Music Blocks: A Musical Microworld

Walter Bender, walter@sugarlabs.org
Sugar Labs, a member project of the Software Freedom Conservancy

Devin Ulibarri, devin@devinulibarri.com
New England Conservatory Preparatory School and YMCA Malden

Yash Khandelwal, khandelwalyash51294@gmail.com
International Institute of Information Technology, Hyderabad

Abstract
Music Blocks software is designed for teachers and learners to explore the fundamental concepts of music in a visual-coding environment. Music Blocks is both innovative and beneficial to music education in a number of ways: On the one hand, it is a new method for understanding the fundamental concepts of music; on the other, it is a tool for learning coding and logic skills. It integrates both music and STEM fundamentals in a fun, scalable, and authentic way. Lastly—and very importantly—the tool itself is Free/Libre Software, which we argue is the best choice for an equitable and just education because it gives students the freedom to study, without restriction, both how to use the software and how the software itself works, i.e., how it transforms their programmed instructions into their musical inventions.

Keywords
Logo, Constructionism, Programming, Music

1. Introduction

1.1 Music Box: Voices from the past
In 1970 Edward Fredkin and Marvin Minsky designed and built the first digital synthesizer (Fredkin and Minsky 1970), which was commercialized under the name Triadex Muse. Triadex Muse was not intended to just be performed, but also programmed. It came with both the Triadex Muse Manual (`http://trovar.com/muse/book.html`) and the Triadex Muse Programming Guide (`http://trovar.com/muse/book2.html`), which includes a section on music theory. It is reasonable to infer that Fredkin and Minsky intended that the Triadex Muse present an opportunity for creating music while learning music theory and programming.

In a related effort, Jeane Bamberger and Terry Winograd built a Logo programming language interface to Minsky's Music Box. The software helped learners make connections between music ideas by actively making music (Bamberger 1991) A 1972 video of children using Music Box is available at `https://www.youtube.com/watch?v=xMzojQFyMo0`. Papert and Cynthia Solomon describe the interaction between learner and computer as a “conversation” in words or numbers [and music]. “Thing to do” #11 is “Make a Music Box and program a tune.” Rather than simply “printing” notes to the synthesizer, learners program songs using procedures attuned to musical structures (Papert and Solomon 1971).

In the example shown at the top of Figure 1, *Frère Jacques* (a nursery rhyme of French origin) is programmed by transcribing each note individually. Pitch is in semitones and represented by a letter of the alphabet (i.e., A = tonic, C = second scale degree, E = third scale degree, etc.). Rhythmic duration is achieved by adding more of the same pitch together in a string. One flaw in this design is that it is not obvious that there are two distinct parameters for each note. A second flaw is that it limits one’s ability to specify unique rhythmic values, such as triplets, etc. Papert and Solomon describe this as a “very bad” design for the program. Papert and Solomon assert that a “better way [is to use] descriptions of music in a good notation”. In the bottom example (which uses the “SING command”), the music is structured around the ideas behind pitch, rhythm, and phrasing, as well as higher-level concepts such as repeat and loop. The goal of Music Box was not just synthesis, but to engage the learner in constructing knowledge about music through programming.

```
TO FREREJACQUES
1 PRINT “AAA!CCC!EEE!AAA!AAA!CCC!EEE!AAA!EEE!FFF!HHHHHH!EEE!FFF!HHHHHH!...
END
```

```
TO FRERE1
1 SING MUSIC OF “1! 3! 5! 1!” “2 2 2 2”
END
TO FRERE2
1 SING MUSIC OF “5! 6! 8!” “2 2 4”
END
TO FRERE3
1 SING MUSIC OF “8! 10! 8! 6! 5! 1!!” “1 1
1 1 2 2”
END
TO FRERE4
1 SING MUSIC OF “1 -8! 1!” “2 2 4”
END
```

```
TO FREREJACQUES
1 FRERE1
2 FRERE1
3 FRERE2
4 FRERE2
5 FRERE3
6 FRERE3
7 FRERE4
8 FRERE4
9 FREREJACQUES
END
```

Figure 1 Two ways to play Frère Jacques *using Music Box (from Papert and Solomon 1971): (top) individual notes are sent to the synthesizer by concatenating the individual note values into a their total durations; (bottom) the SING command takes two familiar arguments—semi-tones and note durations.*

1.2 Forty years later: Voices from the present

The growth in sophistication of synthesizers has largely paralleled the growth of computing; digital music has become mainstream. Programming languages such as Barry Vercoe's Csound (Vercoe 1986) give unprecedented control over music. Sophisticated tools and apps have been developed on these platforms. Music professionals have access to tools almost unimagined in the early 1970s. But in regard to elementary education, those who were the target users for Logo and Music Box, the advances in music have been less pronounced.

