Electronic Music in Scratch

Robert Moog developed the first commercially available music synthesizer. The Moog synthesizer was used to produce the album, *Switched-On Bach*. This work became only the second classical music album to sell more than a million copies. Today's software synthesizers allow anyone to begin exploring electronic music with considerably less difficulty and expense.

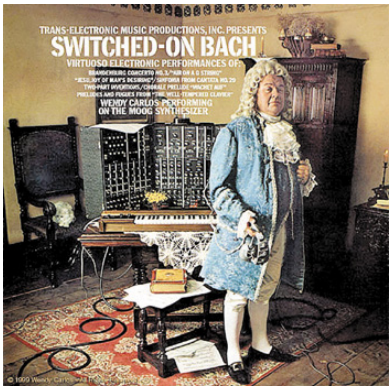

Figure 1. The album, Switched-On Bach, was created with a music synthesizer.

Using a computer to create music require some understanding of *coding*. Coding refers to the arrangement of computer commands to achieve a desired effect. Expertise in *computational thinking* is also helpful. Computational thinking refers to the thought processes required to solve a problem using a computer.

If you have not used Scratch before, first complete the introduction here.

This lesson introduces Scratch procedures by animating a guitar player and a drummer. Through it, you will learn about the Scratch music extension and will use these commands to recreate the AC / DC song *Thunderstruck*. You will then be able to animate a band playing this song using the animations that you developed in the *Computer Animation* chapter.

This understanding will give you the necessary foundation for creation of electronic music through other visual programming languages such *Pure Data* and *Max*.

Pure Data is an open-source programming environment used by professional musicians. It is available as an open source program (free to download).

Max is a commercial program with similar capabilities.

Both are used extensively by professional musicians. However, they can be intimidating if you are not an experienced programmer. Our introduction to electronic music through Scratch will provide this foundational knowledge in a user-friendly environment.

Figure 2. Creating a Scratch Program

Log into Scratch and begin creating by clicking the **Start Creating** button. A new project page will appear. In Chapter 3 you learned about animating movement of sprites using Scratch motion commands. This Chapter will focus on Scratch music commands.

Scratch Music Extension

Scratch commands are shown on the left-hand side of the screen. A program editing window where you can create Scratch programs is located in the middle of the screen.

In addition to built-in Scratch commands, you can access many other commands through Scratch extensions. These extensions include libraries of commands to control motors, play music, translate languages, and program household devices through the Internet of Things.

Click the Scratch extensions icon in the lower left-hand corner of the screen to access these additional extensions. This icon is *highlighted* by the black arrow in the left-hand corner of Figure 3.

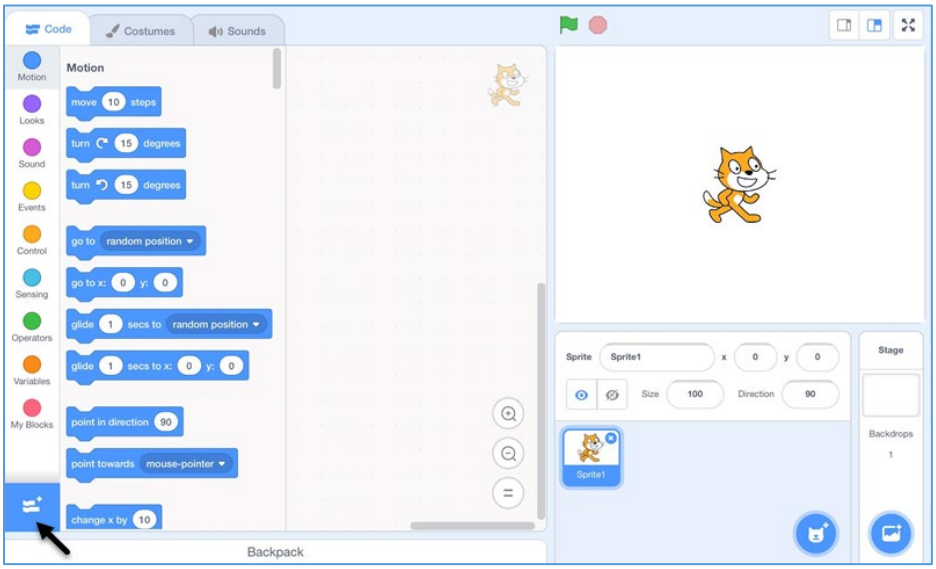

Figure 3. Accessing Scratch Extensions

Once the extension page is open, click the Scratch *Music Extension* icon to add a library of music commands to Scratch.

