7. MIDI Tunes

Glen Bull, Jo Watts, and Rachel Gibson

The *Musical Instrument Digital Interface* (MIDI) is a technical standard that describes ways in which electronic musical instruments can connect to one another. It was extended to include communications between a MIDI instrument and a computer. Consequently, thousands of digital songs are now available in MIDI format.

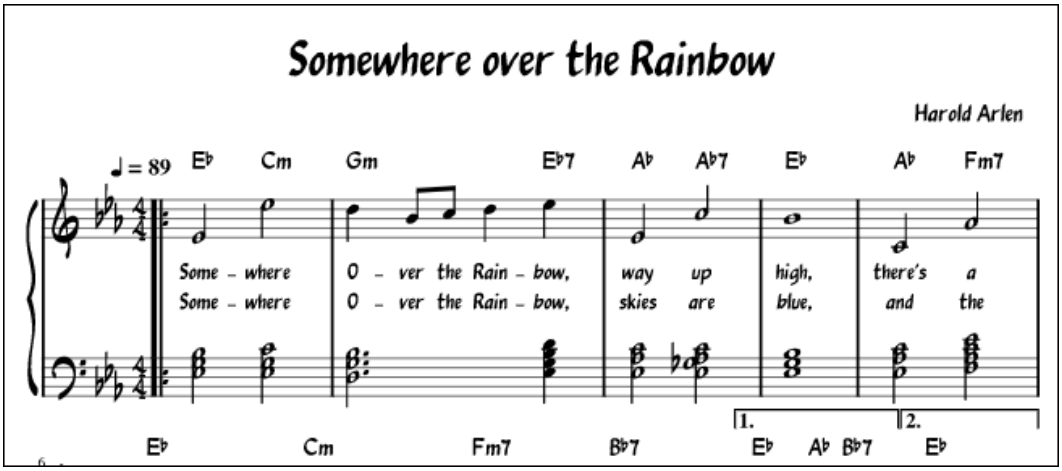
The MIDI standard can extend the capabilities of TuneScope in two ways:

1. The melody of a MIDI file can be extracted and imported into TuneScope.

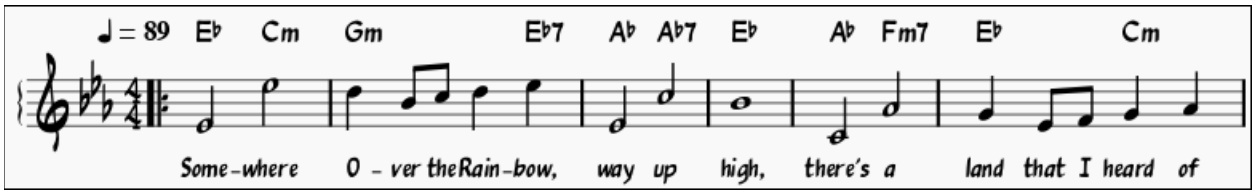
2. An external MIDI keyboard can be used to record a list of notes in TuneScope.

7.1 Extracting a Melody from a MIDI File

The example below illustrates the score for a song. The melody is shown in the top musical staff (the treble clef) while accompanying chords are shown in the bottom staff (the bass clef).



The process of translating the melody into a list of TuneScope notes is simplified if all of the tracks except for the melody are removed from the score. This can be done in a number of different music notation programs such as NoteFlight and MuseScore. A description of how this might be done in MuseScore is provided in the appendix. The resulting score should now consist of just the melody.



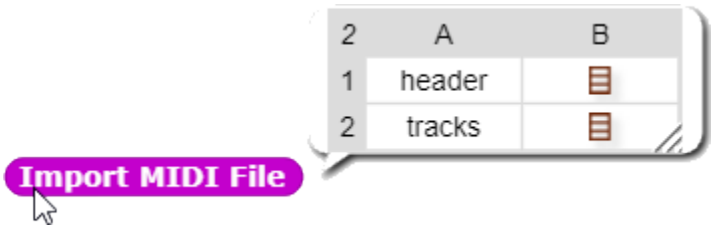
The melody track can then be examined by converting it to a human readable form. JavaScript is a language frequently used for web development. (Snap*!*, for example, is written in JavaScript.) JavaScript Object Notation (JSON) files are a human readable data format. There are a number of tools for converting MIDI files to JSON format. A description of how this can be done using a Tone.js conversion tool is provided in the appendix.

When the MIDI file is converted to JSON format, the MIDI information is displayed in a text format. The file has a header with information such as the time signature and the beats per minute, and tracks of notes. Each note in a track has information such as the name of the note (D#4) and its duration.


```
{
  "duration": 1.2794976020833333,
  "durationTicks": 911,
  "midi": 63,
  "name": "D#4",
  "ticks": 0,
  "time": 0,
  "velocity": 0.6299212598425197
},
```


7.2 Importing a MIDI File into Snap*!*

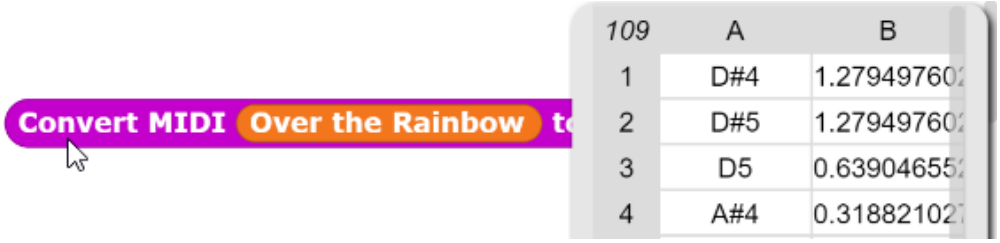
If a MIDI file is converted to JSON format with a tool such as the Tone.js MIDI converter, the resulting JSON file can be imported into Snap*!* by dragging the JSON file from the desktop to the script area. The **Import MIDI File** block saves a step by converting a selected MIDI file into JSON format. Clicking this block produces a dialog box that requests a MIDI file. Once a MIDI file is selected, the output of the reporter block displays a table in JSON format. The header and the tracks described in the preceding section appear in this table.