There are sophisticated music applications targeting children that aim to help a novice compose music. Their user interfaces are intuitive and approachable, but they provide little access to tools essential to explore underlying musical structures beyond reusing precomposed phrases.

Csound has been used as the underlying engine for many music applications, but its design is intended neither for children nor beginning programmers. There are a plethora of programming environments for children, notably Scratch (`http://scratch.mit.edu`), which do provide some easily accessible mechanisms for exploring music. However, when it comes to exploring music, these programming environments have notable drawbacks. When programming music in Scratch software, for example, there is a block to generate a tone of a specific frequency and duration (in some extensions of Scratch these tones can be specified using MIDI notation), but any higher-level musical manipulation tools, even ones that are essential to music, must be programmed entirely from scratch. While it can be fun to program music with Scratch, the approach taken tends to resemble the “very bad” example described by Papert and Solomon as they lack the mechanisms to create larger, concrete musical structure.

There are a variety of tools used routinely in music education. Tools popular for teaching and learning musical dictation and ear training tend to be limited to music dictation and have no creative component. Furthermore, they expect that the end-user is proficient in reading Western music notation—there are no representations for learners who may be unfamiliar with notation. Tools popular for transcription and musical engraving (music notation) are used to create sheet music with affordances for composition. A variety of representations are available (such as guitar tablature and chord charts), which may be more useful to those more familiar with these systems. However, it would be daunting to use this software to explore music's rudiments in a creative and straightforward way. Tools popular for sampling and recording let you play, mix, and record songs/pieces. There is no requirement that the user be versed in language specific to music. However, none are geared towards open-ended exploration of the fundamentals of music. Furthermore, many of these tools are proprietary—students are not able to view or study the internals of the tools and they may be “locked in” to using a proprietary tool in the future.

1.3 Microworlds

Papert used the term “Microworld” to describe the world of geometry explored when children used Logo. A microworld is a “subset of reality or a constructed reality whose structure matches that of a given cognitive mechanism so as to provide an environment where the latter can operate effectively. The concept leads to the project of inventing microworlds so structured as to allow a human learner to exercise particular powerful ideas or intellectual skills.” (Papert 1980)

In a microworld, an individual is able to use a technological tool for thinking and cognitive exploration that would not be possible without the technology. But not just any technology. “The use of the microworld provides a model of a learning theory in which active learning consists of exploration by the learner of a microworld sufficiently bounded and transparent for constructive exploration and yet sufficiently rich for significant discovery.“ (Papert 1980)

There are some precedents for microworlds of music. Evgenia Sendova used a Logo microworld to enable learners to test hypotheses about musical phenomenon such as melody, rhythm, and harmony (Sendova 2001). Viera Proulx created a Java library that is used to design classes that represent musical phrases, melodies, chords, and effects (Proulx 2010); this addition of musical structure led students to add musical effects and to manage more complexity in their programs.

Music Blocks provides a collection of technological tools specific to a microworld of music, starting with pitch and rhythmic note values, but also providing affordances for repetition, transposition, etc. Music Blocks is more than an interface to a synthesizer and more than a transcription/engraving tool—it is a scalable and modular collection of essential building blocks which are at the crux of all powerful ideas in music.

Minsky remarked, “Until you understand something more than one way, you don't really understand it.” Music Blocks encourages children to explore multiple ways of understanding concepts in music. (With programming, as with music, there is rarely one “right way” to do something.) The very recognition that there are multiple ways to solve problems gives learners a deeper understanding of the world around them. Music Blocks gives the learner the ability to seamlessly connect ideas across domains: music, graphics, mathematics, sensors, etc.