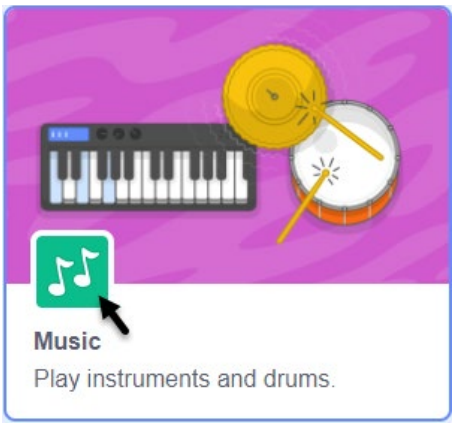

Figure 4. Selecting the Scratch Music Extension

After the Scratch *Music Extension* library is selected, a set of additional music commands such as *Play Note* will appear on the left-hand side of the Scratch screen.

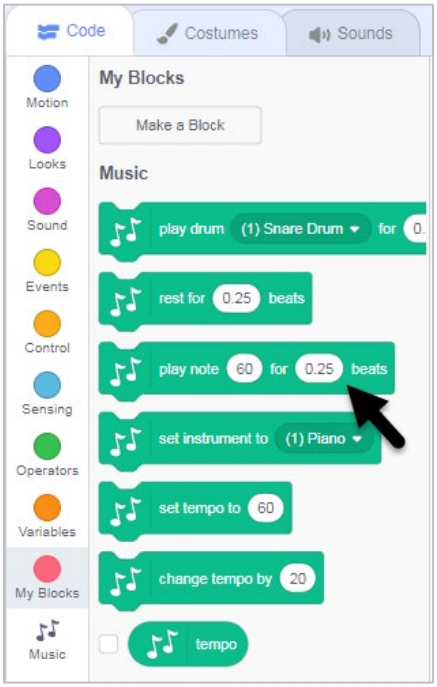

Figure 5. Selecting the "Play Note" Command

Scratch Music Commands

The activities that follow introduce both musical notation and computational thinking. You will learn to use Scratch to construct a song even if you have no prior musical background. In the process you will learn more about the underlying structure of songs.

The Scratch **Play Note** command can be used to play a musical note. Drag a copy of the Play Note command from the Code Block area on the left-hand side of the screen into the program creation / editing area in the middle of the screen. Click the icon of the musical notes on the left-hand side of the **Play Note** code block that you moved into the editing area to hear a piano note.

If you place the cursor on the number 60 (to the right of the words *play note*) and click, a piano keyboard will appear. The cursor must be in the exact center of the number 60 for this to occur, so some experimentation may be required. Success will be rewarded with the appearance of a piano keyboard below the **Play Note** code block.

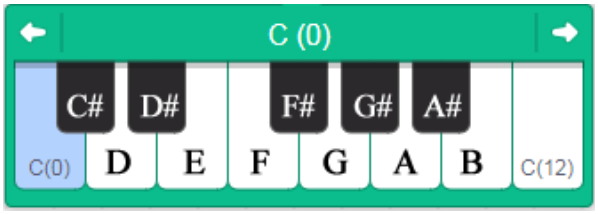

Figure 6. Notes for One Octave of a Piano Keyboard

Each note has a corresponding number that is used by the Scratch **Play Note** command (see Figure 7).

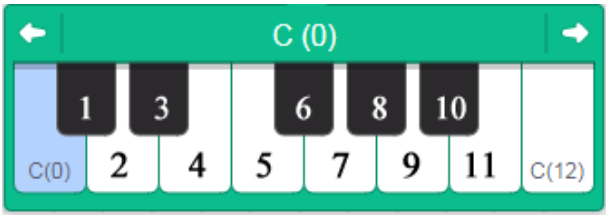

Figure 7. Numbers Used by Scratch that Correspond to Each Note

There are twelve notes in each octave – eight white keys and five black keys. The virtual keyboard spans 11 octaves. The corresponding numbers associated with each note therefore range from 0 (beginning with the note "C" in the first octave) to 132.