The output of the reporter block can be assigned to a variable.

7.3 Converting a MIDI File to TuneScope Format

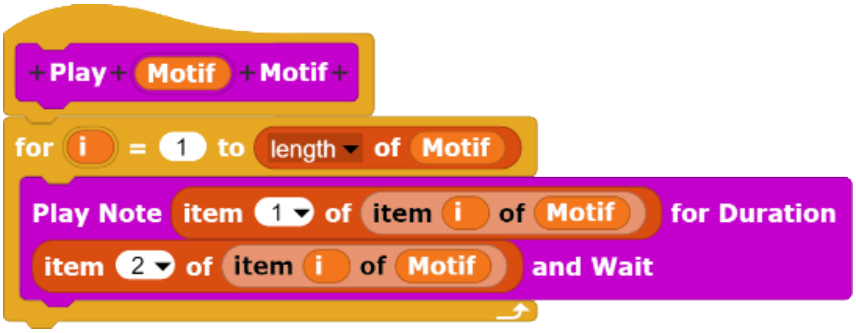
The **Convert MIDI to TuneScope** block converts the imported MIDI file into TuneScope format consisting a of a list of notes and durations (in seconds)



The current version of the **Convert MIDI** block can only process a melody consisting of individual notes. It cannot process chords and cannot process multiple tracks. Therefore, preprocessing to remove extraneous elements such as chords and multiple tracks must be completed before the **Convert MIDI** block is used.

7.4 Playing a Converted Melody

Once a melody in MIDI format has been converted to TuneScope format, the **Play Motif** block (introduced in a previous chapter) can be used to play the resulting list of notes.

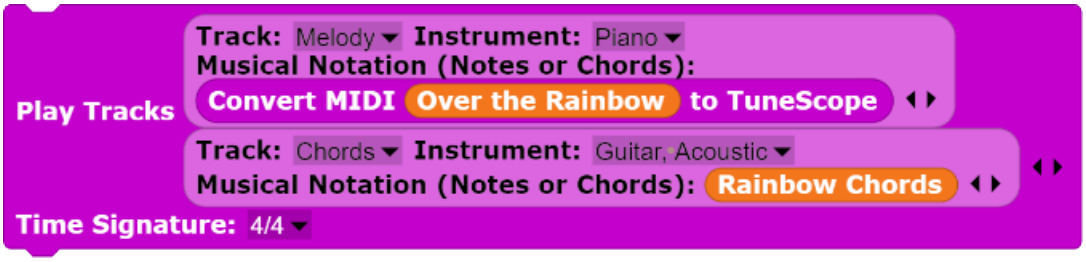


This block will play the converted MIDI track just as it would any other list of notes.

7.5 Playing TuneScope Tracks

The original TuneScope track blocks only process durations in the format of *quarter note*, *half note*, etc. An experimental version of the track block in the alpha TuneScope site can also process durations in seconds (the format used in MIDI files).

This makes it possible to import a melody in MIDI format, and combine the converted track in TuneScope format with a chord track or a drum loop.



Once this feature has completed its review and testing on the Snap*!* development site, it will be added to the TuneScope extensions in Snap*!*

7.6 MIDI Keyboards

The **Play MIDI Controller** block can be used to tell TuneScope that a MIDI keboard is connected and available for use. A small number of MIDI keyboard models used to test this feature are available in the dropdown menu. Other MIDI controllers can be specified by typing the name into the input slot. (If the exact name of the MIDI controller connected to the computer is not typed correctly, the MIDI connection to TuneScope will not work.)

Once the musical instrument (*Piano* in the example) and the name of the MIDI controller (MPK mini 3 in this example) are specified, the **Play MIDI Controller** block is clicked once. The MIDI initialization will then be active for the remainder of the session.

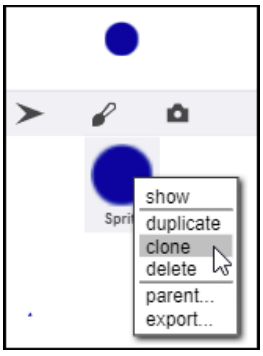
When a key is pressed on the MIDI keyboard, the key pressed is stored in the *Current Note* variable.

In addition to displaying the current note when a key on the MIDI keyboard is pressed, the corresponding MIDI number is displayed in a variable named *Current Note MIDI*. Each note has a corresponding MIDI number. C4 corresponds to the MIDI number 60. C#4 corresponds to the MIDI number 61, D4 corresponds to the MIDI number 62, and so on. MIDI numbers can be useful for transposing a series of notes to a different scale.

The ability to detect when a MIDI key is pressed and take an action in TuneScope based on the keypress can be used to create interactive music programs.