With Music Blocks, there are no black boxes; the learner is given full liberty to view and to study what something does and how it does it. The source code to every block is available and the license and means to modify or extend the language are provided. Taking something apart and reassembling it in different ways is a method to understanding it.

Music Blocks users have the freedom to run, copy, distribute, study, change, and improve the software. By using Music Blocks, they become members of a community that openly shares programs they have written. Any one who uses the software has the right to use the program as-is, or to adapt it, without needing permission. Free/Libre Software is a culture that encourages every child to be a creative force within their community, and to take ownership—and the responsibility that comes hand-in-hand with ownership—of the tools that they use for learning.

2. Power Piece: An approach to understanding music

Learning with Music Blocks is based on an approach known as “power peice”. A power piece is a melody or a song that is taught because it is powerful and becomes more powerful as it is taught. A power piece is authentic, archetypal, and timeless, yet historically relevant, integrative, and infinitely malleable. *Twinkle-Twinkle Little Star*, a popular English lullaby sung to the tune of the French melody *Ah! vous dirai-je, maman,* published in 1761, is a classic power song. Mozart recast it as a highly embellished theme in his set of *Twelve Variations*. Since its inclusion as a Suzuki-method staple, it has become even more widely known and transformed, further expanding its potency. Its simple and elegant design allows for embellishment, refinement, and pedagogical repurposing (e.g., to teach new rhythms, as a vehicle for learning harmony, etc.).

The *power piece* approach differs from just learning more repertoire. Musical pieces are chosen, studied, and manipulated as a means to deepen one's understanding of musical concepts. For example, studying even the first section of the jazz standard *Autumn Leaves* (originally *Les feuilles mortes,* published in 1945 by Joseph Kosma), in all twelve keys in various ways (melody, harmony, guiding tone lines, with improvisation) would deepen a learner's understanding of the archetypal and important minor cadence in jazz.

Music Blocks brings the building blocks of music to the fore. It starts with music's most fundamental parameters: pitch and rhythmic note values. Combining the two parameters creates a note with a definite pitch and duration. Ordering notes in a sequence creates melody. Stacking different notes of the same rhythmic note value at the same point in time creates harmony. Once a melody has been created (with or without chords), "chunks" can be created and combined to create higher-level musical forms. Building in complexity, Music Blocks software provides blocks to manipulate pitch and rhythmic note values through repetition, transposition, note value augmentation and diminution—just to name a few. These tools, because of their inherently fundamental nature, become a powerful metaphor when scripting power pieces.

Take, for example, *Frère Jacques* which exemplifies the following musical concepts: perpetual canon/round (each sub-phrase can be performed with any other sub-phrase to create pleasant harmony in a major key); repetition (each sub-phrase repeats twice); and scale degrees 1–6 of a major scale. When tasked to dictate *Frère Jacques* with Music Blocks, a resourceful user will do the following: (1) Put repeat blocks around each chunk so that they repeat; (2) Create and save each of the four sub-phrases as "chunks" that can be ordered, reordered, and manipulated; (3) Find a way—through adding rests or by staggering the order of the chunks—to achieve the harmonious effect of the canon that is exemplified by *Frère Jacques; and* (4) Further experimentation and discovery of possibilities; e.g. Does it sound good with three repeats? What if one voice repeats three times while the others repeat twice? What happens when I transpose a voice or two or three? The possibilities are endless.

This type of problem solving is not very different from what a student of music theory might do to analyze a piece of music. Music Blocks places all of the analytic tools in the foreground and makes them easily accessible with minimal music-specific jargon.

3. Music Blocks

Turtle Blocks (Bender, et al. 2015) is an activity with a Logo-inspired graphical "turtle" that draws colorful art based on snap-together visual programming elements designed to run in a web browser. Music Blocks (`http://walterbender.github.io/musicblocks`) is a "fork" from Turtle Blocks (i.e. built from Turtle Blocks' code). It also runs in a browser. It expands upon Turtle Blocks in that it has a collection of features relating to pitch and rhythm. Many of the ideas in Music Blocks were inspired by the music-in-education ideas, representations, and practices developed by Lawrence Scripp (Scripp et al. 2014).