Selecting a key on the virtual keyboard places the corresponding number in the **Play Note** command. For example, selecting the note *B* in the fifth octave inserts the corresponding number 71 into the **Play Note** command.

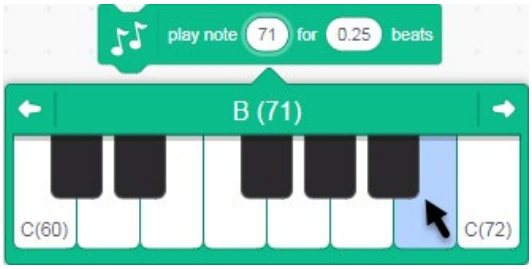

Figure 8. The Scratch "Play Note" Note Selection Interface

A series of musical notes can be combined to play a musical sequence. The song *Thunderstruck* by the band, AC/DC, begins with a lead-in note (B) followed by this sequence of notes: "D♯, B, F♯, B". This sequence corresponds to the following keys on the Scratch virtual keyboard: "75, 71, 78, 71." The following Scratch commands will play this initial sequence of notes.

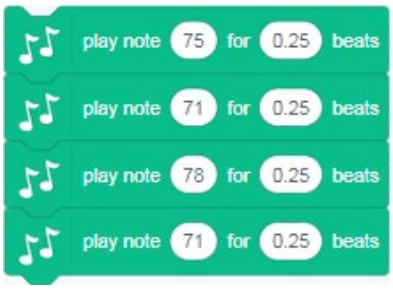

Figure 9. A Four-note Sequence in Scratch

Music Notation

Guitarists often learn to play songs using a form of musical notation known as a *tablature* or 'tab." You will learn to translate the notation on a guitar tab into a format that can be used in Scratch in the following activity. Once you learn how to do this, you will be able to translate any song in a similar way.

The song *Thunderstruck* by the band, AC/DC, provides a good introduction to the translation process because: (1) the notes are all played on one string of the guitar and (2) the notes are all of equal length. This simplifies the translation process, and provides a good foundation for later translating songs with notes of different lengths played on more than one string.

The guitar tab for the opening sequence of Thunderstruck is shown in Figure 10. The tab excerpt indicates that there are 4 beats in each measure, with four measures for a total of 16 notes.

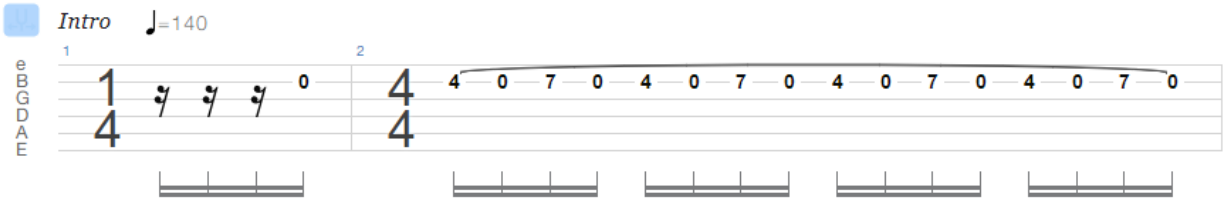

Figure 10. Guitar Tab for the Opening Sequence of Notes in Thunderstruck

The six lines shown in Figure 10 represent the *six strings of the guitar*. The numbers represent the *frets* where the guitar string is pressed down when a note is played. This example represents notes played on the second string of the guitar.