7.7 Interactive Music Visualization with MIDI Instruments

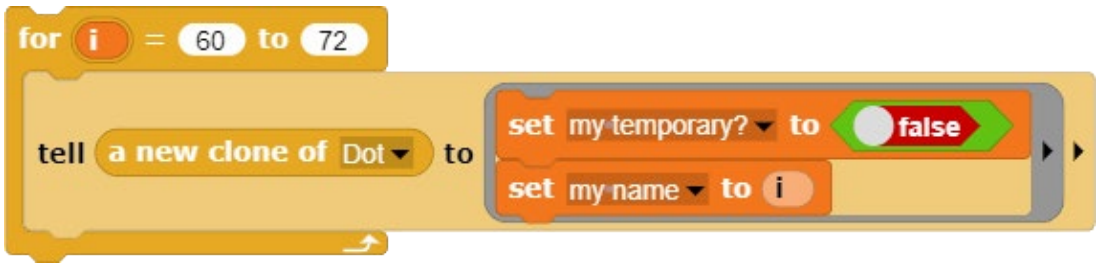
TuneScope music blocks lend themselves to creation of music visualizations. Many different ways of visualizing music have been developed. Light shows often use color to represent different elements of the musical scale. Visualizations could be created that highlight different major or minor scales. In Snap*!* / TuneScope, many types of visualizations can be constructed using a separate sprite to represent each note. In Snap*!*, clones provide a way to create multiple instances of a sprite. A clone of a sprite can be created by right-clicking the sprite icon in the sprite corral below the stage.



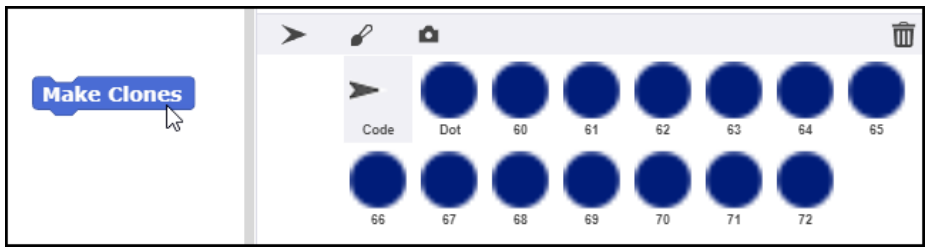
Two types of clones can be created with a script. Temporary clones are ones that disappear after they are no longer needed. For example, a shower of shooting stars might streak across the sky and

disappear after they burn up. A permanent clone is one that persists, like a regular sprite, but shares attributes of its template with the parent sprite.

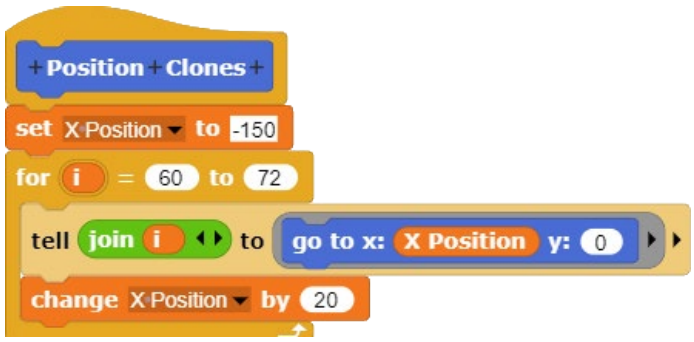
The following script creates a permanent clone for each item from 60 to 72. (These are the MIDI numbers that correspond to the octave beginning with Middle C (i.e., C4). The default state of a new program-generated clone is "temporary"; this script sets the *temporary* status to false, making the clone permanent. It then assigns each clone with a name corresponding to the current indexed number ("i").



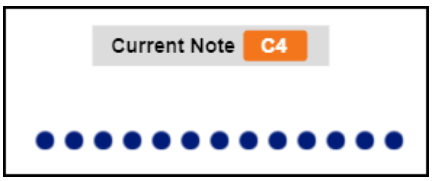
A code block **Make Clones** incorporates this script. When the **Make Clones** block is run, the numbered clones of the sprite named "Dot" appear in the sprite corral.



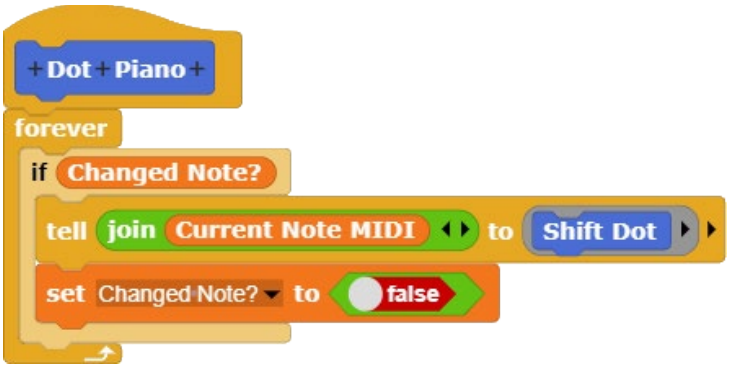
Once the clones have been created, the following procedure can be used to position the clones in a row across the screen. The variable "X Position" is set to an initial point at the left side of the stage. Each successive clone is then moved to a position to the right of the clone before. (The **Join** block is a string operator that is used to ensure that numbers such as "60" are treated as the name of a sprite rather than as a number that may be used in an arithmetic operation.)



After the **Position Clones** block is run, the clones will appear in a row across the screen.

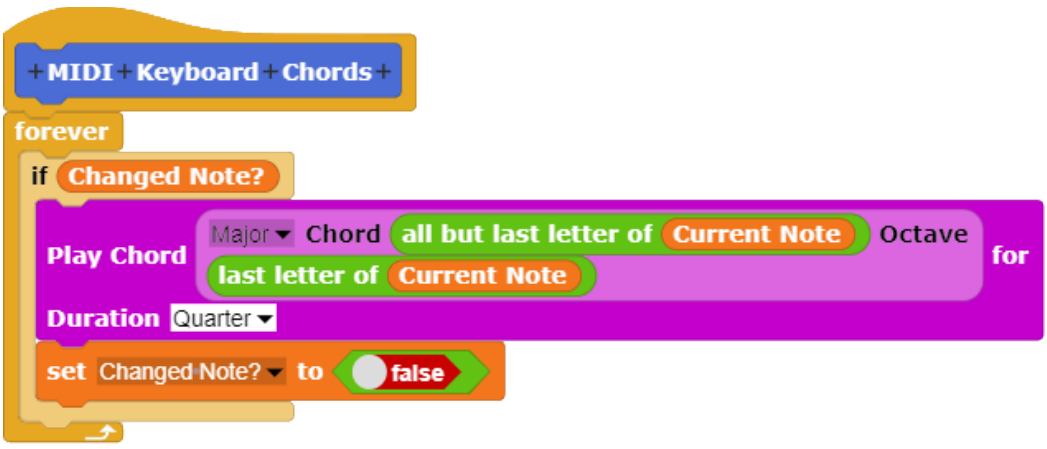


After this setup, a Dot Piano block can be created to shift each dot upward when the corresponding note on the external MIDI keyboard is played. The *Changed Note* variable can be used to determine when a new note has been played.



