Research Manual

Institute of Technology, Carlow
B.Sc.(Honours) in Software Development
CW238

Project Title:
Musical Editor

Document Date:
23/11/12

Student Name and ID:
Robert Connolly C00123951

Supervisor:
Paul Barry

Abstract

This research manual, is for research in relation to a musical editor I will be developing for my Honours degree project. The musical editor will allow musicians, or non musicians, to write, read(for learning other's music), and playback tablature that's written for electric/bass guitar, and drums in the editor. I will review offline and online musical editors similar to what I will be developing, giving pro's and con's to features of them that will be the core features of my project. The first review will be an in depth look at these core features, with subsequent reviews reviewing only differences in these features from previous reviews. It will also give an overview of some programming language GUI and audio libraries, that can be potentially used to develop my project.

Table of Contents

Similar Offline Musical Editors

Power Tab

"...Power Tab Editor is a tablature authoring tool for the Windows operating system. It is intended to be used to create guitar sheet music, more commonly known to musicians as guitar tablature and bass tablature. (aka guitar tab/bass tab). The program provides the most commonly used symbols in tablature, including chord names, chord diagrams, rhythm slashes, bends, slides, hammer-ons/pull-offs, harmonics and palm muting. A useful piece of software for people who want to learn how to play guitar, and for experienced guitarists who want to transcribe their own music and/or guitar lessons. The software can be used by both acoustic and electric guitar players alike..." **1.[Pow]**

Features

1. Tablature Notation

The structure of the tablature, see Fig 1, is that of a graphical representation of the strings on an electric or bass guitar. The number of strings we want represented for either instrument can be altered, depending on how many we have on our guitar, or even if we just want to tablature a subset of our strings.

To write the tablature, all we need to do is place the onscreen cursor over the appropriate string of our guitar. Then just type in from the keyboard the fret number to be played, and so on in the sequence the music is played. There is no need to know musical notation to write our music, although it does automatically write it for us above the tablature as we write it.

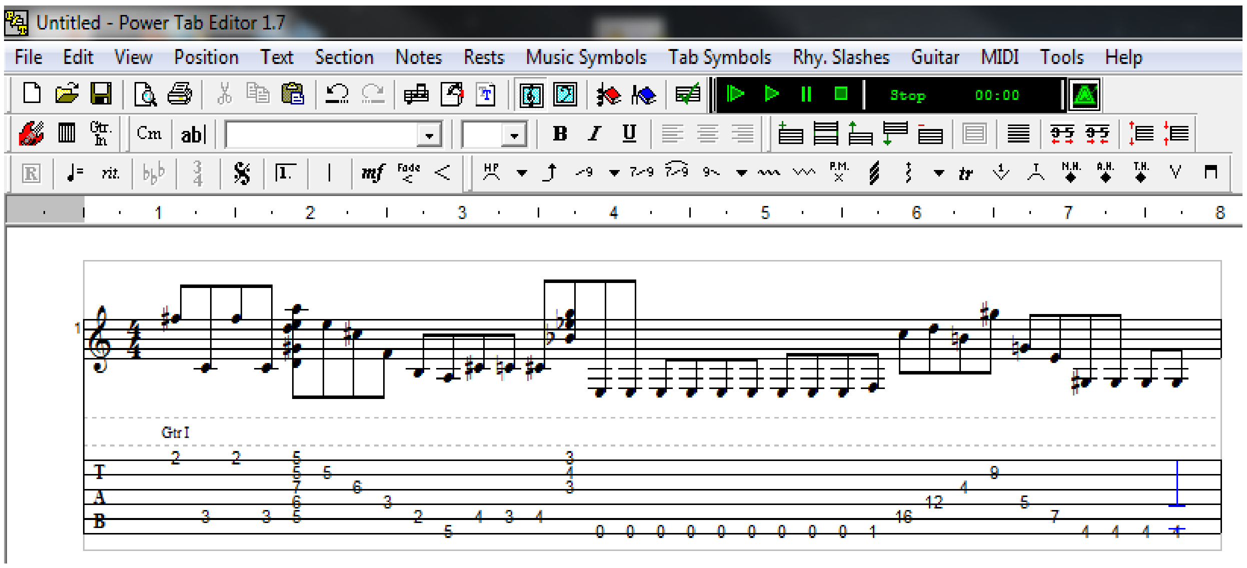

Untitled - Power Tab Editor 1.7
File Edit View Position Text Section Notes Rests Music Symbols Tab Symbols Rhy. Slashes Guitar MIDI Tools Help
Stop 00:00
Gtr I

Fig 1: Tablature written in Power Tab

Pros:

- Easy to read and understand tablature notation.

- Very intuitive interface. It's very easy to write what frets we play on the right string, and fret, at any given time within the music. Excellent for beginners and musicians who have no musical notation knowledge. User friendly to musicians and to users who don't play a guitar.

- Suitable for musicians who know music notation, as it is written along side the tablature. As a result, its a good tool for advanced musicians to learn somebody elses music from.

- Copy and pasting of repeated music in a song.

Cons:

- Can't write music in musical notation.

2. Electric and bass guitar scores

The editor allows us to enter tablature for both electric and bass guitar separately. It also offers the musician the ability to add up to seven different guitar parts in total for both scores. See *Fig 2*.

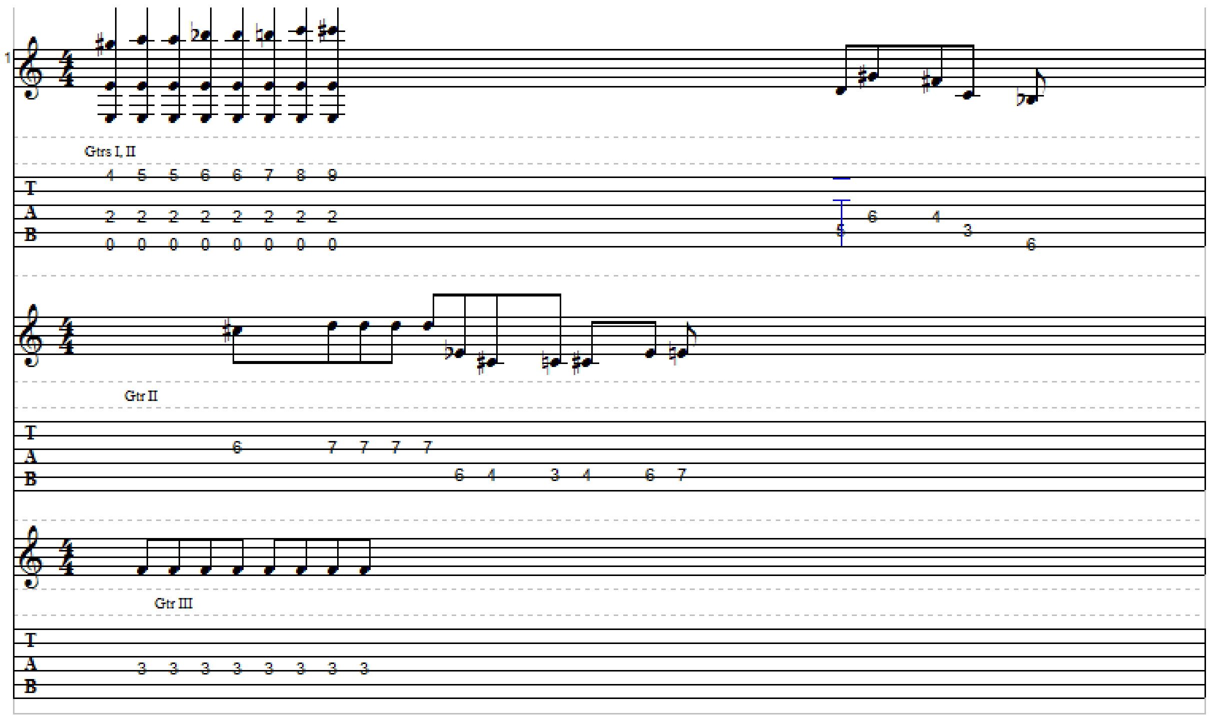


Fig 2: Electric guitar section with three staffs

Pros:

- Allows the musician to type in tablature for the bass and electric guitar separately, which is crucial as the bass usually plays a subset of the electric guitar. Also, as the bass is a different instrument to the electric, different music scores will mostly be written for it at times in the music.