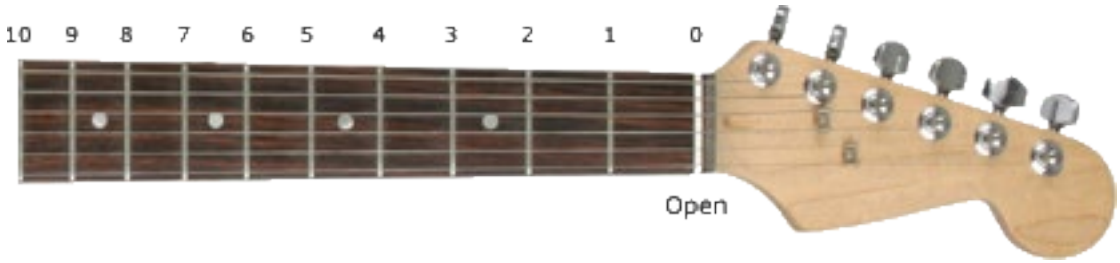

Figure 11. The First Ten Frets on the Neck of a Guitar

The number 4 (on the second line from the top of Figure 10) near the beginning of the tab represents the fourth fret on the second string. The guitar player presses the second string down on the fourth fret as the string is plucked to play the note. Similarly, the number 7 indicates that the string is pressed down on the

seventh fret as the note is played. The number 0 is a special case. It means that the note is played without pressing down on any fret. This called an *open* (or *unfretted*) note.

Plucking the second string without pressing down on a fret yields the note *B* in standard tuning. Because playing this string open in standard tuning gives a B note, the second string of a guitar is known as the B string. The note played on the B string with the string pressed down at the fourth fret produces an E♭ (E flat) note. This note is also known as D♯ (D sharp). The note written as 7 in the tab produces an F♯ note when played. All of the notes in this excerpt from Thunderstruck are played on the B string of a guitar, with every other note being a B note, played on the open B string.

The numbering systems used to represent notes on the Scratch virtual piano and on the guitar tab employ different notations. The Scratch virtual piano begins numbering with the first octave, beginning with 0, and continues numbering cumulatively across all of the keys on the keyboard. The number 0 in the guitar tab above is equivalent to the number 71 in the Scratch virtual keyboard.

Table 1. Musical Notes Represented in the Guitar Tab Above

Notes	*B*	*C*	*C♯*	*D*	*D♯*	*E*	*F*	*F♯*	*G*	*G♯*	*A*
Guitar Tab	0	1	2	3	4	5	6	7	8	9	10
Scratch	71	72	73	74	75	76	77	78	79	80	81

This information in this table shows the connections between: (1) the musical notes (B, C, D, etc.), (3) the numbers on the guitar tab, and (3) the numbers used by Scratch. Understanding these relationships will enable you to translate other songs from one notation format to another.

Scratch Procedures

The concept of procedures is one of the most powerful ideas in computer science. A procedure is a sequence of commands that have been assigned a name. (If you are not familiar with Scratch procedures, complete the introduction to Scratch in Chapter 3 before continuing.)

Once a procedure has been defined, the entire sequence of commands can be invoked (called upon) by using the name of the newly-defined procedure. Procedures allows sequences of commands to be reused without reentering the code sequence. The process of defining code sequences as procedures is an example of computational thinking concept known as *abstraction*.

To define a new Scratch procedure, click the pink icon (dot) labeled **My Blocks** on the left-hand side of the screen (just above the Music icon). An option labeled **Make a Block** will appear. In Scratch the terms *block* and *procedure* are used interchangeably.

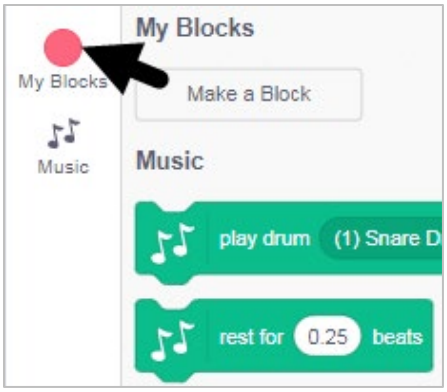

Figure 12. Making a New Scratch Block

Click the **Make a Block** button to make a new block. A new pink code block will appear.

