April 2015

Voice Controlled Music Sequencer

Victoria Jade Valcour
Worcester Polytechnic Institute

VOICE CONTROLLED MUSIC SEQUENCER

Interactive Qualifying Project Report completed in partial fulfillment

of the Bachelor of Science degree at

Worcester Polytechnic Institute, Worcester, MA

Submitted to: Professor Bianchi

Ma'at Ford _____

Victoria Valcour _____

(date of submission)

_____

Abstract

This project served as a proof-of-concept to determine the validity of a voice-controlled music sequencer. Working with Digital Performer and Dragon Dictate software, the project explored the history of voice-recognition technology, and the feasibility of this technology within a music sequencing environment.

Table of Contents

Executive Summary

The objective of this IQP was to develop a voice-controlled music sequencer prototype that would serve as a proof-of-concept to determine the validity of developing a more sophisticated model. A voice-controlled sequencer would allow the user to initiate and realize a number of editing and data manipulation procedures through voice commands. Execution of these commands will help facilitate the preparation of large and involved music sequences that have traditionally required time intensive editing strategies. The speech-recognition technology would allow users to easily alter properties of songs without having to use keystrokes associated with their desired command. Voice-activated music sequencers are also necessary for increasing accessibility for disabled users, who experience difficulties utilizing traditional sequencers due to visual or other physical impairments. Ultimately, a voice controlled music sequencer will make the editing procedures more representative of the interaction that occurs between a conductor and a live acoustic orchestra. This technology is necessary to increase versatility in terms of music production, as it provides more ways in which producers may create and implement innovative musical ideas and concepts. In order to meet the primary objective, the project team created a crude blueprint for a voice activated interface by linking voice software and macros (recorded actions that perform an automated computer action).

At the onset of this IQP, the project team was provided and presented with specific software to use by advisor Professor Bianchi: AppleScript, Digital Performer 7.22, and Mac speakable items. However, only certain items in AppleScript are fully compatible with the commands in Mac speakable items. Thus, the project team decided to make use of different software to meet the objective, specifically, Dragon Dictate. Dragon Dictate is a speech

recognition software that allows the user to perform tasks on the computer with custom voice commands.

The project team experimented with different types of software to fulfill the automation portion of the prototype. Mouse recording software recognizes movements that the user performs with the computer mouse and compiles them into a file that is executable. By linking this software and Dragon Dictate, the user is able to easily modify musical compositions by simply saying the name of a file that contains the respective command to be performed.

Introduction

In computer science, speech recognition is the conversion of speech to a computer command or on-screen text. In order to complete this conversion, a computer must complete a series of complex steps:

1. When a user speaks, their voice creates vibrations in the air. A computer samples these by taking precise measurements at frequent intervals of the vibrations' waves.
2. The computer filters the sampled sound to remove unwanted noise, and adjusts it to a constant volume and speed that will match the template sound sample stored in the system's memory.
3. The sound is divided into small segments that are matched with known phoneme, representations of sounds that humans make to form meaningful expression.
4. The software examines the phonemes in the context of others around them, running them through a complex statistical model and comparing them to a large library of words, phrases, and sentences. Thus, the software is able to determine what the user was saying, and as a result, executes a command (Grabianowski).

Throughout the last sixty years, speech recognition software has evolved and emerged with applications in various industries that include healthcare, military, and telephony. In the 1950s, the first speech recognition systems were only able to process digits. In 1952, Bell Laboratories designed the "Audrey" system, which recognized digits spoken by a single user. Ten years later at the World Fair, IBM revealed its "Shoebox" system, which was able to understand sixteen spoken English words.

In the 1970s, interest from the United States Department of Defense allowed speech recognition technology to make major strides -- perhaps most notably in the funding of Carnegie Mellon's "Harpy" system. Harpy was able to process approximately 1,011 words, the vocabulary of an average three year old. Additionally, Bell Laboratories was able to introduce the first system that was able to interpret the voices of multiple users.

