Guitar Trainer

CS A470 Final Write-up

Adam Janke

4/29/2010

Contents

[*Abstract* 3](#_Toc260347572)

[*Introduction* 3](#_Toc260347573)

[Overview 3](#_Toc260347574)

[Input Files 3](#_Toc260347575)

[GuitarPro File Format 4](#_Toc260347576)

[*Project Requirements* 13](#_Toc260347577)

[*System Design* 14](#_Toc260347578)

[User Interface 14](#_Toc260347579)

[Data Structures 15](#_Toc260347580)

[Class Diagrams 15](#_Toc260347581)

[Algorithms 16](#_Toc260347582)

[*Software Development Process* 17](#_Toc260347583)

[Testing and Debugging 17](#_Toc260347584)

[Work Breakdown 17](#_Toc260347585)

[*Results* 18](#_Toc260347586)

[Summary and Conclusions 18](#_Toc260347587)

[Appendix A: User Manual 19](#_Toc260347588)

[Minimum System Requirements 19](#_Toc260347589)

[Installation/Running 19](#_Toc260347590)

# Abstract

Since their inception, guitars have held an important role in society across the globe. It is no surprise, then, that they still remain one of the most played instruments throughout the world. While the guitar is not horribly complex to play, like everything it does require practice and patience to learn. A large deterrent to new players is that, at the beginning, the guitar is not fun to play. If a new player decides to take formal lessons, the instructor generally focuses on basics and scales. This in and of itself is not bad, however it is not exciting. More often than not a student wanted to learn guitar to play music, not to study theory or play scales. The dichotomy of this method is a new player attempting to teach themselves without professional assistance. With no prior experience, this method can be overwhelming. It is the people in the latter group who this program is intended for – to make learning new songs as easy as possible.

# Introduction

This project is not only to assist brand new guitar players. It can be very useful to beginner and experienced alike. As a beginner player myself, I see immediate benefits from using it; however I have shown it to more experienced players who have expressed interest in the finalized product also. The only subset of guitar players who most likely would not find any use out of this program would be those experienced enough and with such a musical ear that they can figure out songs just by hearing them. Even then, however, they may want to double check what they interpreted with the official way to play the song. Similarly, there are many ways to play the same note on the guitar with the various strings. It is possible that there may be an easier way to play a song using one of these alternate methods than what the way the experienced player arrived at.

# Overview

The goal of this project is to create an application to assist guitar players to learn new songs in a highly visual and reactive way. The easiest way to learn a song is to have someone show you how to play it. Since this is not always a readily available option, the next best option is to have a computer show you how to play it and provide as many components possible to facilitate the learning process.

### Input Files

A decision was made early in the development process to use the Guitar Pro file format for a multitude of reasons. The main reason this format was chosen is that it already has a large user base and just about any relatively popular song of the past 60 years has already been composed in at least one version and is available for free. Another large reason is that it is a defined format. This way it isn’t required to reinvent the wheel with a far inferior product. One final major reason for using this format is that it contains a huge amount of supplementary data. For instance, it is capable of storing lyrics, MIDI information, and supplementary tracks. All in addition to the necessary information like notes, rhythm, tempo, and duration.

It is important to note that the GuitarPro file format has gone through four major revisions – GuitarPro v3, GuitarPro v4, GuitarPro v5, and most recently GuitarPro v6. In its current state, this project supports GuitarPro v4. At the time of this project’s inception, version 4 had the most song files available and it was the best documented format I was able to encounter. GuitarPro v3 did not last very long and was quickly replaced by version 4 which had a long run. Version 5 was released in 2005, but version 4 files were still being released because it had become the standard people were used to. It has since started to pick up steam, but version 6 was released in early April, 2010 which will likely curtail any major effort to use version 5. It is because all of this that version 4 seemed to be the best option at the time.