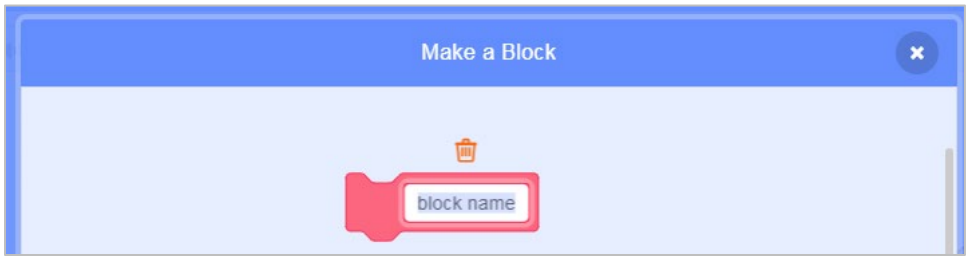

Figure 13. Naming the New Scratch Block

Type the name of the new procedure into the space labeled **block name**. In our example, because the new procedure consists of the opening sequence of *Thunderstruck* notes, the name **Thunderstruck Opening Notes** was entered into this space.

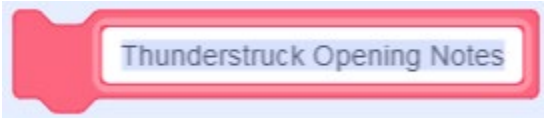

Figure 14. Naming the New Scratch Block: Thundestruck Opening Notes

Once you are satisfied with the name of your new procedure, click **OK** (in the lower right-hand corner of the code editing screen) to confirm your choice.

A new **Definition** block will appear in the program editing area.

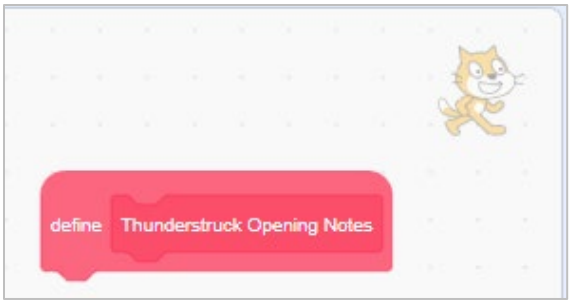

Figure 15. The Scratch Code Definition Block

Drag the commands that will be used in the new procedure to the space below the **Definition** block. The block of commands will snap into place.

Figure 16. Defining the Sequence of Notes in the New Code Block

Once the procedure is defined, a code block with the name of the new procedure will appear in the **My Blocks** area. In essence, you have defined a new command that has been added to the Scratch language.

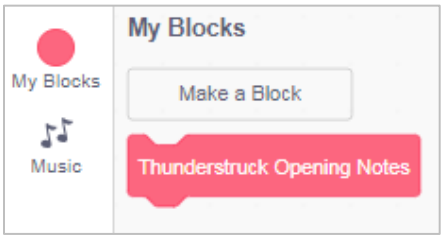

Figure 17. A New Scratch Command Appears in the "My Blocks" Area

Drag a copy of the new command into the program editing area of the Scratch screen and click it. The initial sequence of *Thunderstruck* notes will play each time this code block is clicked.

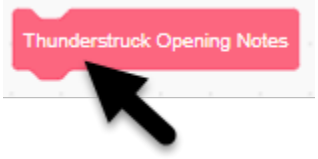

Figure 18. Playing the New Scratch Command

Inputs to Procedures

A Scratch procedure can be defined that adds a numeric offset to the number representing each note. This procedure adds 71 to each note from the guitar tab. The procedure renders 0 in the guitar tab as 71 in Scratch, 1 in the guitar tab as 72 in Scratch, 2 in the guitar tab as 73, 3 in the guitar tab as 74, and so on.

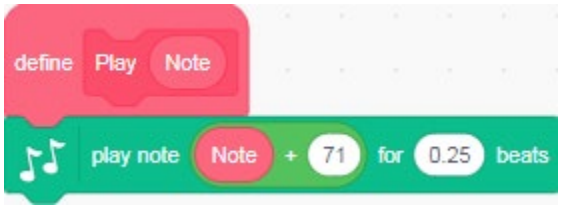

Figure 19. Defining the "Play" Block

This procedure simplifies translation of the guitar tab into Scratch musical notes, since the original guitar tab notation can now be used in the Scratch procedure.

Open String
4th Fret
Open String
7th Fret

Figure 21. Using the Newly Defined "Play" Command

Thunderstruck is notable for the use of the note B in every other note. In music, a note that is sounded throughout a piece is known as a *drone note*. A Scratch procedure to play the drone note *B* (shown as the number "0" in the guitar tab) after every other note could be defined in the following way.