In the 1980s, the hidden Markov model was developed. This was a new statistical method that considered the probability of unknown sounds being words, rather than simply looking for sound patterns. With this model, speech recognition began to emerge in commercial applications and business settings.

In the 1990s, speech recognition software became viable for ordinary users with the aid of more efficient computer processors. The first consumer speech recognition product, Dragon Dictate, was launched in 1990. This software was able to recognize a user's continuous speech after forty-five minutes of training. In 1996, BellSouth produced the first dial-in interactive speech recognition system, "Val", which gave users information based on what they said into a telephone.

In the 2000s, speech recognition emerged into the mobile phone industry with the arrival of Google Voice Search app for the iPhone. This was particularly significant for the development of speech recognition, as the tiny size of mobile devices provided incentive to develop more advanced and alternative user-input methods (Pinola).

Methodology

The project team began by experimenting with an application that detects keystrokes, "Keyboard and Mouse Recorder" (KAMR), developed by AlphaOmega Software. The project team recorded basic keystroke commands (i.e. start, stop, play), all of which were executed properly when played back in Digital Performer. The project team tried to further this method by naming these executable files to their respective commands (i.e. a recording that clicks the play button or activates the play keystroke is named "play") and by voicing the name of the file by using Speakable Items. However, this method was crude and tedious to perform for several reasons. Whenever the keystroke was pressed to begin recording, the operator had to immediately perform the keystroke or the mouse command. No matter how swiftly the action was performed during the recording, there was a significant delay. Additionally, the version of KAMR that the project team obtained was a trial version, so actions could only be recorded for a certain length in time. The project team's advisor, Professor Bianchi, was willing to purchase the full version of the software, but eventually it was deduced that this method was inadequate for an eventual voice-activated music sequencer.

While the project team initially began utilizing AppleScript and Speakable Items software, it was ultimately decided to make use of other software applications. This decision was made primarily due to AppleScript being a somewhat complex programming language that takes significant time to learn. With little knowledge in AppleScript software, the project team felt that time would be better spent generating results as opposed to learning a new programming language. Furthermore, AppleScript and OS X's built in Speakable Items worked much more

cooperatively with proprietary Apple Applications (Automator, iMovie, Safari, Finder, iMessage, etc.).

In further research on an unaffiliated Digital Performer (DP) forum concerning scriptability, the project team learned that DP is indeed scriptable in AppleScript. However, DP is not a very "well-behaved" application. In the AppleScript editor, a dictionary of commands will be published for each scriptable application. While DP does have a command dictionary, it contains only one command titled "doscript", with no script syntax. The project team examined a 1998 MQP in which a team created an actual voice-activated music sequencer. In doing so, the project team noted that the team used a voice speaking software titled "Dragon Naturally Speaking" developed by Nuance Communications for Windows. Because this project is performed completely on Mac OS X, the project team decided to use the Macintosh alternative to Dragon Naturally Speaking, Dragon Dictate.

The project team obtained a copy of Dragon Dictate and installed it on one of the Apple computers in WPI's Riley Lab. Dragon Dictate is a speech-to-text software that transcribes what is verbally dictated to the computer by the user. Dragon Dictate contains a tutorial for new users on how to use the software. Users create their own voice profiles when Dragon is turned on. In order to obtain optimal performance out of Dragon Dictate, our team read through all of the passages so the software could get acclimated to our voices.

The project team tested Dragon Dictate with other applications (TextEdit, Finder, and Safari) in order to see the feasibility of using it eventually with Digital Performer. The project team articulated simple commands ("delete", "go forward", "go back", "open", and "close") at reasonable, intelligible speeds to ensure that the software understood the input clearly. Dragon

Dictate understood read commands with an approximate 90% success rate. If there is any error, the user may state “scratch that” and Dragon will provide the user with a list of possible phrases that may match what was spoken. The user would simply select one of these phrases and the information from the selected phrase would be sent to the user’s Dragon Profile for the software to “learn” from the user’s voice. The project team verbalized other commands (i.e. file, edit, format, etc.) to see if Dragon would execute them. There were some commands that Dragon would not execute, due to Dragon misinterpreting a command for another word. Ultimately, Dragon performed sufficiently with the commands that were provided off of the Dragon Dictate website for all three applications that the project team worked with.