- Allows music to be written that has more than one guitar in it. So, harmonies and solos in the music can be included and written over the rhythm of the electric and bass. There is enough different guitar parts, up to seven, to write all parts of virtually all styles of guitar music.

- It allows us to show how many guitars play a certain part within the music, without having to repeat the tablature for a different guitar. We see what is being played by each part within a score altogether on the tablature. This is done simply by labelling the appropriate section of the tablature with, which guitar parts play it. It aids in writing parts of guitar in the right key, as we can see exactly what another part is playing as we write a part.

Cons:

- The way the different guitar parts are implemented has downsides. There are only two different scores in which to write more than one electric or bass line. Each score is broken up into sections. Each section contains all the guitar parts that are played in a song for that section.

A section can have only up to three staffs, which can be three different parts played at the same time. Its fine for the bass score as there is mostly only ever one bass line in music, and possibly a harmony part. As for the electric, for some music styles there is more than three guitar lines at a time when rhythm, harmonies and solos are taken into account. So, the electric guitar score is a little limited in its scope.

- Also, there is no option to have different scores for different electric or bass guitar parts within themselves. It can be simpler to read different guitar parts when they have there own separate score without other parts being on screen, which may cause confusion at times. Its easier visually to read parts isolated by themselves.

  For writing purposes, it can be easier to write a guitar part by itself in a separate score. This is because we don't have to worry about adding extra staffs to a section each time, and labelling it the correct guitar part, which can be cumbersome. If we leave writing a part fully until another day, it's also visually easier to find where we left off. Mistakes could have been made in correctly labeling guitar parts in a previous session, and confusion may easily arise as to which guitar part is playing what. Having separate isolated scores for each part can eliminate this potential messiness and mix up of parts. Especially if the song is not actually written yet, as the composer is unsure of each part because they are still in the process of writing it, and the individual guitar parts.

  Separate scores for each guitar part can greatly simplify the writing process and make it easier to do when potential problems just mentioned can be removed. It allows for a more pleasant experience. The focus should be on writing or learning the music, not being impeded by the tablature due to potential over complication and mess of writing it in the first place.

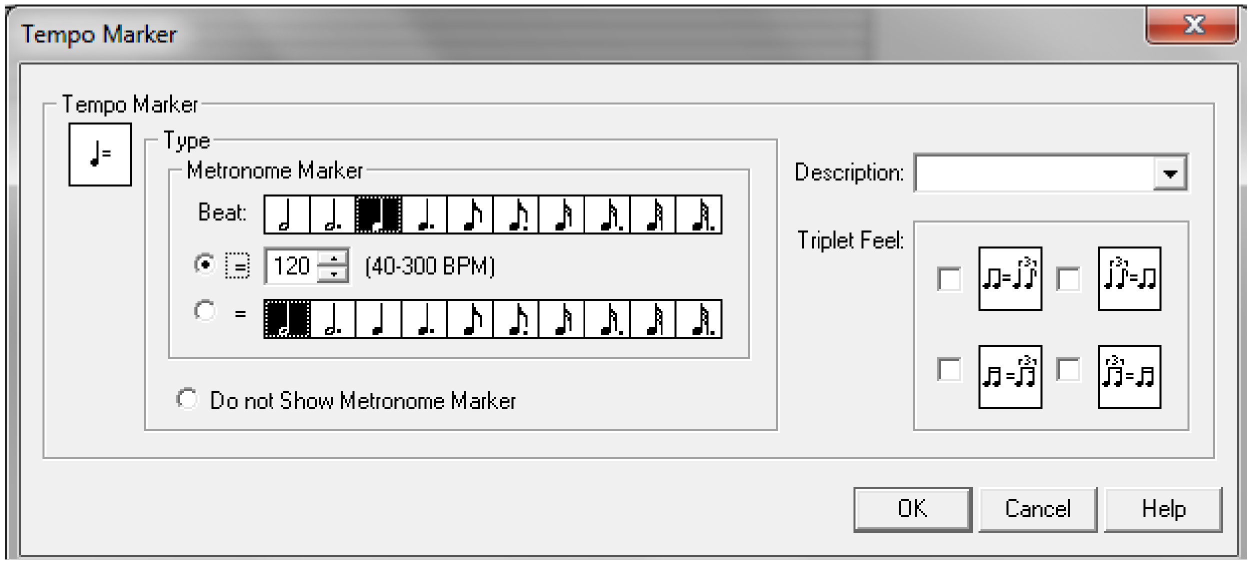
3. Metronome

The metronome is a series of audible electronic pulses or beats that sound at regular timed intervals. It is used by musicians to keep a steady tempo to their music, in order to aid staying accurately in tempo and on correct time when playing music. A metronome has beats per minutes(bpm), see *Fig 3*, that give it its tempo. Increase or decrease the bpm for faster or slower tempos, respectively.

It also uses a time signature, see *Fig 4*. This is usually written in fractions, e.g. 4/4. The top number is how many beats is played per measure(bar) of music. The lower number is how long the note is played. Taking a full note as some certain length of time and dividing it by the lower number gives you how long each beat of the top number is played, e.g. /4 means a quarter note, or quarter the length of time as a full note that each beat is played. Although, the beat that is heard is not played the full length of the lower value, as it would be extremely difficult to tell the difference between beats if they played continuously to the next one. The length is more about how long it takes before the next beat is played, not how long a beat is actually played or heard.

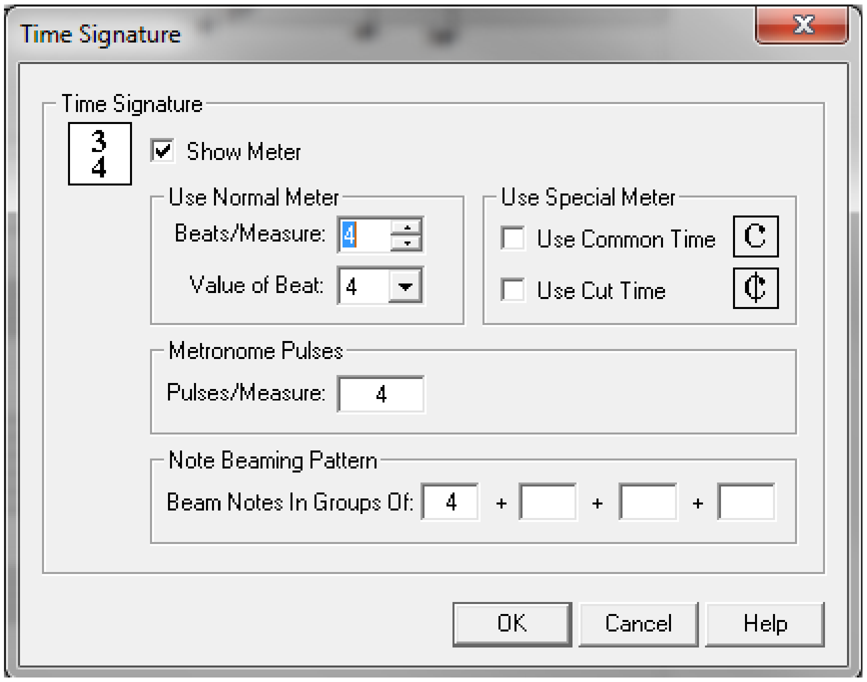
A different sounding beat is usually heard for the last beat of a bar to distinguish the ending of a measure and the beginning of a new.