The **Shift Dot** block moves the dot upward for a moment and then back downward again.

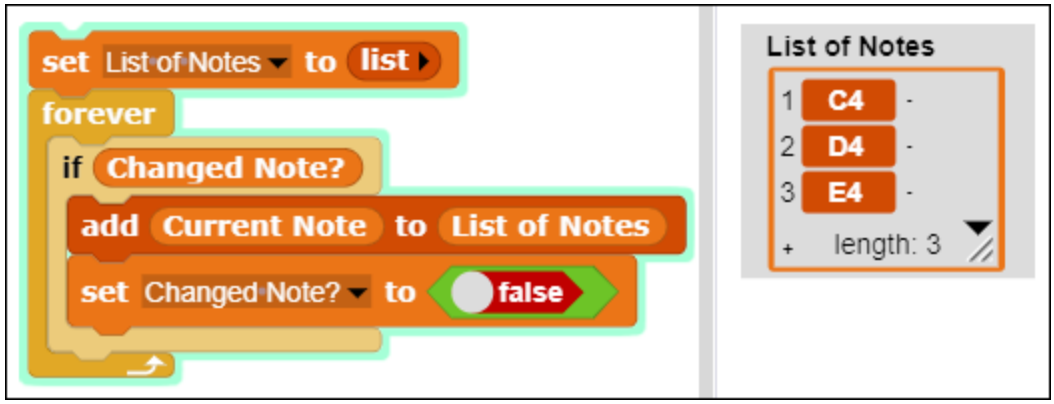
Many other musical variations are possible. For example, the following procedure plays the chord corresponding to each note played.



If a note such as “C#4” is played, the string operator **All But Last Letter** separates the note “C#” from the octave “4”. Similarly, the **Last Letter** block extracts the octave.

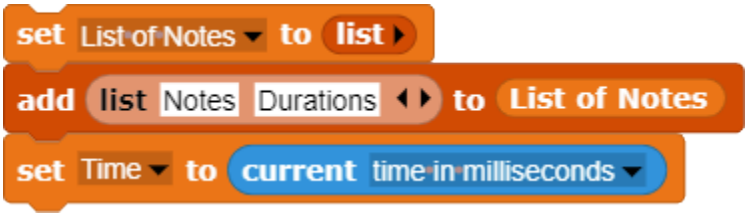
7.8 Recording Notes with a MIDI Keyboard

A MIDI keyboard can also be used as an input device to play and record a list of notes. If the current note changes, the new note is added to the variable *List of Notes*. The *Changed Note* variable is reset to *False* after the note is recorded. The loop then waits until the next note is played.

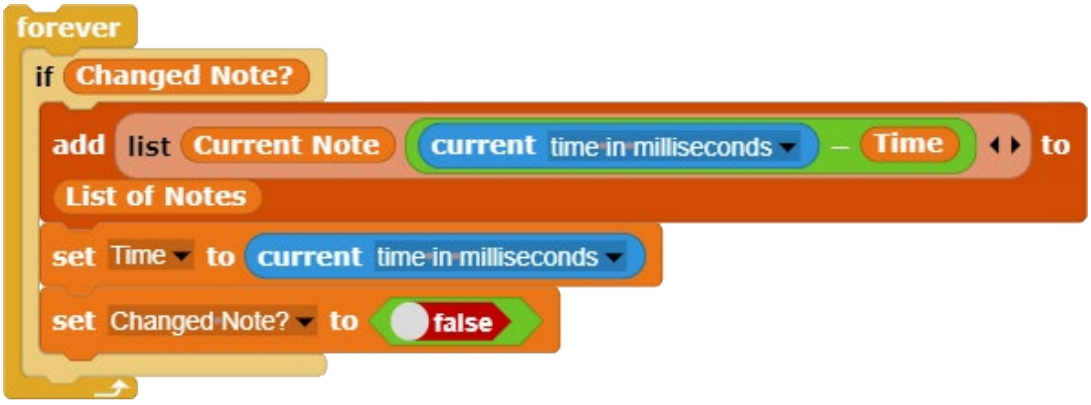


The **Current Time in Milliseconds** reporter block can also be used to record the duration between notes in milliseconds.

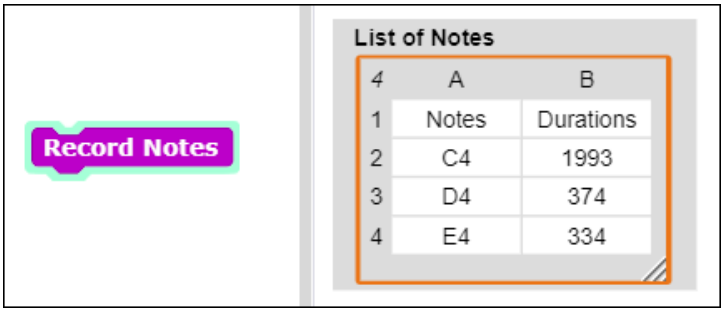
The **Record Note** procedure is initialized by creating a table with the heading “Notes” for the first column of the table and the heading “Durations” for the second column of the table. The current time in milliseconds is also recorded.