After a few weeks of experimenting with Dragon’s interface and using its commands, the project team began to integrate Dragon Dictate with Digital Performer. The initial plan of our proof of concept was to use KAMR software to record an action, save the file, and execute the command via user voice. This proved to be a very tedious process. Additionally, Dragon Dictate was only able to open applications, and not files. Each time the project team attempted to open a file with Dragon, another window with information about that file would open. Another problem that the project team encountered using KAMR was that even when mouse recording files were executed, Digital Performer needed to be the current application running over all others. These setbacks ultimately proved that the project team’s developed concept, at that point, was not feasible.

While Dragon would not work with the recorded files for Keyboard and Mouse Recorder (KAMR), Speakable Items would. After placing some command files in the KAMR folder and resetting the Speakable Items application, the commands were successfully executed. Since the voice-to-interface engine in Speakable Items is not as complex as Dragon, commands needed to

be enunciated loudly and clearly. While this was significant progress, the project team continued to look for software that allowed the user to exert their voice less.

After researching ways to integrate Dragon with other applications, the project team found that Dragon came with its own scripting software. In Dragon Dictate, there is a list of applications with custom commands (accessible through the “commands” option) and a list of parameters for any command that is created. If a user wanted to create a command that opened the DP Effects Window, the user would create a new application context for Dragon Dictate to receive, and then adjust the command parameters. This command script editor offers an extensive amount of options in how the user chooses to execute the command.

Results

In the command list for Dragon, the project team selected Digital Performer and created a new command titled “drum roll”. The project team selected the respective keystroke that opened the drum roll command, voiced “drum roll” to the software, and the drum roll window successfully opened. This simple test serves as proof-of-concept for a voice-activated music sequencer. Provided with the versatility and simplicity of Dragon Dictate’s scripting program, the team found it much easier to communicate with the interface for Digital Performer.