Power Tab uses a metronome to help us keep our written tablature in proper timing, and the tempo we want our music to be played at.



Beat: 120

Fig 3: Metronome tempo settings(bpm)



Show Meter [✓]
Beats/Measure: 4
Value of Beat: 4
Pulses/Measure: 4
Beam Notes In Groups Of: 4 + + +

Fig 4: Time signature settings

Pros:

- Crucial for any tablature editor that has a playback feature. Allows someone learning our music to hear the exact timing and tempo of our song.

- It has a count in before the music is played, so another musician wanting to play along with our music can come in right on time with the start of the music.

- Simple to turn on and off, and change the bpm and time signature.

Cons:

- The metronome can only be played during playback. It would better if the musician could physically play their instrument along to just the metronome first, to figure out what tempo they want to play their music at, and figure out their timings within the song. It makes writing the tablature easier because they don't have to write some of the tablature first to hear if its at the right tempo, and what time signatures to use for the bars of the music. Its much quicker to figure that out before writing the tablature.

- The different sound that is used in Power Tab to signal the end of a bar is not noticeably different from the other beats. It should be more distinct, as it allows a clear audible indication to the musician when one bar ends and another begins. This is very important, as a musician can become easily disorientated in terms of where they are in a song, and their timing can suffer greatly as a result because they can't tell exactly where a bar ends and begins. In fact, the metronome may even cause worse timing than actually not using one, due to the disorientation caused by all beats sounding the same.

  Its poorly implemented. Power Tab tries to use different sounding volumes to achieve this distinctness called a 'Strong Accent', versus 'Weak Accent' for the rest of the beats. This can leave us turning down the other beats so low we can't really hear them. A clear distinct change in pitch is far more suitable.

4. Types of guitar notes

On an electric and bass guitar there are various ways we can play and manipulate notes. We can have hammer-ons, pull-offs, slides, bends, etc, and how long we play each note, or groups of notes for. Power Tab has extensive options for this for writing tablature and how each note should be played and for how long, see *Fig 5* and *Fig 6*.
In order for a musician to tablaturise their music properly, its crucial to be able to notate how each note is played for teaching others or remembrance for future reference. When reading the tablature for learning purposes, it tells the musician how to play each note.

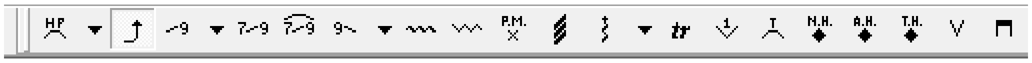


Fig 5: Toolbar for note types. Hammers-on, pull-offs, bends, etc.

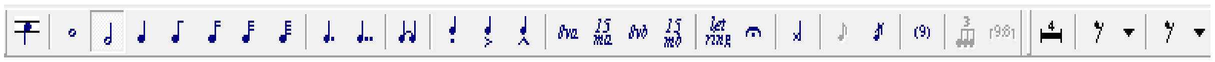


Fig 6: Toolbar for note length duration. Half note, quarter note, etc.

Pros:

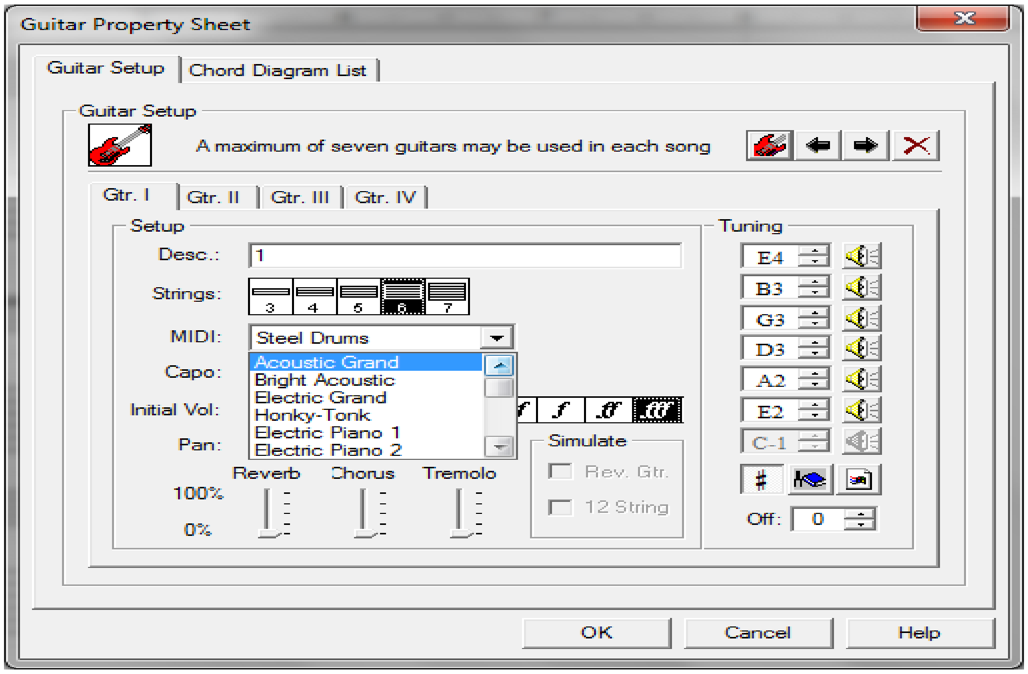
- Power Tab has a toolbar for most if not all the note types, hammer-ons, pull-offs, etc, we could possibly need for accurately writing our tablature. It also has a toolbar for the duration of how long a note should be played, which is also crucial in reading and learning how each note is played.

- User friendly toolbars, with easy access.

5. Playback

*2. [Ent]*Power Tab has a playback feature that allows us playback and hear the music we have tablatured. This is done through MIDI(Musical Instrument Digital Interface). MIDI allows digital instruments and computers and other similar devices, the ability to communicate with each other using MIDI information as the common standard to communicate and share musical information.

Power Tab uses it to simulate the sound of many different instruments virtually, see *Fig 7*. It's crucial for musical editors to simulate different instruments together on playback, when tablurising many different instruments, in order to give a clear audible indication of how the music sounds with distinct definition to each instrument. So, we can tell the bass from a guitar for instance.


Guitar Property Sheet

Guitar Setup | Chord Diagram List

Guitar Setup

A maximum of seven guitars may be used in each song

Gtr. I | Gtr. II | Gtr. III | Gtr. IV

Setup

Desc.: 1

Strings: 3 4 5 6 7

MIDI: Steel Drums

Acoustic Grand
Bright Acoustic
Electric Grand
Honky-Tonk
Electric Piano 1
Electric Piano 2

Capo:

Initial Vol:

Pan:

Reverb Chorus Tremolo

100%

0%

Simulate

Rev. Gtr.

12 String

Tuning

E4
B3
G3
D3
A2
E2
C-1

Off: 0

OK Cancel Help

Fig 7: Guitar setup showing the MIDI soundbank for different instruments

Pros:

- Its indispensable having a playback feature of an editor, in order for a musician to hear what they're tablaturising is accurately what it sounds like. Therefore, helps musicians to accurately learn the tablature.

- Aids song writing if we have no physical instrument or can't play one, so we can tell what our music sounds like.

- It's great for users who can't play an instrument so they hear what they are writing, and helps musicians hear what they play physically is being represented properly in the tablature.

- Power Tab has a large MIDI sound bank to choose from to get a sound we like. This is also crucial when we have multiple guitars, such as bass and electric, in order to distinguish between them in playback.