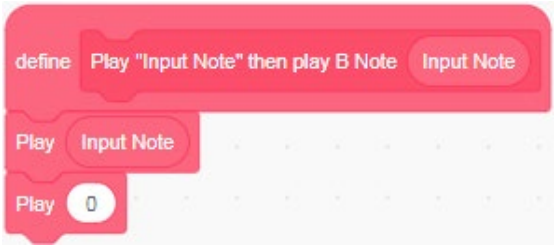

Figure 22. Defining a Command to Play Two Notes

This strategy permits the opening sequence to be further simplified. The first note is provided as an input to the procedure. The procedure then plays that note followed by the note *B* (defined as "0"). Consequently, the two lines of code below would play the following sequence of notes: 4, 0, 7, 0.

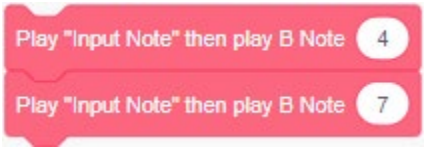

Figure 23. Playing a Note Followed by a Drone Note

This opening sequence of notes is repeated eight times. Scratch provides a *control structure* to repeat a sequence of commands a specified number of times in our example below, the two commands inside the Repeat block are repeated eight times before ending.

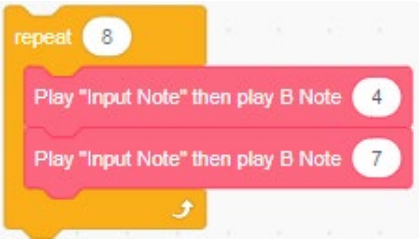

Figure 24. The Scratch Repeat Command

Thunderstruck consists of variations on four note sequences consisting of a note followed by a drone note and a second note followed by a drone note, such as (in guitar tab notation): 4 0 7 0. The "Play Stanza" procedure shown below repeats this type of four-note sequence a specified number of times.

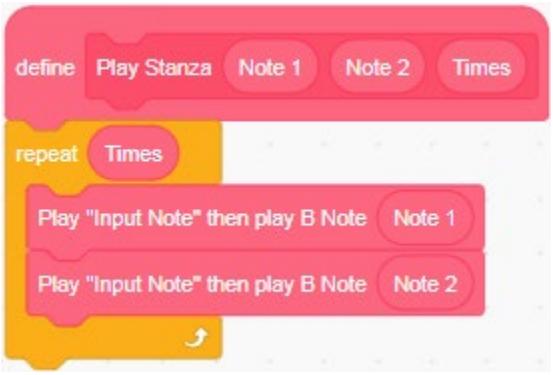

Figure 25. Using a Variable to Control the Number of Times the Notes Are Repeated

The following **Play Stanza** command repeats the four-note sequence "4 0 7 0" eight times. This allows a single command to play a sequence of 24 notes.

Figure 26. Using the Newly Defined "Play Stanza" Command

A musical *riff* is a short, repeated sequence of notes associated with a song. The initial riff at the beginning of Thunderstruck consists of the stanza "4 0 7 0" repeated 8 times followed by the stanza "5 0 8 0" repeated 8 times. You could describe it in the following way using the newly defined **Play Stanza** command.

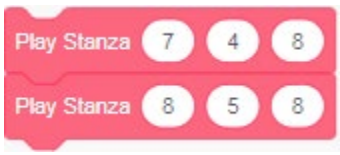

Figure 27. Notes in the Opening Riff

This entire sequence is repeated twice. The opening riff, therefore, can be defined in the following way.

Figure 28. Defining the First Thunderstruck Riff

Starting and Stopping a Program

The opening sequence of notes begins with a drone note (expressed as "0," for "Open String" in the guitar tab notation) followed by the initial riff. The Scratch **Green Flag** block allows a program to be run by clicking the green flag. The Scratch **Tempo** command controls the rate at which the notes are played.

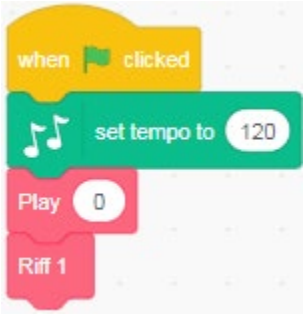

Figure 29. Using the Start (Green Flag) Command

Addition of a variable, *Drone Note*, makes the procedure clearer and more flexible. It allows the drone note *B* to be replaced with a different drone note by changing the value of the variable at the beginning of the procedure.

