>D-7</td>
</tr>
<tr>
<td>15</td>
<td>D#-1</td>
<td>47</td>
<td>B-1</td>
<td>79</td>
<td>G-4</td>
<td>111</td>
<td>D#-7</td>
</tr>
<tr>
<td>16</td>
<td>E-1</td>
<td>48</td>
<td>C-2</td>
<td>80</td>
<td>G#4</td>
<td>112</td>
<td>E-7</td>
</tr>
<tr>
<td>17</td>
<td>F-1</td>
<td>49</td>
<td>C#2</td>
<td>81</td>
<td>A-4</td>
<td>113</td>
<td>F-7</td>
</tr>
<tr>
<td>18</td>
<td>F#-1</td>
<td>50</td>
<td>D-2</td>
<td>82</td>
<td>A#4</td>
<td>114</td>
<td>F#-7</td>
</tr>
<tr>
<td>19</td>
<td>G-1</td>
<td>51</td>
<td>D#2</td>
<td>83</td>
<td>B-4</td>
<td>115</td>
<td>G-7</td>
</tr>
<tr>
<td>20</td>
<td>G#-1</td>
<td>52</td>
<td>E-2</td>
<td>84</td>
<td>C-5</td>
<td>116</td>
<td>G#-7</td>
</tr>
<tr>
<td>21</td>
<td>A-1</td>
<td>53</td>
<td>F-2</td>
<td>85</td>
<td>C#5</td>
<td>117</td>
<td>A-7</td>
</tr>
<tr>
<td>22</td>
<td>A#-1</td>
<td>54</td>
<td>F#2</td>
<td>86</td>
<td>D-5</td>
<td>118</td>
<td>A#-7</td>
</tr>
<tr>
<td>23</td>
<td>B-1</td>
<td>55</td>
<td>G-2</td>
<td>87</td>
<td>D#5</td>
<td>119</td>
<td>B-7</td>
</tr>
<tr>
<td>24</td>
<td>C-0</td>
<td>56</td>
<td>G#2</td>
<td>88</td>
<td>E-5</td>
<td>120</td>
<td>C-8</td>
</tr>
<tr>
<td>25</td>
<td>C#0</td>
<td>57</td>
<td>A-2</td>
<td>89</td>
<td>F-5</td>
<td>121</td>
<td>C#-8</td>
</tr>
<tr>
<td>26</td>
<td>D-0</td>
<td>58</td>
<td>A#2</td>
<td>90</td>
<td>F#5</td>
<td>122</td>
<td>D-8</td>
</tr>
<tr>
<td>27</td>
<td>D#-0</td>
<td>59</td>
<td>B-2</td>
<td>91</td>
<td>G-5</td>
<td>123</td>
<td>D#-8</td>
</tr>
<tr>
<td>28</td>
<td>E-0</td>
<td>60</td>
<td>C-3</td>
<td>92</td>
<td>G#5</td>
<td>124</td>
<td>E-8</td>
</tr>
<tr>
<td>29</td>
<td>F-0</td>
<td>61</td>
<td>C#3</td>
<td>93</td>
<td>A-5</td>
<td>125</td>
<td>F-8</td>
</tr>
<tr>
<td>30</td>
<td>F#-0</td>
<td>62</td>
<td>D-3</td>
<td>94</td>
<td>A#5</td>
<td>126</td>
<td>F#-8</td>
</tr>
<tr>
<td>31</td>
<td>G-0</td>
<td>63</td>
<td>D#3</td>
<td>95</td>
<td>B-5</td>
<td>127</td>
<td>G-8</td>
</tr>
</tbody>
</table>

Appendix 1 - MIDI Values Chart (etechdocs.com)
<table>
<thead>
<tr>
<th>MIDI number</th>
<th>Note name</th>
<th>Keyboard</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>A0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>B0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>C1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>D1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>E1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>F1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>G1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>A1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>B1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>C2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>D2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>E2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>F2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>G2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>A2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>B2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>C3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>D3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>E3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>F3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>G3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>A3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>B3</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>60</strong></td>
<td><strong>C4</strong></td>
<td></td>
<td><strong>261.63</strong></td>
</tr>
<tr>
<td>62</td>
<td>D4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>E4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>F4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>G4</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>69</strong></td>
<td><strong>A4</strong></td>
<td></td>
<td><strong>440.00</strong></td>
</tr>
<tr>
<td>71</td>
<td>B4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>C5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>D5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>E5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>F5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>G5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>A5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>B5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>C6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>86</td>
<td>D6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>E6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>89</td>
<td>F6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>G6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>A6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>B6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>C7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>98</td>
<td>D7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>E7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>F7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>G7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>A7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>107</td>
<td>B7</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>108</strong></td>
<td><strong>C8</strong></td>
<td></td>
<td><strong>4186.0</strong></td>
</tr>
</tbody>
</table>

Appendix 2 - MIDI Keyboard Values (J. Wolfe, UNSW)
Appendix 3 - Adobe Flash IDE
Appendix 4 - Note Break Down Diagram (quamut.com)

Appendix 5 - Notes on a Keyboard (quamut.com)

Appendix 6 - Mnemonic Device / Memory Aid (quamut.com)
Appendix 7 - Flash Music Package Structure

Appendix 8 - Abumarkub Package Structure
Appendix 9 - Flash Music Online Folder Structure
Appendix 10 - Persona

**Sid Almond**

**Student**

"I love to play computer games at school and when I get home. Sometimes we even play games which help to teach us at school"

**Web experience:**
Grew up around computers and Dial-up Internet. Now uses fast speed broadband. Uses computers at school. Also uses a computer at home to play games and do homework.

**Psychographics**

**Personal:**
Parents own a PC
Parents also own a Mac Book which he uses sometimes
Broadband access at home and school
Keen interest in computer games
Plays simple online games
Likes to experiment on a keyboard in music lessons at school

**Work:**
Homework due twice a week

**Goals:**
Wants to have fun
Wants to complete his homework on time
Wants to experiment on a keyboard at home as well as school

**Issues:**
Has a short attention span and is confused by complicated / busy websites
From a low income family, cannot afford private music lessons
Does not like the pressure of competing with school peers
Needs praise for any achievements as a form of encouragement
Forgetful, requires frequent reminders and revision
Has limited reading abilities
Requires sentences to be broken down into bite-size chunks
Appendix 11 - MIDI to Flash Proof of Concept

Appendix 12 - MIDI Bridge Scheme (Abumarkub)