Cons:

- Playback can only start from the start of a bar, not within it.

- Bars need insertion manually. They are not automatically generated through the time signature. Adding them in manually, is very tedious.

Tux Guitar

[3.[Tux]]Tux guitar is an open source tablature editor. Its features offer:

- *"...Tablature editor*
- *Score Viewer*
- *Multitrack display*
- *Autoscroll while playing*
- *Note duration management*
- *Various effects (bend, slide, vibrato, hammer-on/pull-off)*
- *Support for triplets (5,6,7,9,10,11,12)*
- *Percussion*
- *Repeat open and close*
- *Time signature management*
- *Tempo management*
- *Imports and exports gp3, gp4 and gp5 files..."*

In this section, I'll just be highlighting the different features Tux Guitar has for tablaturising music over Power Tab.

Features

1. Tablature Notation

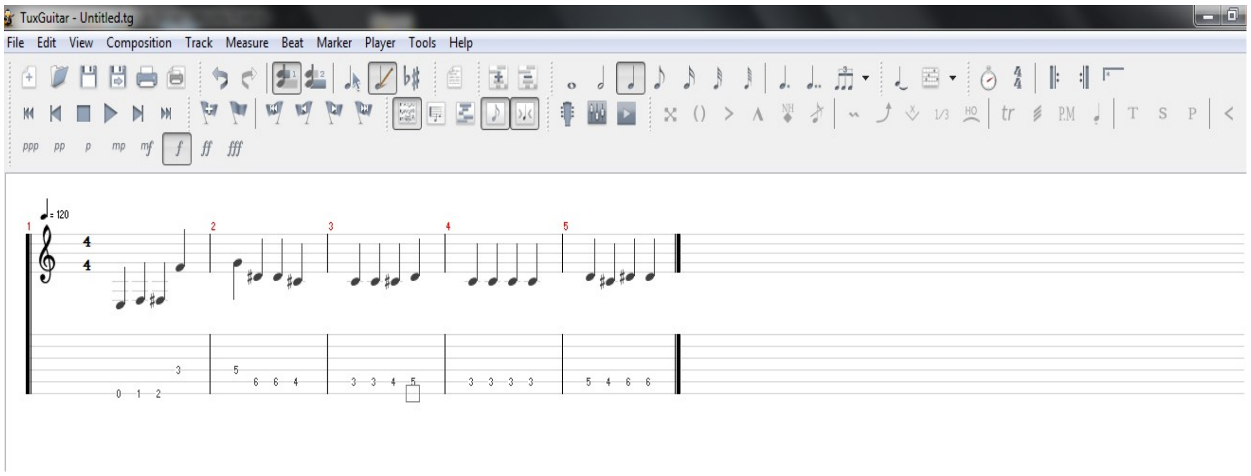

TuxGuitar - Untitled.tg
File Edit View Composition Track Measure Beat Marker Player Tools Help
ppp pp p mp mf f ff fff
= 120

Fig 8: Tablature written in TuxGuitar

TuxGuitar, see *Fig 8*, has the same type of default tablature interface as Power Tab and tablature is written in the same way(see 'Tablature Notation' description for Power Tab).

Fretboard:

A graphical representation of a guitar fretboard can be used to tablature the music. Simply, put the mouse cursor over the fret and string we want and click. The fret number is then automatically generated on the tablature for us, on the appropriate string we used on the fretboard. See *Fig 9*.

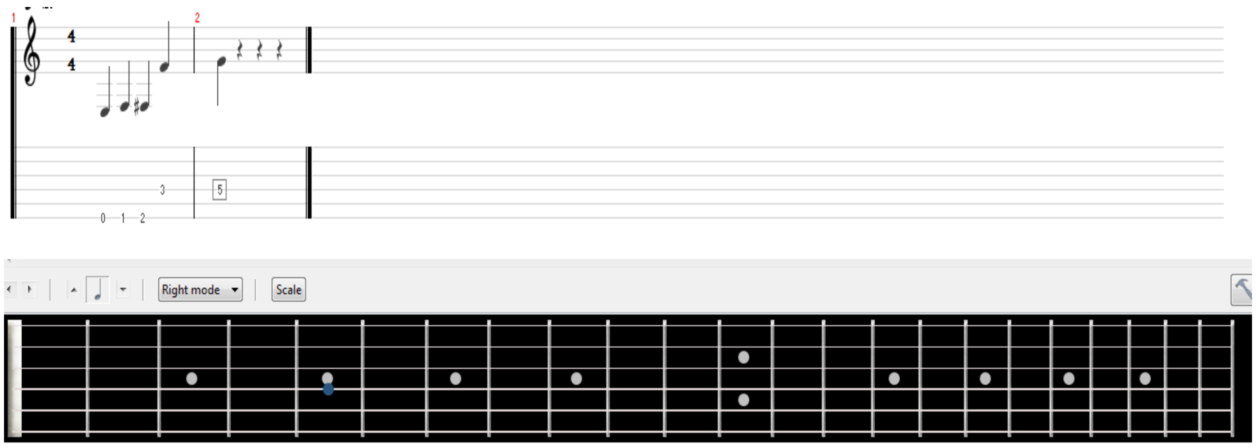


Fig 9: Fretboard uses to tablaturise music

Pros:

- Very easy and more intuitive to use than typing in the tablature manually. Makes remembering how the guitar track is played alot easier for a musician, as they can see the fretboard.
  "...I found TuxGuitar 1 very easy to use, real easy to set up, Love the guitar fret board..."

4. [CNET1]

Piano Editor:

A piano editor can also be used to tablaturise. Its simply a graphical representation of a piano's keys. You just click a key on the piano and it automatically generates a fret number on a string of the tablature. See *Fig 10*.

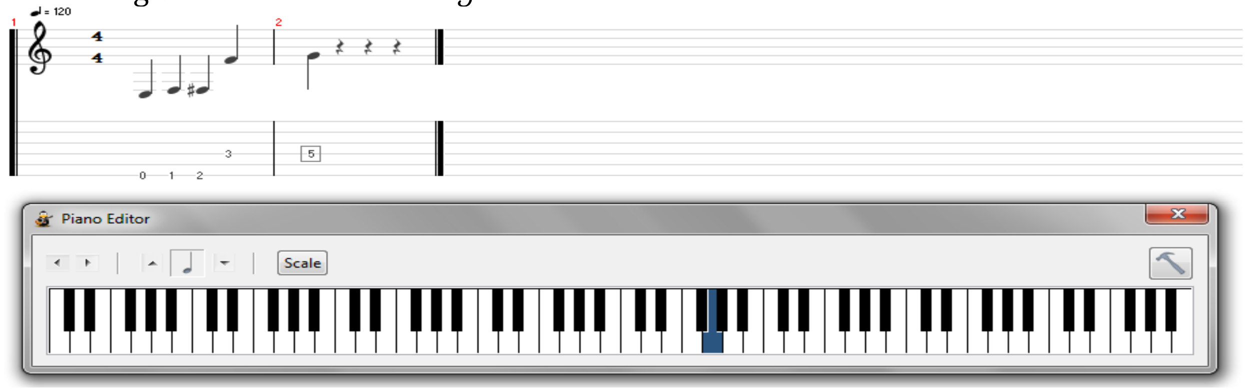


Fig 10: Piano editor for tablaturising

Pros:

- It has an option to choose different scales we want to play in, and it will highlight which keys on the piano are in that scale.

Cons:

- Doesn't work very well for guitar tablature or for percussion, as each key on the piano is not labelled with any notes or percussion types. Not intuitive at all because its very hard to tell what note and on what string, we are pressing one of the piano's keys for a guitar. The same for the percussion, because there is no indication either of what part of the percussion we're pressing, e.g. cymbal, snare, etc. Also, this editor only tablaturises in guitar form, so the sum of all these factors makes it not easily usable.