In the main loop of the procedure, the time between the keypresses is recorded by subtracting the current time that a new key is pressed from the previously recorded time. The current note and duration are then added to the table.



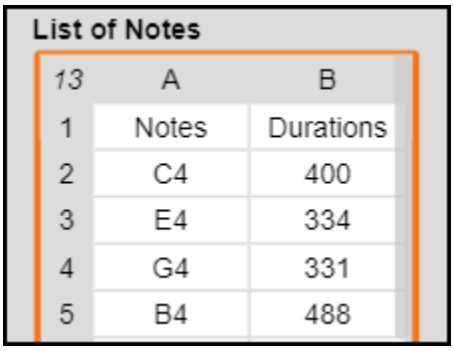
Any key on the MIDI keyboard is pressed to begin. The duration in milliseconds between keypresses is then recorded for each subsequent note.



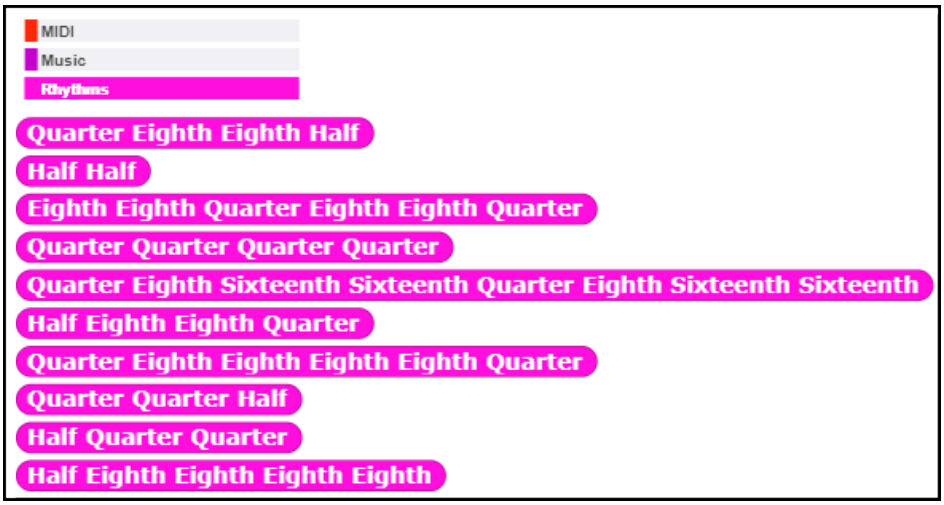
Since the previously recorded time is indeterminate for the first note, the duration for the first note is not meaningful and can be discarded.

7.9 Musical Patterns

Once a sequence of notes has been created, either by capturing notes played on an external MIDI keyboard or by manually typing notes into a list, a corresponding duration must be provided for each note. The **Record Notes** block records the time elapsed between each note played on the MIDI keyboard. That provides an approximation of note durations that can be converted into musical terms such as quarter note, half notes, etc.

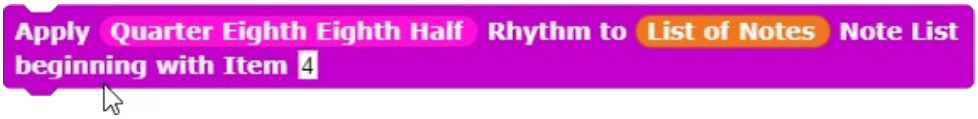


This can be done by manually editing the durations or by applying a pattern. A number of musical patterns have been provided in a program accompanying the MIDI beta site in a palette titled *Rhythms*:

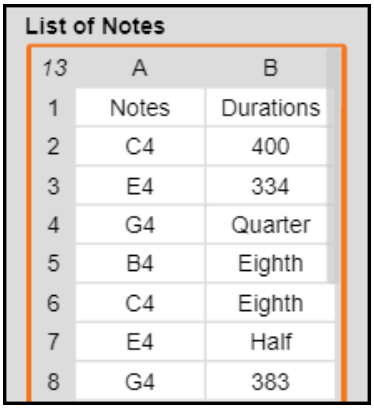


For example, the **Quarter Eighth Eighth Half** reporter block reports that sequence of durations:

The **Apply Rhythm to Note List** block can be used to apply the pattern of durations to a list of notes.

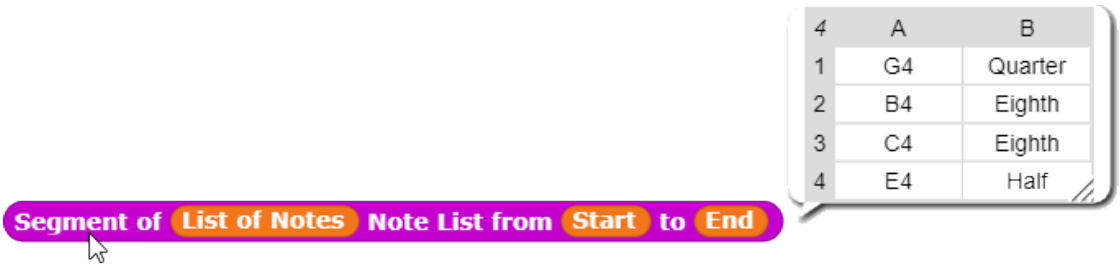


The selected pattern of durations is then applied to the list of notes beginning with the selected item in the list (Item 4, in this example).