### GuitarPro File Format

A basic overview of how the version 4 file is laid out can be seen below:

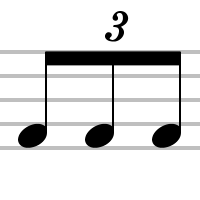
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 | | |  
 | | Version |  
 | |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|  
 | | |  
 | | Tablature |  
 | |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|  
 |Headers| |  
 | | Lyrics |  
 | |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|  
 | | |  
 | | Other Tablature Information |  
 |\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|  
File | | |  
 | | Measures |  
 | |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|  
 | | |  
 | | Tracks |  
 | |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|  
 | | | | |  
 | | | | Note 1 |  
 |Body | | Beat 1 |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|  
 | | | | |  
 | | | | Note i ... |  
 | | Measure-Track |\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|  
 | | Pairs | | |  
 | | | | Note 1 |  
 | | | Beat i ... |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|  
 | | | | |  
 | | | | Note i ... |  
 |\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|  
 | |  
 | Chord Diagrams |  
 |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

The reading of a GuitarPro file is a lengthy and somewhat convoluted affair. Below is a more in-depth outline of how to read this file format:

File Header

1. Version
2. Tablature Information
   1. Title of the song
   2. The subtitle of the song
   3. The interpret of the piece (i.e. the name of the band)
   4. The album this song appeared on
   5. The original author of the song
   6. Who owns the copyright to the song
   7. The name of the author of the GuitarPro file
   8. An “instructional” section (this has always been blank in my experiences)
   9. The notes of the song
   10. Whether or not
   11. The triplet feel of the piece
3. Lyrics
4. Other Tablature information
   1. Tempo
   2. Key
   3. Octave
   4. MIDI channels
      1. MIDI instrument (acoustic guitar, drums, bass, violin, etc)
      2. Volume
      3. Balance
      4. Chorus
      5. Reverb
      6. Phaser
      7. Tremolo
      8. Two blanks (for backwards compatibility with GuitarPro version 3)
   5. Number of measures in the piece
   6. Number of tracks in the piece