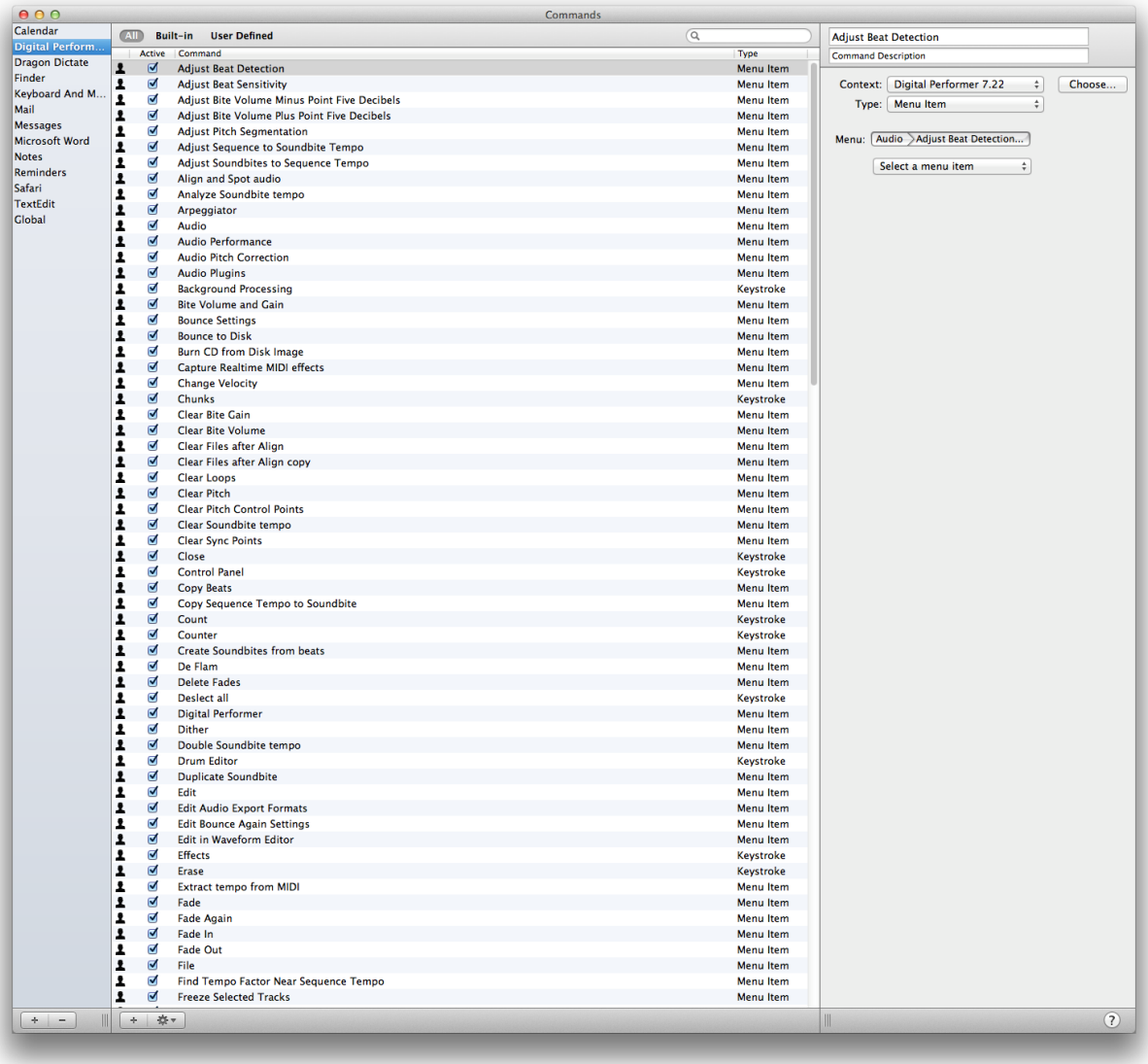
To further test the capabilities of Dragon’s scripting with Digital Performer, the project team began to use the list of Digital Performer commands that were executable by keystroke and integrating them into Dragon Dictate. The new voice commands that the project team created worked seamlessly in Digital Performer.