Matrix Editor:

- A matrix editor can be used to tablaturise percussion. It gives a list on the left of the types of percussions such as snare, hi-hat, bass drum, etc. On the right, is a grid for each type of percussion. It is broken up into how many beats there are in a bar. We simply click on the appropriate grid for a percussion, for the appropriate beat in the bar. The percussions get represented on the strings of the guitar tablature, as the number of the percussion's position, in the list of percussions on the left of the editor. If we have six strings setup in the tablature, we can have up to six different percussions at once. See *Fig 11*. 5. [Tux2]

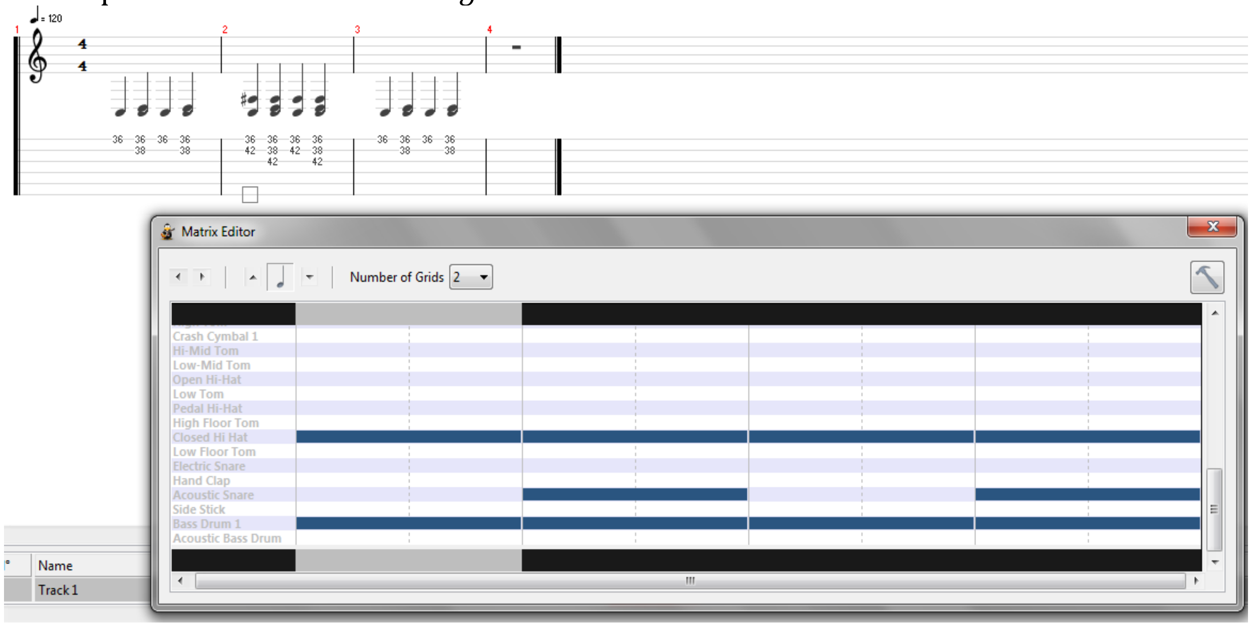


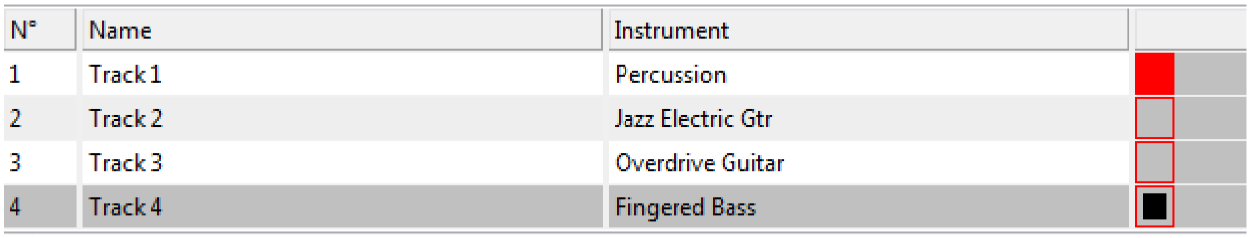
Fig 11: Matrix editor for percussion tablaturising

Pros:

- Excellent for tablaturising the percussion. It shows us what percussion we're using at any time, and on what beat of the time signature we're using the percussion. Great for writing percussion accurately and in time in the music tempo. This allows for a very intuitive way to tablaturise the drums. 5. [Tux2]

- We can tablaturise all the bars of the percussion in the music as we can scroll to each bar to write it. 5. [Tux2]

2. Electric and bass guitar scores



N°	Name	Instrument	
1	Track 1	Percussion	🟥
2	Track 2	Jazz Electric Gtr	▢
3	Track 3	Overdrive Guitar	▢
4	Track 4	Fingered Bass	■

Fig 12: Tracks used in a current song, which can be switched between from here.

Tux guitar is a multi-track editor, which means it allows us to tablaturise multiple instruments in the same song. Each instrument has a track of its own, and each track has its own tablature.
We can switch between tracks at the bottom of the tablature, simply by clicking on the appropriate track for an instrument, see *Fig 12*.

Pros:

- Most songs have more than one instrument in them. Tux guitar allows us to tablaturise all these different instruments in one song.

- Each instrument has its own track, therefore tablature, to write its own part. It allows for clearer writing of individual parts without the confusion that could be caused by having to write different individual parts on the same track, as parts could potentially be mixed up and badly labelled.
  Separate tracks also make it very easy for anyone learning the song from the tablature, to easily tell what each part is by itself, as each part is separated into tracks. Much easier to read and learn as a result, with no confusion as to what to play for each part.

Similar Online Musical Editor

Noteflight

Noteflight is an online musical editor. There is a free version and a purchase version. I will be concentrating on the free version. The free version offers:[6. [Note]]

- Create up to 10 Scores, with any number of staves plus guitar & bass tabs.
- 15 Basic Instruments, for composition and audio playback of your scores.
- Share:- Link, embed and share scores online.

Features

Noteflight offers the same functionalities as described for Power Tab and Tux Guitar in terms of writing tablature, with some differences. I will be giving an overview of the differences.

1. Musical Notation

Unlike Power Tab and Tux Guitar, Noteflight allows the user to write the music in musical notation, as well as tablature notation. See *Fig 13*.