In the sections that follow, we describe some of the numerous extensions we made to Turtle Blocks in order to provide affordances to access powerful ideas in music: what makes Music Blocks a Musical Microworld, as opposed to yet another block-based programming language.

3.1 The graphical-notation matrix

We chose to use a matrix (cartesian coordinate grid for pitch value vs. note duration) for two reasons: (1) because not everyone who uses Music Blocks is expected to be proficient at Western musical notation; and (2) the matrix representation allows for flexibility and scalability.

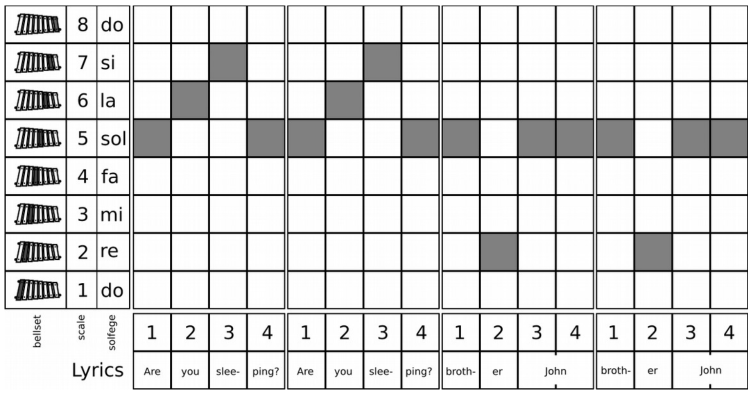

Figure 2: A matrix used for learning how to perform two portions of the melody Frère Jacques *on bellsets. This handout was used at the MusicLaunch community music program at Boston Chinatown Neighborhood Center as a tool to help beginner students collaborate with more advanced students in a performance of this well-known round. The three columns on the left are helpful in understanding how solfege, numbering, and bellset-location are all representations of the same concept.*

If you imagine that the collection of pitches represent a musical bell set, then adding/removing pitches is like adding or removing keys to a bell set. This has the benefit of focusing the musical creation process into a particular pitch-space (e.g. “key” or mode) for their intended sound world. Too often, music software programs present the user with a large keyboard of notes, which provides all of the “wrong” (unintended) notes alongside all of the “right” (intended) notes indiscriminately and side by side. Plus, most singable melodies fall within a register of less than an octave and singing one’s musical creations is very good ear-training. The graphical-notation matrix is designed for this type of mindful, creative practice (See Figure 2).

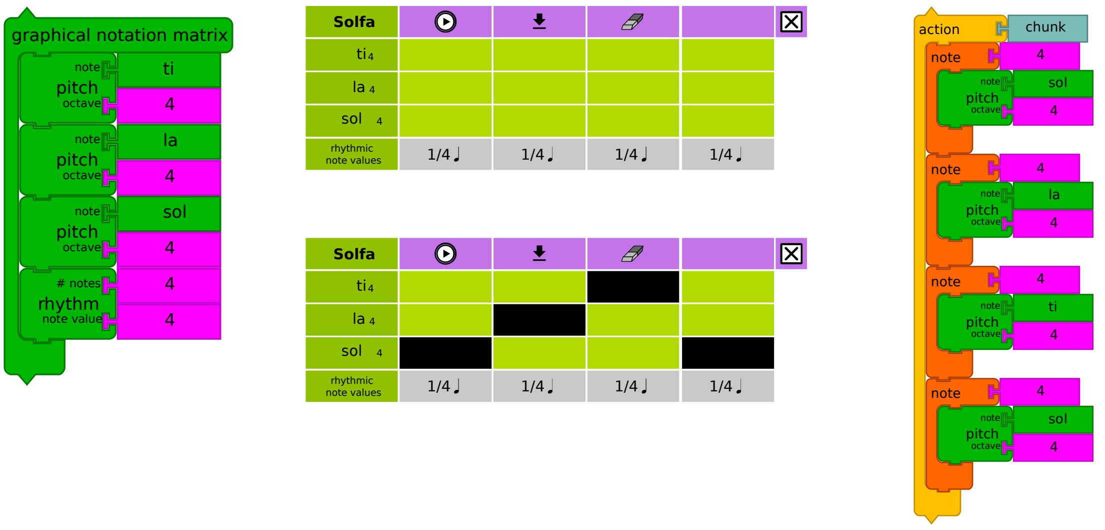

Figure 3: On the left is a stack of blocks used to generate a graphical-notation matrix (three pitches in octave 4 and four quarter notes). In the center top is the resulting matrix. In the center bottom, several notes have been selected in a matrix; On the right, a stack of blocks (chunk) generated from the matrix.