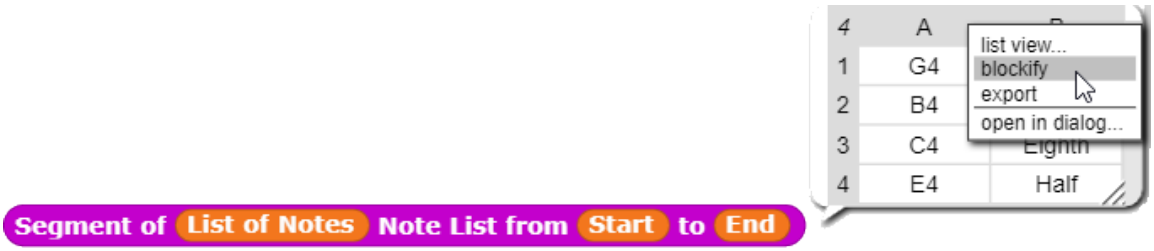


The starting and end points of the segment are recorded as global variables *Start* and *End*. These variables can be used with the **Play Segment** block to play the pattern.

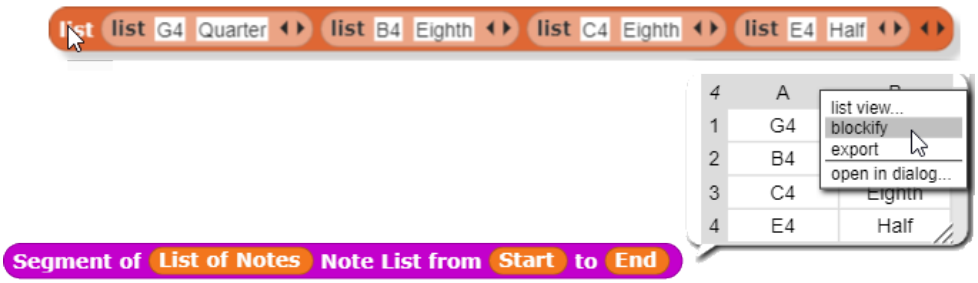
The **Segment of Note List** reporter block can be used to display the selected segment.



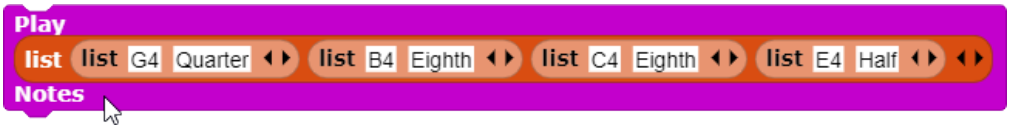
A right-click on the displayed table of notes presents an option to “Blockify” the list.



The *blockify* option creates a list of the notes and durations:



The list of the notes can then be played in the same manner as any other list of notes:



The **Add Segment to Melody** block adds the selected segment to a list named *Melody*.

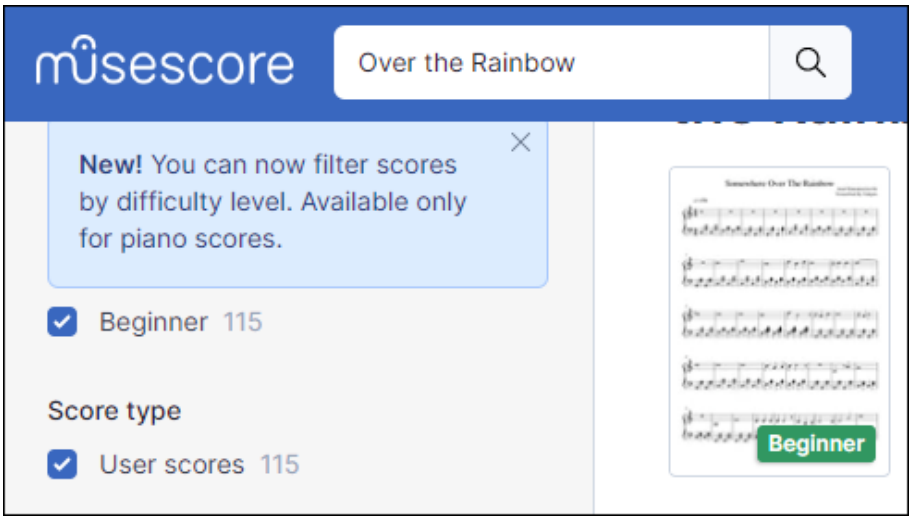
Through a process of exploration, sequences of notes can be generated and combined with patterns of durations. The segments of notes created in this way can then be combined to form a melody.

MIDI Appendix

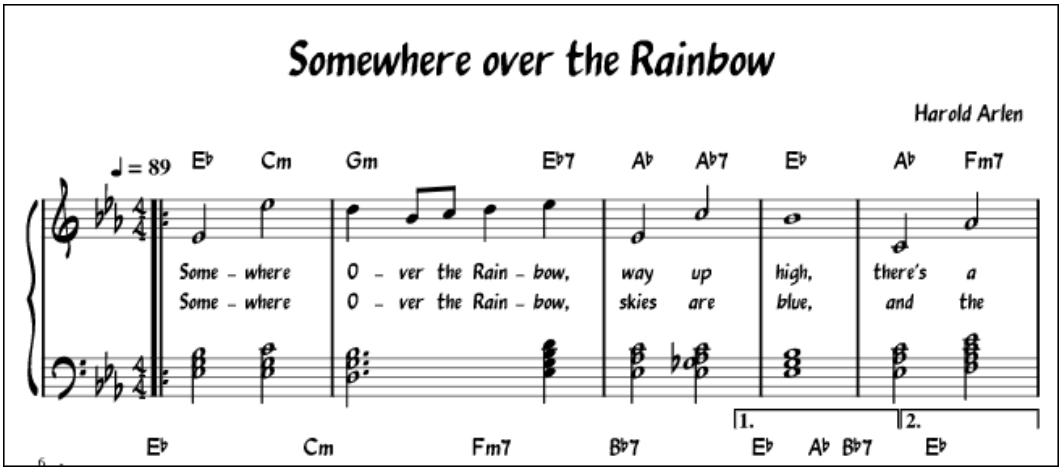
A. Editing MIDI Files with Music Notation Programs

Music notation programs such as *NoteFlight* and *MuseScore* can be used to import and edit MIDI files. These programs can also include search engines that can be used to search for songs.