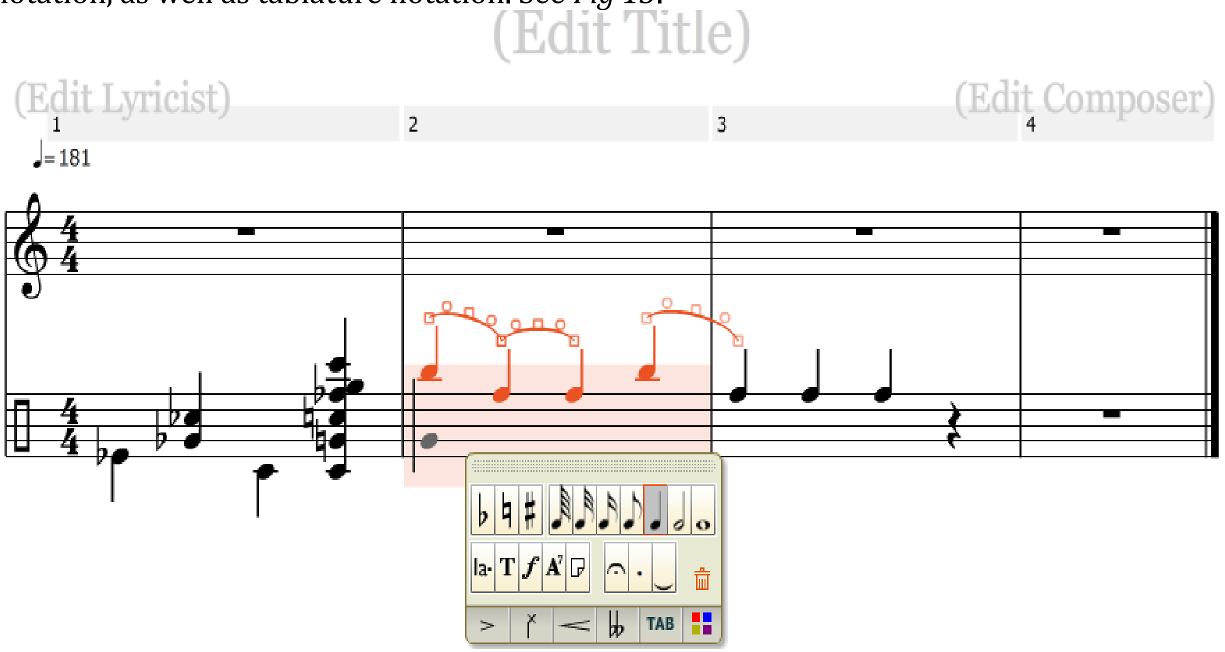


Fig 13: Music written in musical notation in Noteflight

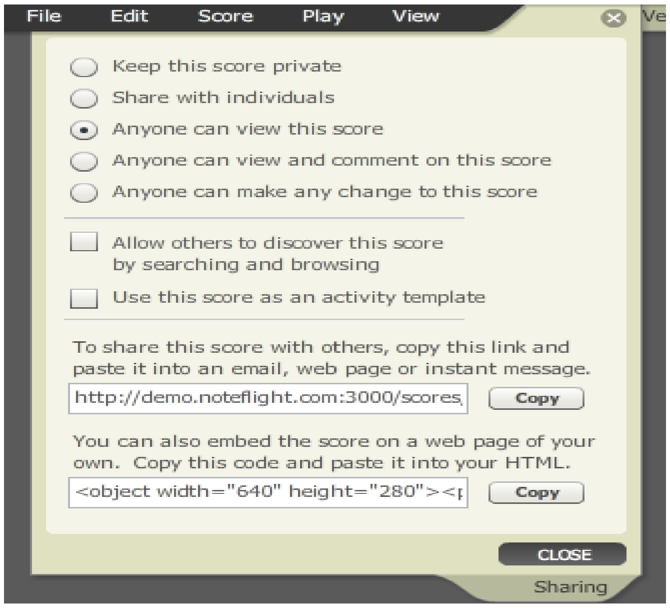
Notes are simply entered in from the keyboard, or mouse, at an insertion point. We can either type in the actual notes, from 'A' to 'G', from the keyboard or use the mouse cursor to place a note where we'll need it on a particular string.[7. Note2]
Drums can be musically notated, too.

Pros:

- Offers the facility for advanced musicians to be able to write out their music, and read other's music, in musical notation.

2. Sharing

Probably the most notable feature of this online editor, compared to offline editors, is the ability to share our music directly. Other users or musicians can look at our music online without having to send a copy of our file and open it up to view it, like we do in an offline editor. See *Fig 14*.
We can decide what people can do with our music when viewing it. For example, they can only view it, they can view it and comment on it, they can view and change it.
We can setup our music that it can be searched for by title, composer, etc., or simply only people who have its url can view it. We can also keep the score private for ourselves. [8. Note3]



Keep this score private
Share with individuals
[✓ Anyone can view this score]
Anyone can view and comment on this score
Anyone can make any change to this score

Allow others to discover this score by searching and browsing
Use this score as an activity template

To share this score with others, copy this link and paste it into an email, web page or instant message.
http://demo.noteflight.com:3000/scores,

You can also embed the score on a web page of your own. Copy this code and paste it into your HTML.
<object width="640" height="280"><

Fig 14: Sharing options for a score.

Pros:

- Far easier to share with other musicians, or users, when we can just share our score as it is online, without having to make a copy of its file and send it to another like we do for offline editors.

- We don’t have to have the editor installed on our computer in order to view another’s score. We can just view it online.

- Far better for collaboration with others, as they can make changes to our exact score that we are working on rather than a copy. Others just need to access the url of our score, rather than upload and send our file via email, then download a file and open it up in an offline editor to view it and change it. As a result, much more efficient and easier to collaborate on.

Overall

Pros:

- No need to install the editor on our computer, as it is used online.

- Can be used on any computer with an internet connection, so we’re not restricted to using it on one machine because we don’t have to install it.

- Much easier to share and collaborate on score’s, as we and others only need the url of the score to view and/or change the score.

Cons:

- Limited to only 10 scores maximum you can have written at any one time with the free version.

Technologies

In this section, I will be looking at the various technology options I have for my musical editor. I will need a graphical user interface(GUI) for my editor, and a technology for producing the MIDI sounds within the editor.

GUI Libraries:

Java

Two of Java's GUI libraries will be looked at in this section. They are Swing and SWT(Standard Widget Toolkit).

1. Swing

This a GUI toolkit for java. Its based on a previous toolkit called AWT(Abstract WIndows Toolkit), with more functionalities. ***9. [IBM]***

Pros:

- It correct's AWT's variation in behaviour of controls on different hosts, because it doesn't depend on the hosts controls to implement the GUI's.
  "...To overcome the variation in behavior across different hosts, Swing minimized its dependence on host controls. In fact, Swing uses peers for only top-level components like windows and frames. Most components (JComponent and its subclasses) are emulated in pure-Java code..."

- *"...Swing is naturally portable across all hosts. Thus, Swing does not typically look like a native application. In fact, it has multiple looks, some that can emulate -- although often not exactly -- different hosts and some that provide unique looks..."*
  As a result:
  "...Programmable look and feel
  Each component's look (appearance) and feel (how it reacts to input events) is controlled by a separate and dynamically replaceable implementation. This allows the look and feel of all or part of a Swing-based GUI to change..."

- *"...Swing uses the term heavyweight for peer-based components and lightweight for emulated components. In fact, Swing supports mixed heavy and light components in a single GUI, such as mixing AWT and Swing controls in the same JContainer, but care must be taken with drawing order if the components overlay each other..."*

- ***"...Separation of model from the view and controller***
  For all components with models (such as buttons, lists, tables, trees, rich text), the model is separate from the component. This allows the model to be adapted to the needs of the application and for it to be shared by multiple views. Default models per component type are provided for convenience..."
  The model contains data to be worked upon. The view is the implementation of the UI and reads users actions on it. The controller is the code that carries out actions upon the model's data and what's entered in the view.

- *"...Swing supports automatic disposal of GUI components..."*

- *"...Swing also supports the AWT bottom-up and top-down construction methods..."*

- *"...Like AWT, one of Swing's advantages is that it comes standard with Java technology. This means you do not need to install it..."*

Cons:

- *"...Swing may not be able to take advantage of hardware GUI accelerators and special host GUI operations. As a result, Swing applications may be slower than native GUIs. Sun has worked hard on the performance of the recent versions of Swing (Java V1.4 and 1.5), and this disadvantage is becoming less noticeable. Because Swing's design is more robust, its code base is larger. This can mean it takes a beefier machine to run it than AWT or SWT might..."*

- *"...Swing components are not thread-safe, which means you need to be concerned as to which thread in your application updates the GUI. If you err in thread use, unpredictable behavior, including user interface glitches, can occur. Utility routines exist to help manage threading issues..."*
  Thread-safety relates to how multiple threads access shared components in this case. Swing components are not thread-safe, so threads can change them at the same time, which can cause the problems as mentioned.