Having launched Music Blocks, a user starts by clicking on the *Graphical-notation Matrix* block (See Figure 3, left). A grid organized vertically by pitch and horizontally by rhythm appears (Figure 3, top). Because the matrix had three *Pitch* blocks, there are three rows, one for each pitch. (A row at the bottom specifies the rhythms associated with each note.) There are four columns for selecting notes because a *Rhythm* block was used to specifies four quarter notes.

By clicking on individual cells in the grid, the user will hear individual notes (or chords if she clicks on multiple cells in a column). In Figure 3 (center bottom), four quarter notes are selected (black cells). If the user clicks on the Play button (found in the top row of the grid), she will hear a sequence of notes played (from left to right, with their respective rhythmic values): sol, la, ti, sol (G4 E4 F4 G4). Once the user has a group of notes (a "chunk") that she likes, she clicks on the Save button (just to the right of the Play button). This will create a stack of blocks (See Figure 3, right) that can used to play these same notes programmatically (more on that below). The user can rearrange the selected notes in the grid and safe other chunks as well. Since she can define as many chunks as she wants, the user is free to experiment.

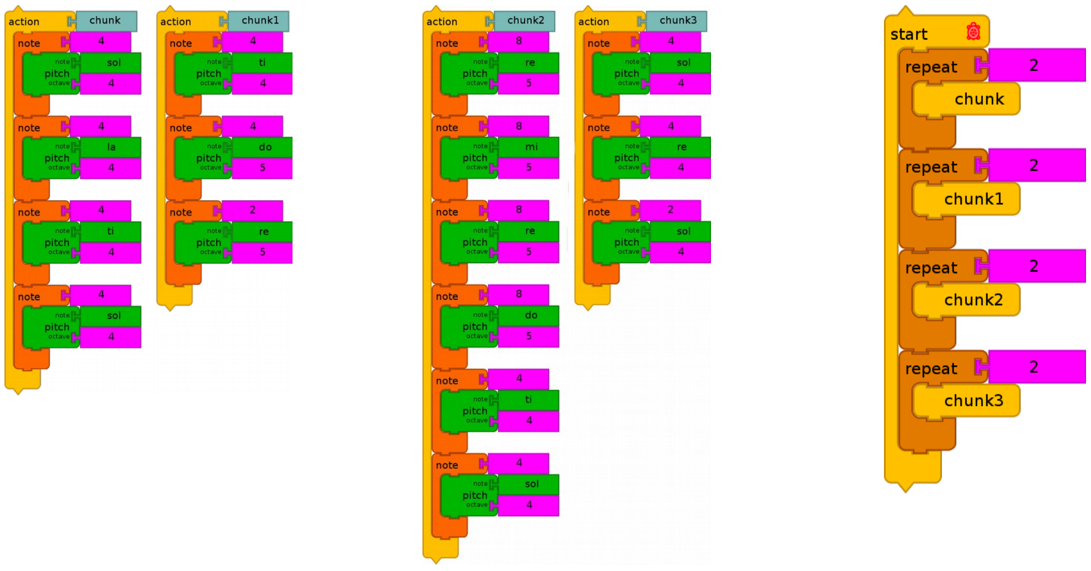

Figure 4: Frère Jacques *programmed in Music Blocks. Note the similarity to the Music Box example shown at the bottom right of Figure 1.*

The chunk created when the user clicks on the matrix *Save* button is a stack of blocks. The blocks are nested: an *Action* block contains *Note* blocks, each of which contains at least one *Pitch* block. The *Action* block has a name automatically generated by the matrix, in this case, chunk (the user can rename the action). Each note has a duration (in this case 4, which represents a quarter note). We encourage users to put different numbers in and see (hear) what happens. Each *Note* block also has a *Pitch* block (if it were a chord, there would be multiple *Pitch* blocks nested inside the clamp of the *Note* block). Each *Pitch* block has a pitch name (Re, Mi, and Sol), and a pitch octave; in this example, the octave is 4 for each pitch. To play the chunk, the user simply clicks on the action block (on the word “action”). She will hear the notes play, ordered from top to bottom with their respective rhythmic durations.