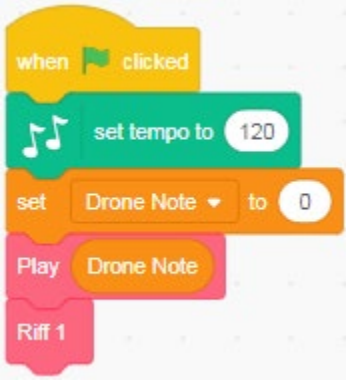

Figure 30. Defining the Drone Variable

Addition of a variable for the guitar tab offset also provides further flexibility. If the key in which the notes are played is changed, the offset also changes. The **Guitar Tab Offset** variable can be used to adjust this offset.

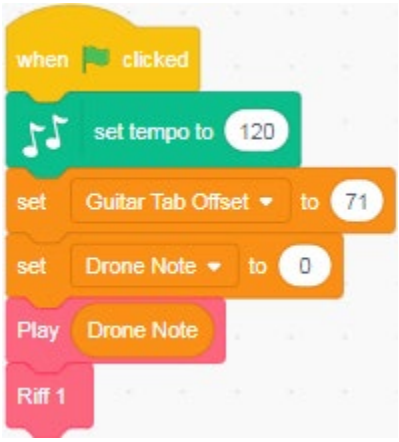

Figure 31. Defining the "Guitar Tab Offset" Variable

Once the **Guitar Tab Offset** variable has been assigned a value in the main procedure, the **Play** procedure can be updated to make use of this variable.

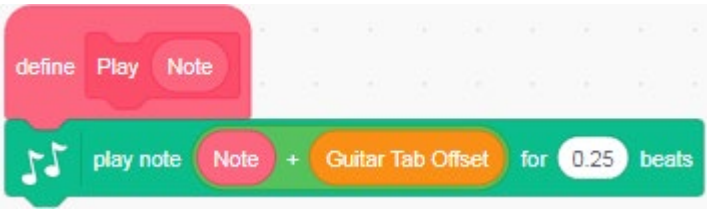

Figure 32. Using "Guitar Tab Offset" Variable in the Play Procedure

The opening riff is followed by a second riff that is defined in the following way.

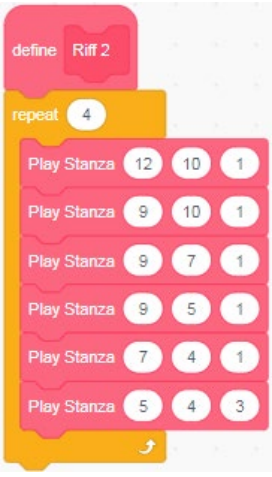

Figure 33. Defining the Second Thunderstruck Riff

Starting a Parallel Scratch Procedure

After the first two riffs are played, the Riff 2 is repeated a second time with a backing drum track. The drum track procedure in *Thunderstruck* can be defined in the following way.

Figure 34. The Thunderstruck Drum Track Procedure

The Scratch **Broadcast** command is used to send a message that initiates a parallel drum track sequence. From a metaphorical perspective, it is as though the main guitar procedure has a radio transmitter, and the drum procedure has a radio receiver. When the main procedure transmits the message "Play Drums," this message is received by the drum procedure.