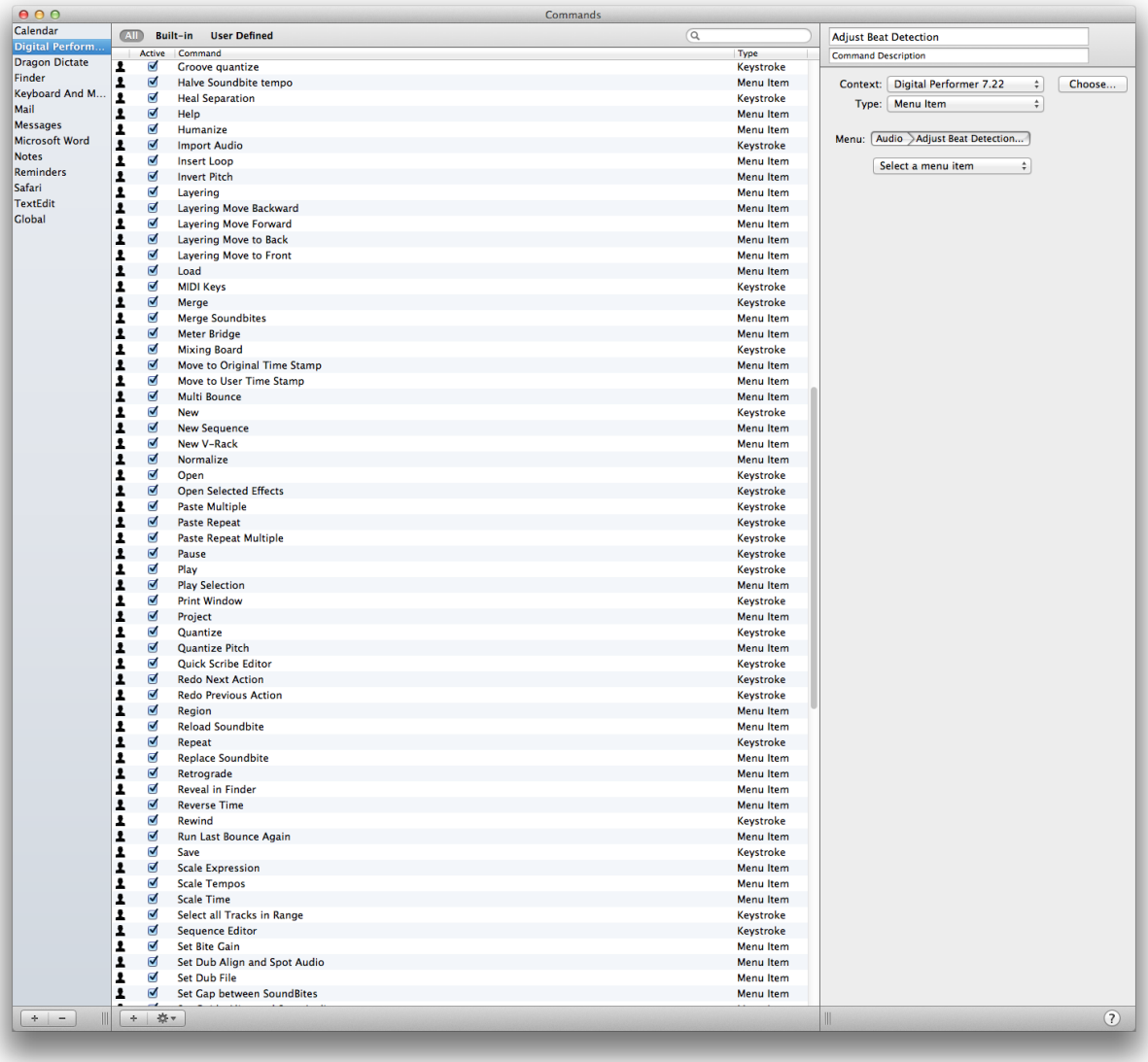
The next step was to script the menu bar for Digital Performer. Dragon also has an option in the command scripting window for the user to activate items on the menu bar. When testing the menu items in Dragon Dictate, there were some interesting points to note. When the user voices one menu item, the menu item needs to be repeated in order to close it. The user cannot vocally switch from one menu item to another while one is currently open. If the user continues to switch while one menu is open, a glitch will occur in which Dragon will open the recited menu for a split second.