File Body

1. Measures
   1. Measure header (1 byte) – These are only binary values to determine if they are present in the following parts of the measure. In other words, parts ‘b’ and following are only evaluated if the corresponding header bit is a 1.
      1. Bit 7 – Presence of a double bar
      2. Bit 6 – Tonality of the measure
      3. Bit 5 – Presence of a marker
      4. Bit 4 – Number of alternate endings
      5. Bit 3 – End of repeat
      6. Bit 2 – Beginning of repeat
      7. Bit 1 – Denominator of the key signature (the ‘4’ in a 3/4 key signature)
      8. Bit 0 – Numerator of the key signature (the ‘3’ in a 3/4 key signature)
   2. Numerator of the key signature (1 byte)
   3. Denominator of the key signature (1 byte)
   4. End of repeat (1 byte)
      1. Number of repeats until the previous beginning repeat
   5. Number of alternate endings (1 byte)
   6. Marker – These are written in two steps
      1. An integer of the marker’s name’s length + 1 followed by the actual string
      2. The marker’s color
   7. Tonality of the measure (1 byte) – Represents a key change of the piece
2. Tracks
   1. Track header (1 byte) – Similar to the measures, this is a boolean value to determine their presence
      1. Bits 7 to 3 – Blank
      2. Bit 2 – Banjo track (4.5 strings)
      3. Bit 1 – 12 stringed guitar track
      4. Bit 0 – Drums track
   2. Name (40 character string)
   3. Number of strings (integer)
   4. Tuning of the strings (table of integers)
   5. MIDI port (integer)
   6. MIDI channel (integer)
   7. MIDI channel effects (integer)
   8. Height of capo (integer) – 0 is present if there is no capo, which is theoretically correct in guitar-speak
   9. Track’s color (color, as by RGB values)
3. Measure-track pairs – The list of beats per track and beats per measure is written like:  
   measure 1/track 1  
   measure 1/track 2  
   …  
   measure 2/track 1  
   ….
   1. Number of beats in the measure-track pair (integer) – After this step, beat information is written as described in the next step
4. Beats
   1. Beat header (1 byte)
      1. Bit 7 – Blank
      2. Bit 6 – True if the beat is empty or is a rest. False if there is a note
      3. Bit 5 – Whether or not the beat is an n-tuplet
      4. Bit 4 – Presence of a mix table change event
      5. Bit 3 – Presence of effects on this beat
      6. Bit 2 – Presence of text on this beat
      7. Bit 1 – Presence of a chord diagram on this beat
      8. Bit 0 – Whether or not this beat is dotted
   2. Status (1 byte) – 0x00 if the beat is empty, 0x02 if it is a rest
   3. Beat duration (1 byte)– In GuitarPro, the default note is an eighth note which is the equivalent of the arbitrary time duration of 1; thusly the following relations can be concluded
      1. -2 – Whole note
      2. -1 – Half note
      3. 0 – Quarter note
      4. 1 – Eighth note
      5. 2 – Sixteenth note
      6. Etc…
   4. N-tuplet – An n-tuplet is a quick succession of notes. For instance, this would be a 3-tuplet: 
      1. An n-tuplet can come in the 3, 5, 6, 7, 9, 10, 11, 12, and 13 variety
   5. Chord diagram – The details of this are explained in the Chord Diagram section
   6. Text – Similar to previous strings, first read an integer of the string length + 1, then the string is read in of the length previously read
   7. Effects on beat – The details of this are explained in the Effects on Beat section
   8. Mix table change event – The details of this are explained in the Mix Table Change Event section
   9. Note – The details of this are explain in the note section
5. Chord diagrams
   1. Header – The value 0x01 is expected to indicate a GuitarPro v4 chord
   2. Sharp (byte) – Determines of the chord is sharp or flat
   3. There are 3 blanks here (1 byte each) – Needed for version 3 compatibility
   4. Root (1 byte)
      1. -1 for customized chords
      2. 0 – C
      3. 1 – C#