- It can be easy to become dependent on features provided in more recent versions of java. As a result, users may be forced to upgrade their java runtime versions.

2. SWT

"...SWT is a low-level GUI tool kit comparable in concept to AWT. JFace is a set of enhanced components and utility services to make building GUIs with SWT easier. The builders of SWT learned from the AWT and Swing implementations and tried to build a system that had the advantages of both without their disadvantages..." 10. [IBM2]

Pros:

- *"...SWT is a lot like AWT in that it is based on a peer implementation. It overcomes the LCD* problem faced by AWT by defining a set of controls adequate to make most office applications or developer tools and then, on a host-by-host basis, creating emulated*

(like Swing) controls for any not supplied by the particular host. For most modern hosts, nearly all the controls are based on native peers. This means that an SWT-based GUI has a host look and feel, and host performance. This avoids the most widely held complaints with AWT and Swing. Certain hosts have low-function controls, so SWT supplies extended, often emulated, versions to allow the more generally expected behavior..."
*LCD: This stands for the Lowest Common Denominator, where AWT had only components common to different host systems, which left many other commonly used components on different systems not accounted for. They would have to be written from scratch, which is a big burden.

- *"...SWT is different from AWT in how the peers work. In SWT, the peers are just wrappers on host controls. In AWT, peers can provide services to minimize the differences between hosts (this is where AWT ran into a lot of its behavior issues). This means that an SWT application is really a host application..."*

- Very portable.

Cons:

- *"...SWT does not support automatic disposal of GUI controls. This means that you must explicitly dispose of any controls or resources, such as colors or fonts, you create, as opposed to getting from an API call. This effort is eased somewhat as container controls dispose of their child controls automatically..."*

- *"...SWT only allows you to build GUIs top-down. Thus, a control cannot exist without a parent container, and its parent cannot, in general, change over time. This approach is much less flexible than AWT/Swing. Controls are added to a parent container during creation and removed when disposed. Also, the SWT use of style bits only at construction time can limit the flexibility of some GUI controls. Many styles are hints only and do not work the same on all platforms..."*

- *"...SWT components are not thread-safe. Although, if you err in thread use an exception is thrown to recover from the error. Special utility routines exist to help manage threading issues..."*

Python

1. wxPython

*11. [Zet]*wxPython is a cross platform toolkit for creating desktop GUI applications, that is that it can be used to by developers to create applications on Windows, Mac and various Unix systems.

It is a wrapper around wxWidgets, which is a C++ library. By wrapper, it is meant that python code is used as an interface to implement the C++ code. So you can write in python code, which in turn calls the corresponding wxWidget it is a wrapper for, and the C++ code in the wxWidget carries out the functionality for you. You don't have to worry about the C++ code, only python. wxPython consists of five basic modules:

1. Controls
2. Core
3. GDI
4. Misc
5. Windows

1. *"...Controls module provides the common widgets found in graphical applications. For example a Button, a Toolbar, or a Notebook..."*
2. *"...The Core module consists of elementary classes, that are used in development. These classes include the Object class, which is the mother of all classes, Sizers, which are used for widget layout, Events, basic geometry classses like Point and Rectangle..."*
3. *"...The Graphics Device Interface (**GDI**) is a set of classes used for drawing onto the widgets. This module contains classes for manipulation of Fonts, Colours, Brushes, Pens or Images..."*
4. *"...The Misc module contains of various other classes and module functions. These classes are used for logging, application configuration, system settings, working with display or joystick..."*
5. *"...The Windows module consists of various windows, that form an application. Panel, Dialog, Frame or Scrolled Window..."*

wxPython API

"...wxPython API is a set of methods and objects. Widgets are essential building blocks of a GUI application. In wxPython we have lots of widgets. These can be divided into some logical groups:

Base Widgets

These widgets provide basic functionality for derived widgets. They are called ancestors. They are usually not used directly. See Fig 15.

wx.Window
wx.Control
wx.ControlWithItem

Fig 15: Base Widgets

Top level Widgets

These widgets exist independently of each other. See Fig 16.

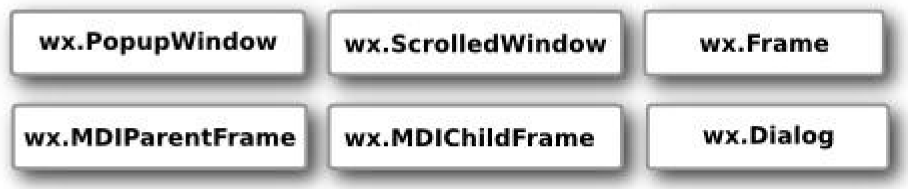


Fig 16: Top level Widgets

Containers

Containers contain other widgets. See Fig 17.

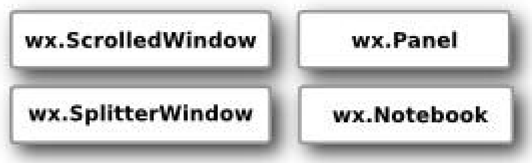


Fig 17: Containers.

Dynamic Widgets

These widgets can be edited by users. See Fig 18.

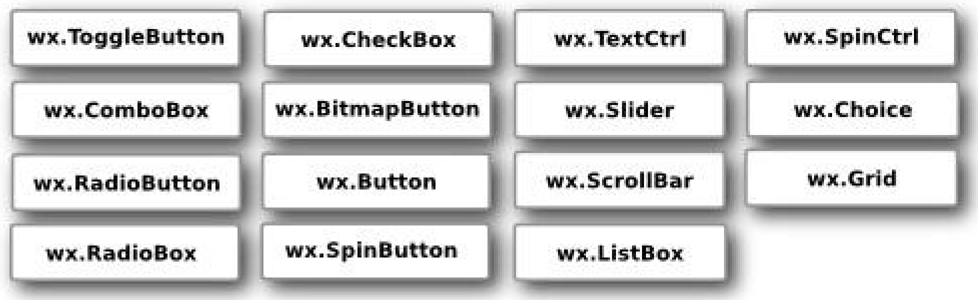


Fig 18: Dynamic Widgets

Static Widgets

These widgets display information. They cannot be edited by user. See Fig 19.

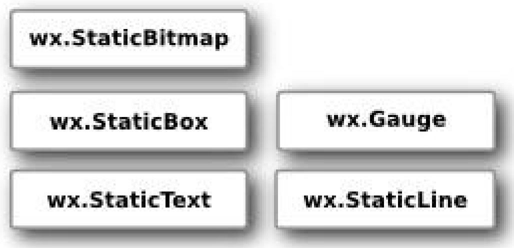


Fig 19: Static Widgets.

Other Widgets

These widgets implement statusbar, toolbar and menubar in an application. See Fig 20.

Fig 20: Other Widgets.

Inheritance

There is a specific relationship among widgets in wxPython. This relationship is developed by inheritance. The inheritance is a crucial part of the object oriented programming. Widgets form a hierarchy. Widgets can inherit functionality from other widgets. Existing classes are called base classes, parents, or ancestors. The widgets that inherit we call derived widgets, child widgets or descendants..."

C++

1. Qt

*12. [Mac]*Qt is a multi-platform graphical toolkit.

"...The first and most basic set of functionality in Qt, is for creating graphical interfaces. Qt includes widgets (the equivalent of Controls) for buttons, checkboxes, radio buttons, tabs, icon panes, canvases, dialog boxes, and other common interface elements.