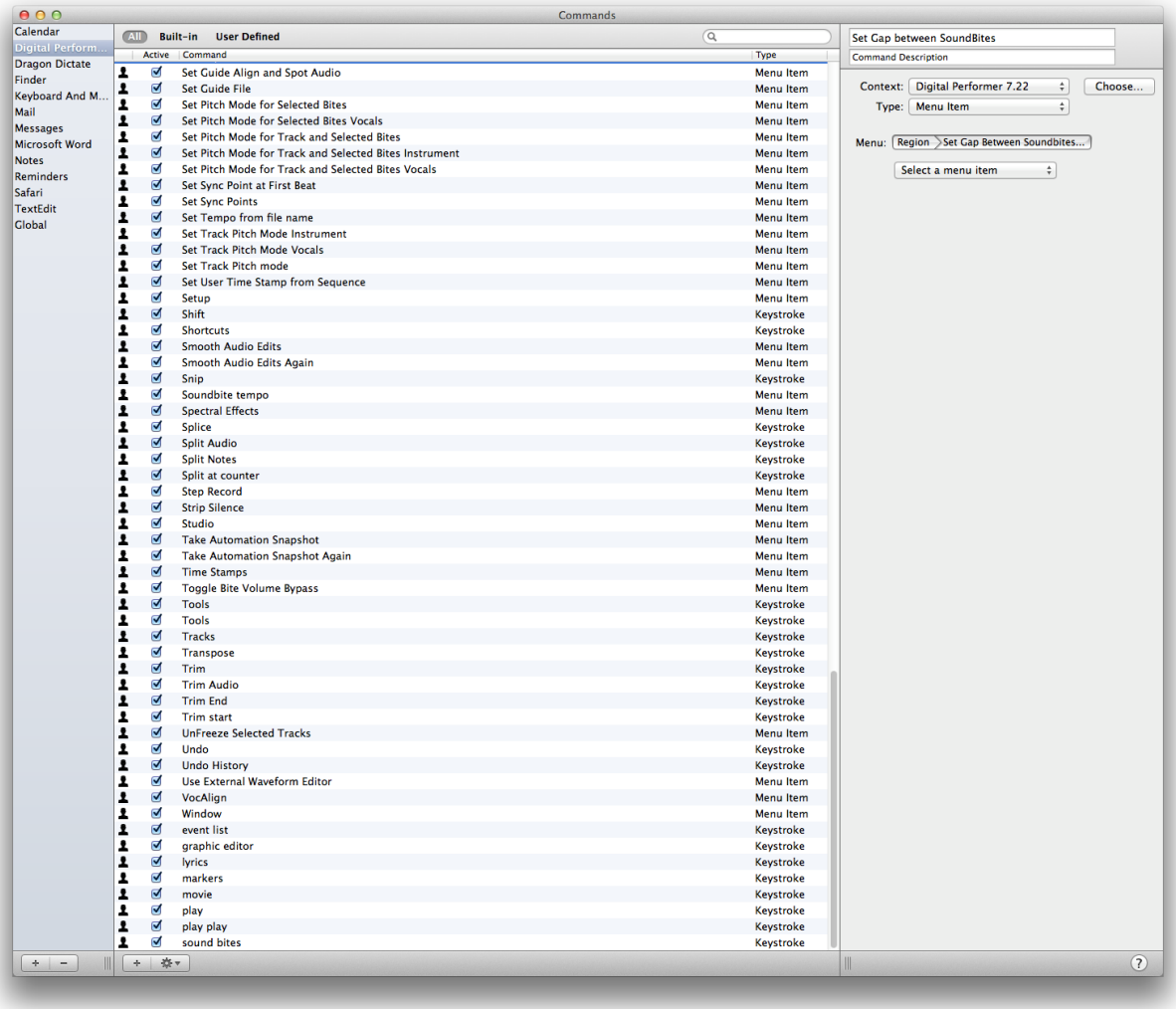
Another note from the project team’s developed method is the manner in which a user may modify music files. The project team was provided with a sample of “Adagio for Strings”, composed by Samuel Parker. When examining the music overlay for Adagio for Strings, the project team wanted to test how well the voice activation features would work with the song

itself, specifically with a command the project team created called “retrograde” (For a full list of commands that the project team created, see Appendix A). Retrograde reverses the order of musical notes, so that when played, the notes are read and played the same backwards as they are naturally played forwards. The project team selected a piece of notes from the song and voiced the command “retrograde”. As expected, the selected piece in the composition reversed the notes.