Pitch blocks are used inside the *Matrix* block to indicate pitches to select. They are also used inside of *Note* blocks to specify the pitch(s) to be played. The user can plug different values into the Pitch block slots for name and octave. The pitch name can be specified using a *Solfege* block (e.g. Do, Re, Mi); a *Pitch-name* block (e.g., C, D, E); or a *Number* block (frequency in Hertz). The octave is specified using a *Number* block and is restricted to whole numbers.

Rhythm blocks are used to generate rhythm patterns in the matrix. The top argument is the number of notes. The bottom argument is the duration of the note. Multiple *Rhythm* blocks may be used inside the *Graphical-notation Matrix* block. For more complex rhythmic patterns, *Tuplets* is a tool to create sets of rhythmic values that are not based on a power of 2. The user can mix and match *Rhythm* and *Tuplets* blocks when defining grids. The user can also specify individual notes and chords in the matrix.

3.2 Programming with music

The chunks created from the matrix are used for programming. (The user can also create or modify chunks by hand, without utilizing the matrix). Music Blocks creates a new block specific to each chunk. Clicking on, or running, this block has the same effect as clicking on the chunk.

She can repeat chunks either by using multiple chunk blocks or using a *Repeat* block. She can also mix and match chunks. Adding a few more chunks results in *Frère Jacques* (See Figure 4).

3.3 Beyond Music Box

In provisioning our musical microworld, we have added tools that can be used to further explore music's fundamentals. Music Blocks includes many ways to transform pitch and rhythm: (1) For pitch transformation, we have tools such as *Sharp*, *Flat*, and *Transposition*. The *Sharp* and *Flat* blocks modify pitch to *Pitch and Note* blocks, or even chunks. The *Transposition* block can be used to make larger shifts in pitch to individual pitches or to all pitches contained within a chunk. To shift an octave up or down, transpose by ±12 half-steps. (2) For rhythmic transformation, we have tools such as *Rhythmic Dot*, *Multiply* or *Divide Beat Factor*, and *Rhythmic Tie*. To "dot" a note, a common musical transformation, use the *Dot* block. A dotted note extends the rhythmic value of contained notes by 50%. The *Multiply/Divide Beat Factor* blocks multiply or divide contained rhythmic note values by the factor defined by the user. (3) For exploring and transforming larger musical form, we have *Repeat* and *Duplicate* blocks. The *Repeat* block will play a sequence of notes or chunks multiple times in their given order; the *Duplicate* block will repeat each note specified. (4) Each *Start* block runs as a separate musical voice in Music Blocks; when you click on the “Run” button, all of the *Start* blocks created are run (performed) concurrently. This feature can be used to achieve musical counterpoint (multiple voices performed simultaneously with differing pitch and note values) to create complex harmony.

There are exciting possibilities when combining the math and logic functions of the original Turtle Blocks code with the Music Blocks extensions. For example, the user can program music which chooses random parameters—rhythmic note value, octave, pitch in Hz, etc.—to create moments of aleatory and improvisation in their compositions. Students can use math and logic to create the harmonic series and find different ways to program rhythmic dots and rhythmic ties.

Music Blocks also has tools that take advantage of a computer's sensors, which means that interactive performances are possible. For example, we could reuse the chunks from *Frère Jacques* as follows: when the mouse is in the lower-left quadrant, chunk0 is played; lower-right quadrant, chunk1; upper-left quadrant, chunk2; and upper-right quadrant, chunk3.

```
mouse = {
g'4 a'4 b'4 g'4 g'4 a'4 b'4 g'4 b'4 c''4 d''2 b'4 c''4 d''2 d''8 e''8 d''8 c''8
b'4 g'4 d''8 e''8 d''8 c''8 b'4 g'4 g'4 d'4 g'2 g'4 d'4 g'2}
\score { << \new Staff = "treble"
{ \clef "treble" \set Staff.instrumentName = #"mouse" \mouse } >> \layout { } }
```

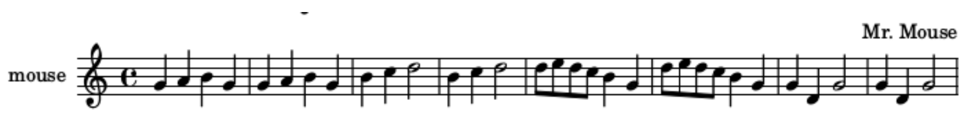

FIgure 5: The output of a Music Blocks program can be saved as a Lilypond format file (“mouse” is one of our default (and make-believe) musical instruments); the compiled Lilypond output.