Object Orientated Programming (OOP) is fundamental to using Qt. Each of the components is available as a class (such as a QPushButton to create a push button widget), and each class has a number of methods to handle common tasks and features. Although some developers use procedural programming for graphical applications, the nature of OOP lends itself well to GUI programming due to the fact that inheritance is fundamental to GUI's. As an example, there is the general concept of a button (something that you click on), and then specific types of buttons (push, radio, toolbar, etc). In Qt, inheritance is used in this way so that the general QButton class is inherited by a more specific QPushButton class, for example. This process gives the developer all the functionality that could be needed for that specific widget, and the lower level functionality for its inherited class.

Qt Designer essentially gives you the ability to draw your graphical interface visually by dropping interface elements onto a window. Qt Designer is a very flexible tool, and has support for all of the graphical widgets that are available in the toolkit. Not only can you add *items such as buttons, checkboxes, radio buttons, scrollbars, textboxes, etc., but you can also add menus, and their items. Adding the components to your application window is as simple as selecting the widget from the toolbar and then drawing it...”*

Audio Libraries:
Java

1. Java Sound

13. [IBM3]The following is an overview of how Java Sound handles MIDI:
“...Fig 21, illustrates the connections of various portions of the MIDI subsystem. A MIDI file is a collection of tracks with each track having a set of notes. The note has a time stamp, a note value, commands such as note on or note off, and other data associated with the note. The MIDI file is loaded into the Sequencer using the setSequence method. Once loaded, the tracks may be edited with the Java Sound programming interface. From the Sequencer, the MIDI tracks may be "sequenced" or merged together in time order. This stream may be sent to a MIDI instrument through a stream generator. The MIDI instrument can synthesize the MIDI stream into analog sound that can be heard over headphones or a speaker.

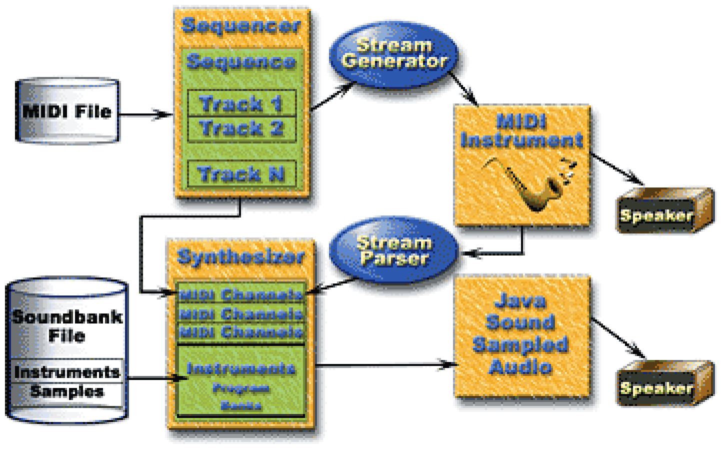


Fig 21: Connections of a MIDI subsystem.

The MIDI sequence also may be sent to the Synthesizer. In the Synthesizer, the MIDI tracks are loaded into channels. These channels are merged with instrument sounds that are loaded from a sound bank. The instruments are stored internally into an array of Programs and Banks that are used to swap in and out logical groups of instruments. Together the tracks and the instruments may be synthesized into sampled audio. Java Sound takes the sampled audio and sends it to your computer sound card that plays it over your computer speakers or headphones...”

Python

1. Python Midi

[14. [GIT]] *"...This toolkit aims to fulfill the goal, of providing you with an easy means to manipulate MIDI data. In particular, it strives to provide a high level framework that is independent of hardware. It tries to offer a reasonable object granularity to make MIDI streams a painless thing to manipulate, sequence, record, and playback. It's important to have a good concept of time, and the event framework provides automatic hooks so you don't have to calculate ticks to wall clock, for example.*

Features

- *High level class types that represent individual MIDI events.*
- *A multitrack aware container, that allows you to manage your MIDI events.*
- *A tempo map that actively keeps track of tempo changes within a track.*
- *A reader and writer, so you can read and write your MIDI tracks to disk..."*

2. MidiUtil

[15. [Midi]] *"...MIDIUtil is a pure Python library that allows one to write multi-track Musical Instrument Digital Interface (MIDI) files from within Python programs. It is object-oriented and allows one to create and write these files with a minimum of fuss..."*

3. Pyo

[16. [Pyo]] *"...pyo is a Python module containing classes for a wide variety of audio signal processing types. With pyo, user will be able to include signal processing chains directly in Python scripts or projects, and to manipulate them in real time through the interpreter. Tools in pyo module offer primitives, like mathematical operations on audio signal, basic signal processing (filters, delays, synthesis generators, etc.), but also complex algorithms to create sound granulation and others creative sound manipulations. pyo supports OSC protocol (Open Sound Control), to ease communications between softwares, and MIDI protocol, for generating sound events and controlling process parameters. pyo allows creation of sophisticated signal processing chains with all the benefits of a mature, and wild used, general programming language..."*

C++

1. libjdkmidi

*17. [Code]*The following is the functionalities of this library:
*"...MIDI parsing, MIDI Show Control message creation and handling, Standard MIDI File type 0 and type 1 reading and writing, Timestamped MIDI message and System Exclusive encapsulation, Efficient MIDI Track objects for sequencing, MIDI Track objects for editing MIDI events, MIDI message process chains, Containers for multiple MIDI Tracks with iterators, MIDI Sequencer core for sequencing and triggering GUI events, MIDI Driver abstractions for I/O and sequencing, MIDI Driver implementation for Win32 for I/O and sequencing Tempo calculations, MIDI Matrix to count note on's and off's and hold pedals to avoid stuck notes and all-notes-off problems when merging midi streams
SMPTE management and calculations..."*

Bibliography

1. [Pow], Power Tab Editor v1.7 (Build 80), http://www.power-tab.net/guitar.php, [Accessed 15/10/12].
2. [Ent], How MIDI Works, http://entertainment.howstuffworks.com/midi.htm, Accessed[14/11/2012]
3. [Tux], Tux Guitar, http://tuxguitar.herac.com.ar/, [Accessed 15/11/12].
4. [CNET1], http://download.cnet.com/TuxGuitar/3000-2133_4-10819921.html, [Accessed 15/11/2102].
5. [Tux2], Matrix Editor, http://tuxguitar.herac.com.ar/tgwiki/doku.php?id=doc:matrix_editor, [Accessed 15/11/2012].
6. [Note], Upgrade to Crescendo, http://www.noteflight.com/commerce/premium_benefits, [Accessed 18/11/2012].
7. [Note2], Adding and editing notes, http://www.noteflight.com/info/help, [Accessed 18/11/2012].
8. [Note3], Sharing and Publishing Scores, http://www.noteflight.com/info/help, [Accessed 18/11/2012].
9. [IBM], A look at Swing, http://www.ibm.com/developerworks/grid/library/os-swingswt/, [Accessed 20/11/12].
10. [IBM2], A look at SWT, http://www.ibm.com/developerworks/grid/library/os-swingswt/, [Accessed 20/11/12].
11. [Zet], Introduction to wxPython, http://zetcode.com/wxpython/introduction/, [Accessed 21/11/2012].
12. [Mac], Reviews: Qt, http://www.mactech.com/articles/mactech/Vol.20/20.03/QtReview/index.html, [Accessed 21/11/2012].
13. [IBM3], Understanding and using Java MIDI Audio, http://www.ibm.com/developerworks/library/it/it-0801art38/, [Accessed 22/11/2012].
14. [GIT], GitHub, https://github.com/vishnubob/python-midi/, [Accessed 22/11/2012].
15. [Midi], MidiUtil, http://code.google.com/p/midiutil/, [Accessed 22/11/2012].
16. [Pyo], pyo documentation, http://www.iact.umontreal.ca/pyo/manual/, [Accessed 22/11/2012].
17. [Code], libjdkmidi, http://code.google.com/p/libjdkmidi/, [Accessed 22/11/2102].