      4. 2 – D
      5. Etc…
   5. Major or minor (1 byte)
      1. 0 – Major
      2. 1 – Seven
      3. 2 – Seven major
      4. 3 – Six
      5. Etc…
   6. Nine/Eleven/Thirteen (1 byte)
   7. Bass (integer) – The lowest note of the chord
   8. Diminished/Augmented (integer) – The tonality for the 9/11/13
   9. Add (1 byte) – Determines of an added note is in the chord
   10. Chord name (20 character string)
   11. 2 blank bytes – For backwards compatibility with version 3
   12. Tonality of the 5th (1 byte)
   13. Tonality of the 9th (1 byte)
   14. Tonality of the 11th (1 byte)
   15. Base fret (integer) – The base fret of the chord
   16. Frets (list of 7 integers) – Each integer corresponds to the fret number played on each string
   17. Number of barres (1 byte) – Can contain up to 5 barres
   18. Fret of the barre (list of 5 bytes) – The fret number of each barre
   19. Barre start (list of 5 bytes)
   20. Barre end (list of 5 bytes)
   21. Omission 1, 3, 5, 7, 9, 11, 13 – Gives the notes that are in the chord
   22. Blank (1 byte) – For backwards compatibility with version 3
   23. Fingering (list of 7 bytes) – Describes the fingering used to play the chord
       1. -2 – Unknown
       2. -1 – No finger
       3. 0 – Thumb
       4. 1 – Index
       5. Etc…
   24. Show fingering (1 byte)
       1. 0x01 to display the fingering (this is not implemented in this project)
       2. 0x00 to not show the fingering
6. Effects on beats
   1. Header (1 byte)
      1. Bit 7 – Blank
      2. Bit 6 – Stroke effect
      3. Bit 5 – Tapping, popping, or slapping effect
      4. Bit 4 to 0 – unused, leftover from v3
   2. Header 2 (1 byte)
      1. Bit 7 to 3 – unused
      2. Bit 2 – Tremolo bar
      3. Bit 1 – Pickstroke
      4. Bit 0 – Rasguedo
   3. Tapping/popping/slapping (1 byte)
      1. 0 - None
      2. 1 – Tapping
      3. 2 – Slapping
      4. 3 – Popping (for bass guitar)
   4. Tremolo bar (Bend – to be explained later)
   5. Upstroke (1 byte)
      1. 0 – None
      2. 1 – 128th note
      3. 2 – 64th note
      4. …
      5. 6 – ¼ note
   6. Downstroke (1 byte)
      1. Same as upstroke
   7. Rasugedo – This complements the strokes so it needs no additional information (or byte space)
   8. Pickstroke (1 byte)
      1. 0 – None
      2. 1 – Stroke up
      3. 2 – Stroke down
7. Mix table change event
   1. Instrument (1 byte) – The number of the new instrument
   2. Volume (1 byte) – the new volume
   3. Pan (1 byte) – The new pan value
   4. Chorus (1 byte) – The new chorus value
   5. Reverb (1 byte) – The new reverb value
   6. Phaser (1 byte) – The new phaser value
   7. Tremolo (1 byte) – The new tremolo value
   8. Tempo (integer) – The new tempo value
   9. Volume change duration (1 byte) – the new volume change duration
   10. Pan change duration (1 byte) – The new pan change duration
   11. Chorus change duration (1 byte) – The new chorus change duration
   12. Reverb change duration (1 byte) – The new reverb change duration
   13. Phaser change duration (1 byte) – The new phaser change duration
   14. Tremolo change duration (1 byte) – The new tremolo bar change duration
   15. Tempo change duration (1 byte) – The new tempo change duration
   16. Pseudo-header (1 byte) – Indicates if the changes apply only to the current track (0) or all the tracks (1)
       1. 7 to 6 – Blank
       2. 5 – Tremolo
       3. 4 – Phaser
       4. 3 – Reverb
       5. 2 – Chorus
       6. 1 – Pan
       7. 0 – Volume
8. Note
   1. Header (1 byte)
      1. Bit 7 – Right hand or left hand fingering
      2. Bit 6 – Accentuated note
      3. Bit 5 – Note type (rest, empty, normal)
      4. Bit 4 – Note dynamic
      5. Bit 3 – Presence of effects on the note
      6. Bit 2 – Ghost note
      7. Bit 1 – Dotted note
      8. Bit 0 – Time-independent duration
   2. Note type (short integer)
      1. 0x0100 – Normal
      2. 0x0200 – Ghost note