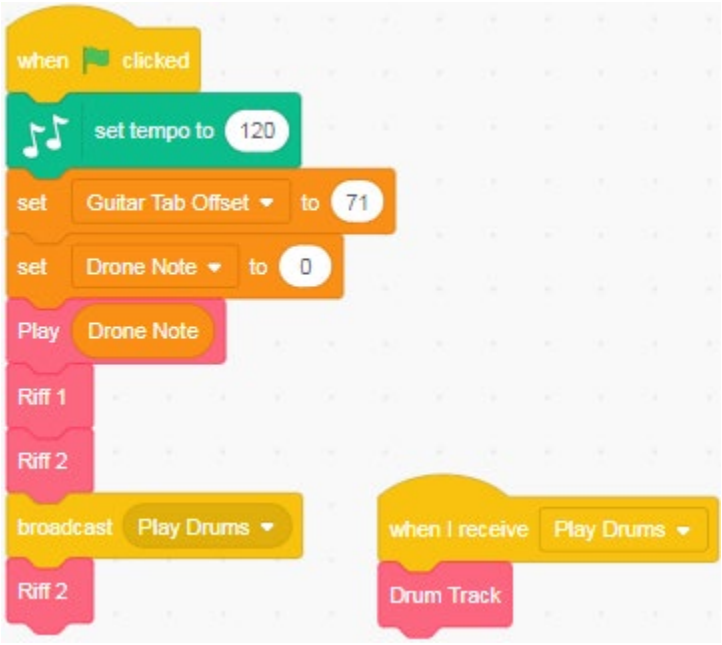

Figure 35. Transmitting a Message from the Main Procedure to the Drum Track Procedure

The ability for one procedure to transmit a message that is received by another procedure in this way makes it possible for multiple events to occur at the same time. While the main procedure is playing the guitar notes, another can play the drum track.

Programming Concepts

These digital music above illustrate a number of key programming concepts that you will need in later chapters. The same computational thinking concepts will be employed in the subsequent chapters related to the *Motion* theme. You will use these concepts to control motors, actuators, and robots, among other activities.

1. Procedures

Procedures are at the core of programming. A procedure is a recipe for a sequence of steps. Once a procedure has been defined and named, it can be recalled repeatedly. The *Play* and *Play Stanza* procedures that you created become extensions of the Scratch programming language that work in much the same way as the built-in Scratch commands. These added commands greatly simplify the digital music program.

2. Variables

Variables allow a general-purpose procedure to be written with specific values substituted in place of the variable name at the time that the program is run. For example, the *Play Stanza* procedure has three inputs:

Note 1: an initial note to be played
Note 2: a second note to be played
Times: the number of times the notes are repeated

Use of variables in the procedure makes it possible to use the procedure to play many different combinations of notes.

3. Control Structures

A control structure can be used to determine how many times a procedure will be repeated. For example, the *Repeat* command in Scratch is a control structure. An "If … Then" command is another example of a control structure: "*If* this occurs, *then* do this."

Computational Thinking

In the examples in this chapter, electronic music was used as an introduction to computational thinking. The repeated patterns in music lend themselves to computational thinking. The same concepts are equally applicable in other *Make to Learn* projects such as design and construction of a hydraulic press or use of industrial controls widely employed in manufacturing.

Computational thinking refers to formulation of a problem in a manner that allows a computational device such as a computer to be used to solve the problem. Three key computational thinking concepts include (1) development of algorithms, (2) abstraction, and (3) analysis.

1. Algorithms

Development of an algorithm involves identification of a pattern. In the case of the digital music procedures, the patterns involved recognition of repeated sequences of notes. These patterns made it possible to write a procedure that could play a sequence of 24 notes. If the pattern had not been

identified, it would have been necessary to write one command for each note, in 24 separate lines of code.

2. Abstraction

Once an algorithm has been identified, it can be implemented as a function – a named routine or block of code using parameters to specify different instances of the activity. This provides a means of burying complexity, allowing the programmer to work at a higher level of abstraction. In the absence of methods of abstraction, the programmer's working memory would soon be overwhelmed.

3. Analysis

Execution of a solution is followed by assessment and analysis of the result. Correctness of a solution is important. In this case, a correct solution means that the opening notes of Thunderstruck are played in the right order and at the correct tempo. Assessment of a solution not only includes an analysis of whether the solution is correct, but also whether the program is easy to understand. If a program is easy to understand, it will be possible to modify and extend the program at a later date.

Remixing a Scratch Program

The complete Scratch program described above is available at:

https://scratch.mit.edu/projects/278994492/

The *Remix* button in the top right-hand corner of a Scratch page allows programs to be remixed. All programs posted on the Scratch web site are available to be shared and remixed.

Figure 36. Remixing a Scratch program.

By remixing this initial set of procedures available through the link above, you can experiment with development of your own set of musical procedures. We invite you to pick a song of your own. You can reuse many of the procedures developed in this introduction, but will need to modify some of the parameters or values.