Appendix A


Commands
Calendar
Digital Perform...
Dragon Dictate
Finder
Keyboard And M...
Mail
Messages
Microsoft Word
Notes
Reminders
Safari
TextEdit
Global
All
Built-in
User Defined
Active
Command
Type
Adjust Beat Detection
Menu Item
Adjust Beat Sensitivity
Menu Item
Adjust Bite Volume Minus Point Five Decibels
Menu Item
Adjust Bite Volume Plus Point Five Decibels
Menu Item
Adjust Pitch Segmentation
Menu Item
Adjust Sequence to Soundbite Tempo
Menu Item
Adjust Soundbites to Sequence Tempo
Menu Item
Align and Spot audio
Menu Item
Analyze Soundbite tempo
Menu Item
Arpeggiator
Menu Item
Audio
Menu Item
Audio Performance
Menu Item
Audio Pitch Correction
Menu Item
Audio Plugins
Menu Item
Background Processing
Keystroke
Bite Volume and Gain
Menu Item
Bounce Settings
Menu Item
Bounce to Disk
Menu Item
Burn CD from Disk Image
Menu Item
Capture Realtime MIDI effects
Menu Item
Change Velocity
Menu Item
Chunks
Keystroke
Clear Bite Gain
Menu Item
Clear Bite Volume
Menu Item
Clear Files after Align
Menu Item
Clear Files after Align copy
Menu Item
Clear Loops
Menu Item
Clear Pitch
Menu Item
Clear Pitch Control Points
Menu Item
Clear Soundbite tempo
Menu Item
Clear Sync Points
Menu Item
Close
Keystroke
Control Panel
Keystroke
Copy Beats
Menu Item
Copy Sequence Tempo to Soundbite
Menu Item
Count
Keystroke
Counter
Keystroke
Create Soundbites from beats
Menu Item
De Flam
Menu Item
Delete Fades
Menu Item
Deslect all
Keystroke
Digital Performer
Menu Item
Dither
Menu Item
Double Soundbite tempo
Menu Item
Drum Editor
Keystroke
Duplicate Soundbite
Menu Item
Edit
Menu Item
Edit Audio Export Formats
Menu Item
Edit Bounce Again Settings
Menu Item
Edit in Waveform Editor
Menu Item
Effects
Keystroke
Erase
Keystroke
Extract tempo from MIDI
Menu Item
Fade
Menu Item
Fade Again
Menu Item
Fade In
Menu Item
Fade Out
Menu Item
File
Menu Item
Find Tempo Factor Near Sequence Tempo
Menu Item
Freeze Selected Tracks
Menu Item
Adjust Beat Detection
Command Description
Context:
Digital Performer 7.22
Choose...
Type:
Menu Item
Menu:
Audio
Adjust Beat Detection...
Select a menu item


Commands
Calendar
Digital Perform...
Dragon Dictate
Finder
Keyboard And M...
Mail
Messages
Microsoft Word
Notes
Reminders
Safari
TextEdit
Global
All
Built-in
User Defined
Active
Command
Type
Groove quantize Keystroke
Halve Soundbite tempo Menu Item
Heal Separation Keystroke
Help Menu Item
Humanize Menu Item
Import Audio Keystroke
Insert Loop Menu Item
Invert Pitch Menu Item
Layering Menu Item
Layering Move Backward Menu Item
Layering Move Forward Menu Item
Layering Move to Back Menu Item
Layering Move to Front Menu Item
Load Menu Item
MIDI Keys Keystroke
Merge Keystroke
Merge Soundbites Menu Item
Meter Bridge Menu Item
Mixing Board Keystroke
Move to Original Time Stamp Menu Item
Move to User Time Stamp Menu Item
Multi Bounce Menu Item
New Keystroke
New Sequence Menu Item
New V-Rack Menu Item
Normalize Menu Item
Open Keystroke
Open Selected Effects Keystroke
Paste Multiple Keystroke
Paste Repeat Keystroke
Paste Repeat Multiple Keystroke
Pause Keystroke
Play Keystroke
Play Selection Menu Item
Print Window Keystroke
Project Menu Item
Quantize Keystroke
Quantize Pitch Menu Item
Quick Scribe Editor Keystroke
Redo Next Action Keystroke
Redo Previous Action Keystroke
Region Menu Item
Reload Soundbite Menu Item
Repeat Keystroke
Replace Soundbite Menu Item
Retrograde Menu Item
Reveal in Finder Menu Item
Reverse Time Menu Item
Rewind Keystroke
Run Last Bounce Again Menu Item
Save Keystroke
Scale Expression Menu Item
Scale Tempos Menu Item
Scale Time Menu Item
Select all Tracks in Range Keystroke
Sequence Editor Keystroke
Set Bite Gain Menu Item
Set Dub Align and Spot Audio Menu Item
Set Dub File Menu Item
Set Gap between SoundBites Menu Item
Adjust Beat Detection
Command Description
Context: Digital Performer 7.22
Choose...
Type: Menu Item
Menu: Audio Adjust Beat Detection...
Select a menu item