      3. 0x0300 – Tie note
   3. Note duration (1 byte) – Similar to beat duration, the default is a quarter note with the value of 1 and other notes stemming up and down from that root
      1. -2 – Whole note
      2. -1 – Half note
      3. 0 – Quarter note
      4. 1 – Eighth note
      5. Etc…
   4. N-tuplet (1 byte) – Note: this ties into the note duration byte previously
      1. Can be 3, 5, 6, 7, 9, 10, 11, 12, 13
   5. Note dynamic (1 byte) – A dynamic is how loud or soft to play a note (‘p’ is soft, ‘f’ is loud)
      1. 1 – ppp
      2. 2 – pp
      3. 3 – p
      4. 4 – mp
      5. 5 – mf
      6. 6 – f
      7. 7 – ff
      8. 8 – fff
   6. Fret number (1 byte) – The fret number on which the note is played
   7. Fingering (2 bytes) – Which finger is to play this note. The numerical value is identical to the fingering charter earlier
   8. Effects on note – To be explained next
9. Effects on note
   1. Header 1 (1 byte)
      1. Bit 7 to 5 – Blank
      2. Bit 4 – Grace note
      3. Bit 3 – Let ring
      4. Bit 2 – Slide from the current note (leftover from v3)
      5. Bit 1 – Hammer on or pull off
      6. Bit 0 – Bend
   2. Header 2 (1 byte)
      1. Bit 7 – Blank
      2. Bit 6 – Left hand vibrato
      3. Bit 5 – Trill
      4. Bit 4 – Harmonic note
      5. Bit 3 – Slide from the current note
      6. Bit 2 – Tremolo picking
      7. Bit 1 – Palm mute
      8. Bit 0 – Play note staccato
   3. Bend – To be explained later
   4. Grace note – To be explained later
   5. Tremolo picking (1 byte)
      1. 1 – 8th note
      2. 2 – 16th note
      3. 3 – 32nd note
   6. Slide (1 byte)
      1. -2 – Slide from above
      2. -1 – Slide from below
      3. 0 –No slide
      4. 1 - Shift slide
      5. Etc…
   7. Harmonics (1 byte)
      1. 0 – None
      2. 1 – Natural
      3. 3 – Tapped
      4. 4 – Pitch
      5. Etc…
   8. Trill
      1. Fret (1 byte) – The fret the trill is made with
      2. Period (1 byte) – The period between each note
         1. 0 – Quarter note
         2. 1 – Eighth note
         3. 2 – Sixteenth note
10. Grace notes
    1. Fret (1 byte) – The fret number the grace note is made from
    2. Dynamic (1 byte) – The grace note dynamic. This is coded identically to the Dynamics section
    3. Transition (1 byte)
       1. 0 – None
       2. 1 – Slide
       3. 2 – Bend
       4. 3 – Hammer
    4. Duration (1 byte)
       1. 1 – 32nd note
       2. 2 – 24th note
       3. 3 – 16th note
11. Bends
    1. Type (byte) – Different types of bends (with a tremolo/whammy bar or a string bend with fingers)
       1. Common
          1. 0 – None
       2. Bend specific
          1. 1 – Bend
          2. 2 – Bend and release
          3. 3 – Bend and release and bend
          4. 4 – Prebend
          5. 5 – Prebend and release
       3. Tremolo bar specific
          1. 6 – Dip
          2. 7 – Dive
          3. 8 – Release up
          4. 9 – Inverted dip
          5. 10 – Return
          6. 11 – Release (down)
    2. Bend height (integer) – It is 100 per tone and increases/decreases by quarter tone
       1. Normal – 0
       2. Quarter – 25
       3. Half – 50
       4. …
       5. Three tones – 3000
    3. Number of points (integer) – The number of points used to display the bend
    4. List of points – The list of points from the previous integer
       1. Absolute time position (integer) – Gives the point position from the previous point
       2. Vertical position (integer) – Similar to bend height, 100 per tone and increases/decreases by quarter tone
       3. Vibrato (1 byte) – Determines how far to play the section with different vibrato types
          1. 0 – none
          2. 1 – fast
          3. 2 – average
          4. 3 – slow
12. Chord diagrams
    1. Number of chords (integer) – The number of chords to write in the chord diagram list (this is not implemented in this program)