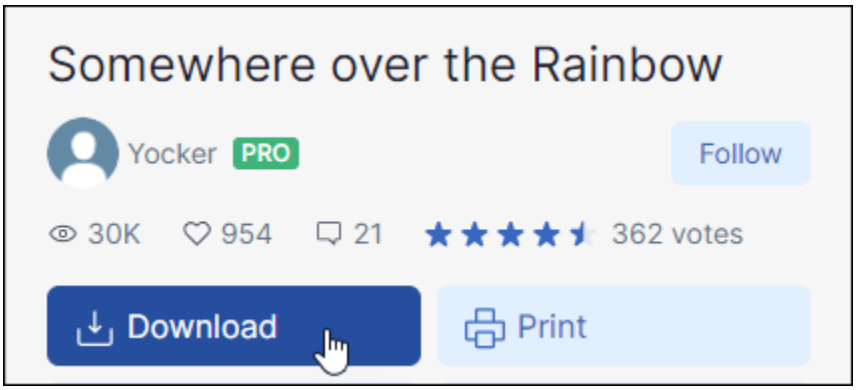
Music scores can be filtered by categories such as “Beginner” and “User Scores”. Filtering search results for scores that have been created by individual users (rather than by a corporation) is more likely to yield music scores that can be edited without a fee.



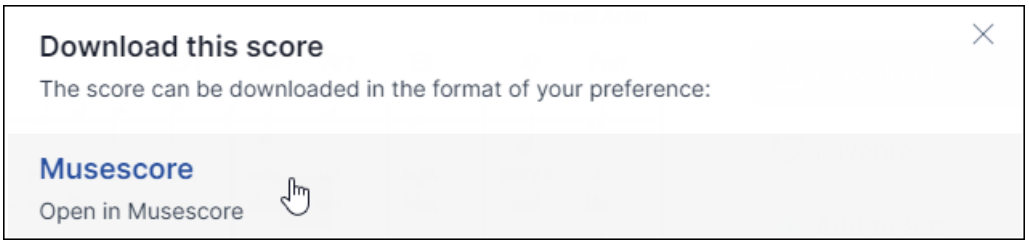
Once a music score for a song is identified, the score looks like this:



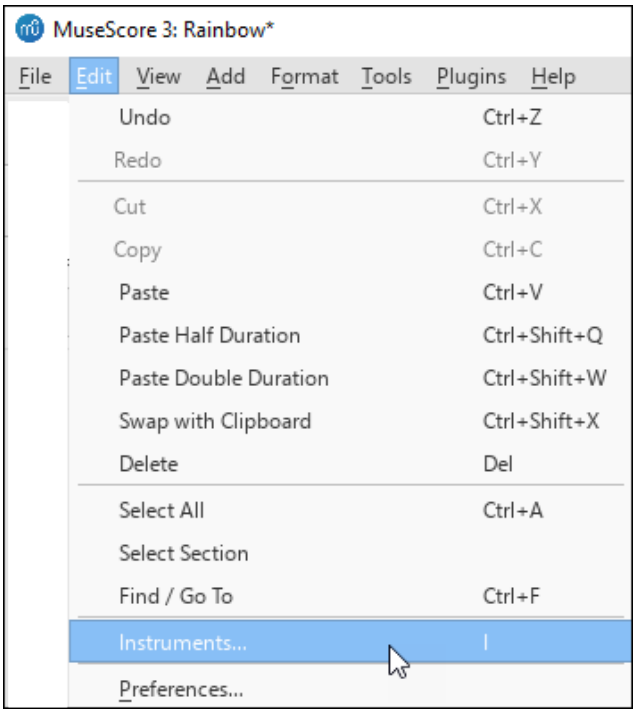
The score can then be downloaded:



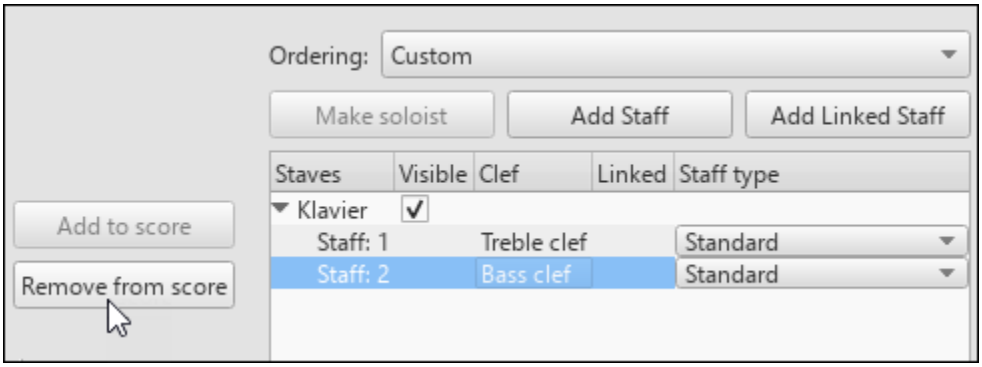
Options for several formats, including Musescore, are available:



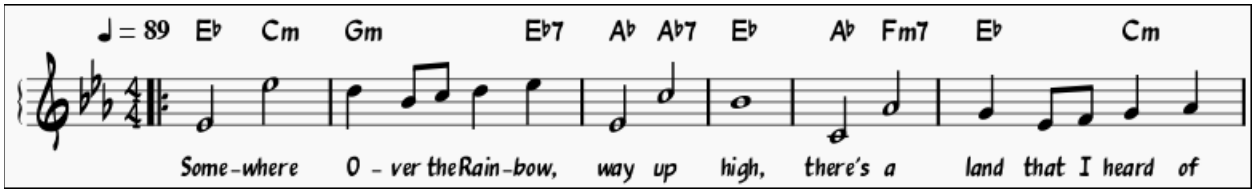
Once the file is exported and saved in native Musescore format, it can be re-opened in Musescore and edited. The native Musescore format includes the option of editing the musical instruments included in the score:



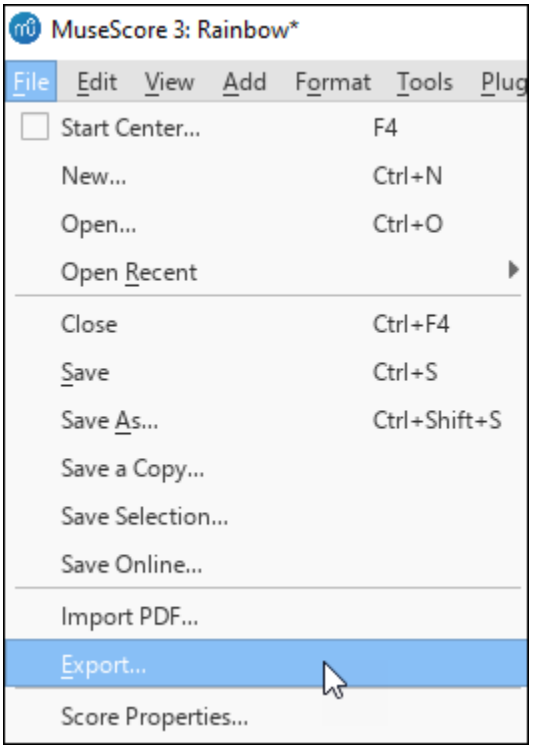
In the example below, the individual notes of the melody are in the treble clef (the top staff of the score) and the chords are in the bass clef (in the bottom staff). To extract the melody, the bass cleff can be removed from the score.



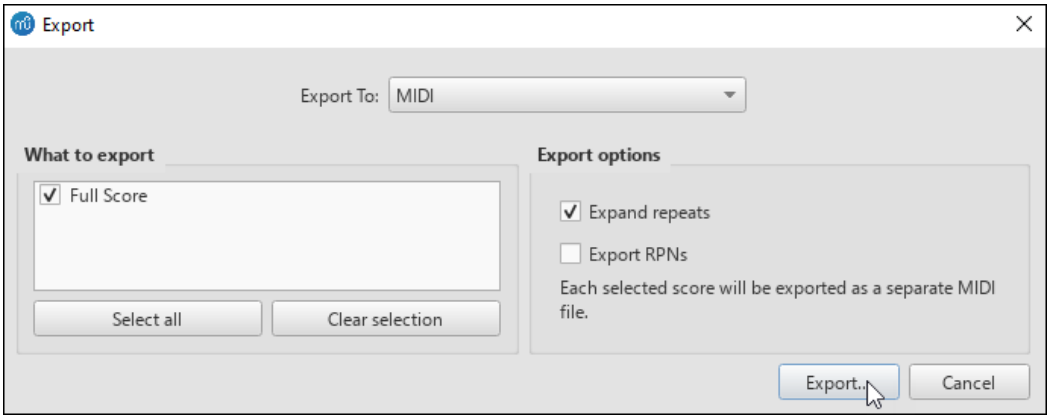
The resulting score now consists of just the melody.



The melody obtained in this way can now be exported for use in TuneScope.



The score should be exported in MIDI format.



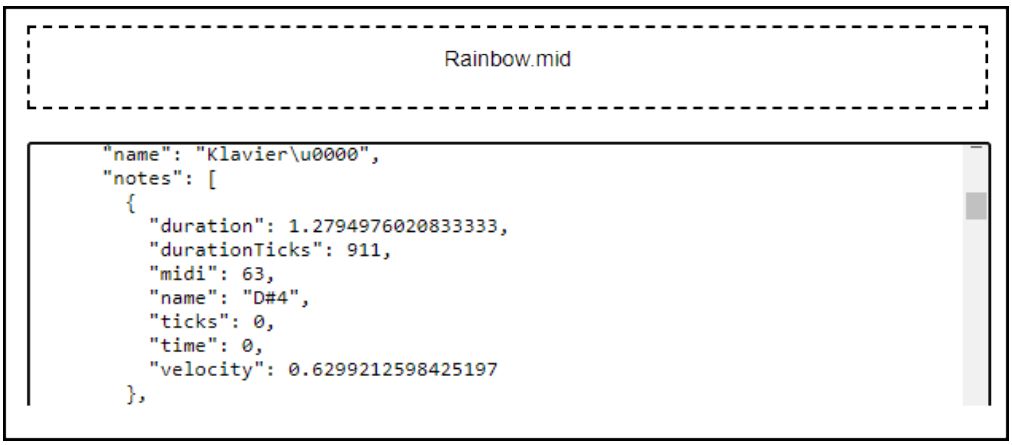
B. Converting MIDI Files to JSON Format

There are a number of programs that can convert a MIDI file into a human-readable JSON format. This example uses the Tone.js MIDI conversion program:

https://tonejs.github.io/Midi/

The Tone.js MIDI conversion interface provides an area into which a MIDI file can be dropped.

The converted file appears in a text window below.



The file in this window can then be copied to the clipboard and pasted into a text editor such as NotePad or NotePad++. The file can then be saved with a ".json" extension. The ".json" file obtained in this way can then be imported into Snap*!* by dragging the file into the script area of Snap*!*