Commands
Calendar
Digital Perform...
Dragon Dictate
Finder
Keyboard And M...
Mail
Messages
Microsoft Word
Notes
Reminders
Safari
TextEdit
Global
All
Built-in
User Defined
Active
Command
Type
Set Guide Align and Spot Audio Menu Item
Set Guide File Menu Item
Set Pitch Mode for Selected Bites Menu Item
Set Pitch Mode for Selected Bites Vocals Menu Item
Set Pitch Mode for Track and Selected Bites Menu Item
Set Pitch Mode for Track and Selected Bites Instrument Menu Item
Set Pitch Mode for Track and Selected Bites Vocals Menu Item
Set Sync Point at First Beat Menu Item
Set Sync Points Menu Item
Set Tempo from file name Menu Item
Set Track Pitch Mode Instrument Menu Item
Set Track Pitch Mode Vocals Menu Item
Set Track Pitch mode Menu Item
Set User Time Stamp from Sequence Menu Item
Setup Menu Item
Shift Keystroke
Shortcuts Keystroke
Smooth Audio Edits Menu Item
Smooth Audio Edits Again Menu Item
Snip Keystroke
Soundbite tempo Menu Item
Spectral Effects Menu Item
Splice Keystroke
Split Audio Keystroke
Split Notes Keystroke
Split at counter Keystroke
Step Record Menu Item
Strip Silence Menu Item
Studio Menu Item
Take Automation Snapshot Menu Item
Take Automation Snapshot Again Menu Item
Time Stamps Menu Item
Toggle Bite Volume Bypass Menu Item
Tools Keystroke
Tools Keystroke
Tracks Keystroke
Transpose Keystroke
Trim Keystroke
Trim Audio Keystroke
Trim End Keystroke
Trim start Keystroke
UnFreeze Selected Tracks Menu Item
Undo Keystroke
Undo History Keystroke
Use External Waveform Editor Menu Item
VocAlign Menu Item
Window Menu Item
event list Keystroke
graphic editor Keystroke
lyrics Keystroke
markers Keystroke
movie Keystroke
play Keystroke
play play Keystroke
sound bites Keystroke
Set Gap between SoundBites
Command Description
Context: Digital Performer 7.22
Choose...
Type: Menu Item
Menu: Region Set Gap Between Soundbites...
Select a menu item

Works Cited

"DP AppleScript Documentation." *Motunation*. N.p., 21 Nov. 2004. Web. 23 Aug. 2013.

"Dragon Dictate for Mac Command Cheat Sheet." *Dragon Dictate*. Nuance, n.d. Web. 23 Aug. 2013.

Grabianowski, Ed. "How Speech Recognition Works." *HowStuffWorks*. N.p., n.d. Web. 23 Aug. 2013.

Pinola, Melanie. "Speech Recognition Through the Decades: How We Ended Up With Siri."*ComputerWorld*. IDG Magazines Norge, 11 Mar. 2011. Web. 23 Aug. 2013.