# Project Requirements

Due to the fact that I did not have a client for this project, the requirements for completion were somewhat flexible. I stayed with the main theme of the project – namely to provide and application to that would assist a guitar player in learning new songs. In my mind, there were three main milestones that needed to be met for this to be considered a success:

1. Represent hand movements, fingering, and strumming in a meaningful and easy-to-read way
2. Provide audio accompaniment to go along with the musical tablature to assist with strum/pick patterns
3. Allow the guitar player to easily seek third-party assistance via YouTube.

The project should capable of playing on any Windows XP or later (as of this writing) and either x86 or x64 architecture. It does not require a whole lot of processing power to use this software; however a screen resolution of 1024 by 786 is highly required to experience the entirety of the application.

# System Design

This application was written in the C# programming language utilizing .NET 3.5. It is entirely object-oriented and makes use of just about every fundamental aspect of object oriented programming languages such as interfaces and abstract classes.

### User Interface

The entire application lies in one frame and is all graphics-oriented. This main form contains everything the user needs to see and interact with. Below in Figure 1 is a screenshot of the application.

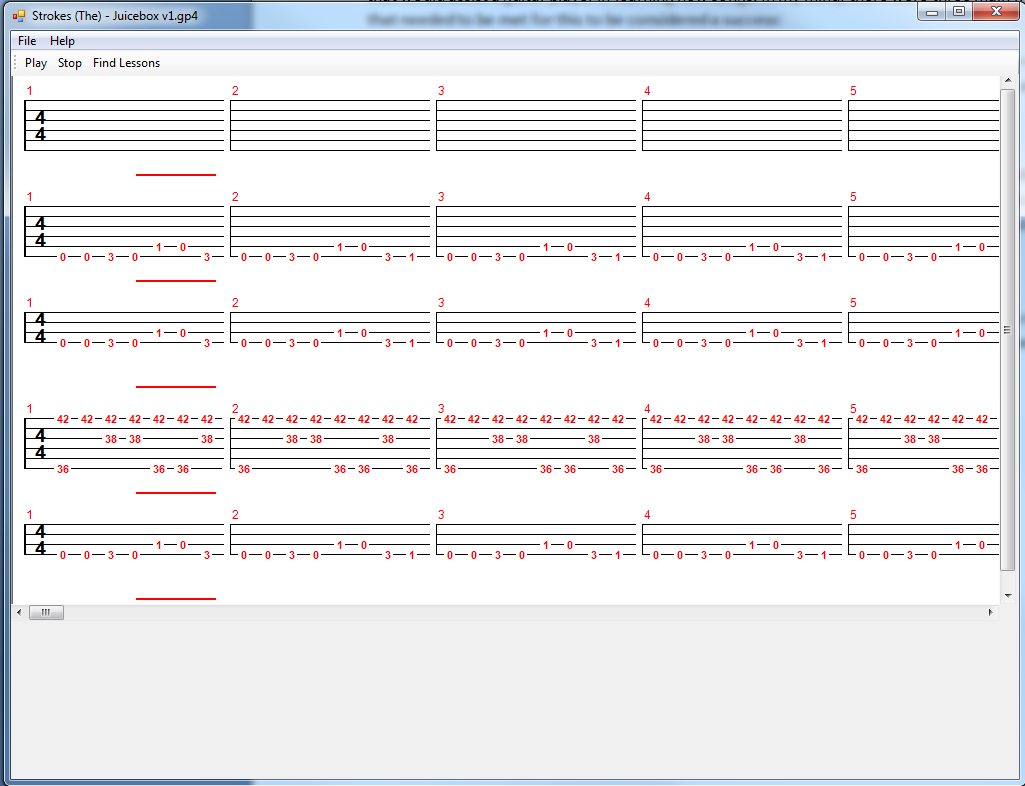


Figure – Main form and tablature

The main form contains a Panel named SongInternalFrame which houses everything in the scroll bars. That panel contains a very large panel named SongPanel which houses all the graphics (tracks, measures, bars, etc). The SongPanel houses each track (named TrackPanel) and each track houses a BarPanel which is essentially one measure. That BarPanel draws the measure lines and has fret numbers, time signatures, music cursor, and any anything else you see on the screen. The basic layout for the graphical portion is as follows:

Main form  SongInteralFrame  SongPanel  TrackPanel  BarPanel  music lines, time signature, notes, music cursor, etc.

* The ‘Play’ button begins playback of the song if a song has been loaded. It does nothing if a song is already playing or if a song has not been loaded
* The ‘Stop’ button ends playback of the song if it is already playing. If the song is not yet playing or if a song has not yet been loaded, it does nothing
* The ‘Find Lessons’ button will open up a browser and bring you to YouTube. If a song is loaded, it will automatically search for the name of the song and artist along with the word “lesson” to attempt to find a guitar lesson of that song. If no song has been loaded, it will simply go to the YouTube homepage.
* The file menu contains two options: Open and Exit. The Open option will bring up a file dialog box asking for a GuitarPro file. Once a GuitarPro file is open, it will then immediately request an MP3 to accompany the GuitarPro file with for audio playback. The Exit option will close the program.
* The help menu contains three options: Instructions, Colors, and About. Instructions will give basic how-to regarding the application. Colors will tell users what each note color means. About gives some information about the author of the program.

### Data Structures

This application did not require complex data structures. The design utilizes objects and classes heavily, occasionally using built-in Lists and arrays. Classes TripletHash and TripletValue use hash tables, but they are defined statically and use the built-in C# Hashtable class. They are merely used as a convenience to look up values with. Because they are never changed and values are only added once, their runtime is not worth talking about.

### Class Diagrams

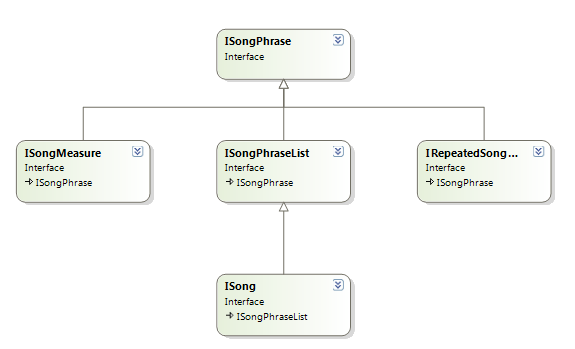
There are a number of class diagrams that are noteworthy. For the most part, however, classes are independent and called from throughout the entire project. Below you will find various implementation trees.

Figure - SongPhrase diagram

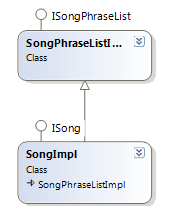
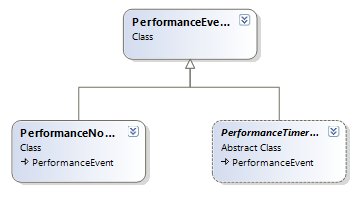


Figure - SongPhraseListener diagram

Figure - PerformanceEvent diagram

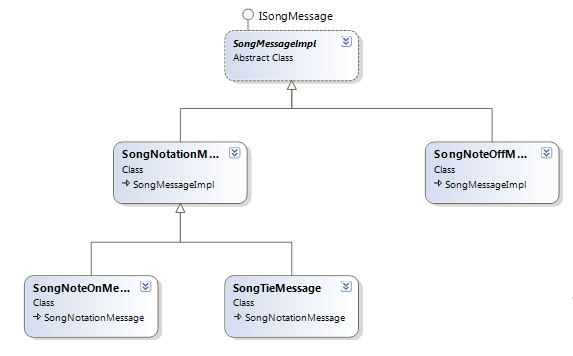


Figure - SongMessage diagram

### Algorithms

Reading in the GuitarPro file was one of the trickiest parts of the implementation. Extreme care must be taken to ensure that it reads in your file correctly and works correctly for all possible combinations of the format. The actual file format is not set in stone. In other words, for instance, byte 2910 will not always be note 100. Everything is dynamic and must be read in sequentially because, with the exception of the file header, all other areas of the piece can, and will, fall into different areas of the file.

Similarly, the actual painted graphics are nested in 6 deep in classes. Various panels and objects within panels must be called and drawn in the correct order or else a big, white background panel could potentially be drawn on top of everything else rendering it useless. The painting actually occurs as the inherited, event driven Panel class’s OnPaint method.

# Software Development Process

Because I knew my requirements would likely change slightly over time, I opted to do a prototyping methodology with myself. It wasn’t a true prototyping method since I always saw everything being created as I created it and I didn’t have to “show off” to myself. I did, however, take small steps, see where I was at and where I wanted to be, and revised my plans accordingly.

### Testing and Debugging

Even though I realize it is a bad idea, I ended up adopting a test-as- you-go methodology. I would attempt to implement a feature, test it until it appeared to work and then move on to the next feature. Because of this, it is difficult to peg down exactly how much time was spent debugging and testing versus how much time was spent implementing.

One of the larger bugs I faced resulted from my inexperience with the C# language. Because the GuitarPro file is a binary file, it is read in byte by byte. In Java, bytes are signed by definition, however in C#, they are unsigned by default. I spent over a week trying to figure out why the file reader kept crashing and throwing errors when I was finally able to track down the fact that C# has a special sbyte primitive that could be used in lieu of the normal byte primitive to solve this problem.

In a very similar error, again due to my inexperience with C#, I learned the hard way that, while array contents are passed by reference, reassigning the array would not be seen by the caller. This caused some trouble, though not as much as the signed-byte issue.

My testing cases consisted of a large set of GuitarPro files that contained as many different aspects of the file format as possible. After each feature adjustment or addition, I made sure that each of those files still loaded and played as they should.

### Work Breakdown

At the beginning of this project, I estimated that I could spend roughly 12 hours per week on this project. This ended up being pretty true to the case, though the addition of the GuitarPro file format put me behind schedule quite a bit. It is impossible to say whether or not sticking with my original implementation plans would have affected the output of the project timeline for the better, but as is indicated by the length and complexity of GuitarPro earlier, it took a large chunk of time to implement; however I believe that in the long run it was worth the extra time. There is so much information available to the programmer and application to add a multitude of other features. Sadly there was not enough time to implement them during the course of this semester.

# Results

Overall I am very pleased with the outcome of the project. I wish I had more time to implement additional features and get the application to how I had pictured it in my head at the beginning of the semester. I realize, however, that projects of this scale are time consuming and road bumps are almost always encountered.

Because of the nature of the application, it is difficult to show screenshots of it in action since it would just be the same thing over and over again – a series of tracks with colored numbers on them. For an example of what it looks like, refer to figure 1.

### Summary and Conclusions

Guitar Trainer was created by a guitar enthusiast for guitar enthusiasts. In that regard I believe it was successful. Given a few more additions to the application (namely MIDI playback ability and a slowdown feature), I believe it will be exceptionally useful.

Unfortunately the sad truth of the matter is that, during the creation of this project, I encountered the website [www.songsterr.com](http://www.songsterr.com) which is nearly identical to this application, only much better and polished. By the dates on the comments, it would appear that it has been around for several years and has had that extra time to polish their presentation. The only downside of the site is that, while you can use it for free, the “premium” features cost money to utilize. One of those features is a slow-down option which can cut the playback in half. While that is a good start, for me, being the novice player, a slider to control the playback speed would be much more useful – specifically to be able to slow the playback down beyond half, to quarter or even eighth speed.

It is for this reason that I plan to continue the development of Guitar Trainer and to perhaps one day a release I’m confident enough about to release out onto the internet. I do not foresee that time in the immediate future, but more realistically in the not-so-distant future.

# Appendix A: User Manual

### Minimum System Requirements

Windows XP, Vista, or 7

x86 or x64 architecture

512mb memory (estimated)

Pentium 4 or later (estimated)

Video resolution of 1024x786 or higher

### Installation/Running

No installation is necessary. Simply run the application file to start the program. Once the application is running, select File -> Open. This will bring up a file dialog box. Navigate to where your GuitarPro4 (\*.gp4) files are located. If you do not have any, you can visit [www.gprotab.net](http://www.gprotab.net) and download any version 4 file. Once you have selected and opened a GuitarPro file, a second file dialog box will open up and ask you to select an MP3 file to accompany the GuitarPro file. Select the actual MP3 that the GuitarPro file is mimicking.

Afterwards, the file will load and be ready for playback. You may press the ‘Play’ button at any time to start file playback. Following the red music cursor will inform you what part of the song is currently playing. If at any time you would like to stop the song, simply press the ‘Stop’ button in the tool bar. At any time you may open a different GuitarPro file and MP3 combination to play along with a different song.

If at any time you need guitar assistance with the song (strumming patterns, for instance), press the ‘Find Lessons’ to be redirected to a YouTube search for lessons of the song you currently have open.

Additional information can be found in the program under the help menu. Instructions on how to open files and being playback, note coloration, and information about the author of the program are all located here.

Exiting the application is as easy as clicking the usual Windows red ‘X’ in the top right corner or choosing File -> Exit from the File menu.