A *Save as Lilypond* block transcribes a composition (`http://www.lilypond.org/`) (See Figure 5). Our goal in building a bridge between Music Blocks and Lilypond (which is also Free/Libre Software) is to give the learner both the ability to communicate with the mainstream world of music and access to a rich set of tools that they may use to further augment their exploration of music. Music Blocks, by design, does not confine a user to its tools—rather it is a tool to propel the ambitious learner to other rich and authentic musical discoveries.

4. Preliminary Conclusions

Why a microworld for music? According to Papert, a microworld must be sufficiently bounded and transparent for constructive exploration and yet sufficiently rich for significant discovery. Similarly, Music Blocks aims to be a musical microworld—an open-ended, yet powerful and musically relevant tool—one that invites learners of varying backgrounds to explore fundamental musical concepts that are both intrinsic to music yet transcendent of a specific discipline.

When discussing microworlds, however, Papert also speaks of the need for guidance, both in the microworld itself and in the teacher’s facilitating a child's exploration of it. Therefore, workshop projects are being actively developed and refined to provide guidance and inspiration to teachers and learners in their use of Music Blocks. These projects use an approach that builds from the *Power Piece* approach mentioned above. Since *Power Pieces* introduce rich musical ideas that can be studied, analysed, transformed, and re-imagined, they are ripe for open-ended explorations as part of workshops. Students may, for example, be given a power piece to work on, along with guided inquiry questions that lead to investigation about music and music integration. Performances and presentations of musical creations at the end of a workshop should be playful, fun, and motivating. Facilitated dialogue after performances and presentations should be opportunities for critical thinking and reflection. The first examples of student programs written in Music Blocks demonstrate the efficacy of the microworld approach.

Why programming and why Free/Libre Software? Forty years of observation suggests that engaging children in using computers *and in programming software to run on computers* is a powerful means to drive learning. The process of writing and then repairing (also known as “debugging”) a program, which Solomon described as “the great educational opportunity of the 21st Century”, provides a basis for active learning through heuristic problem solving. Free Software is enabling student contributions to the Music Blocks design, documentation, and code.

References

Bamberger, J.S. (1991) *The Mind Behind the Musical Ear: How Children Develop Musical Intelligence*, Harvard University Press, Cambridge, pp. 104.

Bender, W., Solomon, C., and Urrea, C. (2015) “(More than) Twenty Things to Do in Turtle Blocks”, *Proceedings of Constructionism 2015*, Vienna, Austria.

Fredkin, E., and Minsky, M. (1970) *Digital Music Synthesizer*, US Patent 3610801.

Papert, S. (1980). Computer-based microworlds as incubators for powerful ideas. In R. Taylor (Ed.), *The computer in the school: Tutor, tool, tutee* (pp. 203–210). Teacher’s College Press.

Papert, S. and Solomon, C. (1971). “Twenty Things To Do With A Computer” *MIT AI Laboratory Memo* 248.

Proulx, V. (2010) Music in Introductory Object Oriented Programming, *Proceedings of Constructionism 2010,* Paris, France.

Sendova, E. (2001) Modelling Creative Processes in Abstract Art and Music, Eurologo 2001, *Proceedings of the 8th European Logo Conference* 21–25 August, Linz, Austria.

Scripp, L., Ulibarri, D., Southerland, S., Gilbert, J., Sienkiewicz, F., Swihart, A. N., and Swihart, E., (2014) *Principal Investigator's Final Report: The Oakland Unified School District’s Music Integration Learning Environment (MILE),* Arts in Education Model Development & Dissemination (AEMDD) Project.

Vercoe, B. (1986) *Csound: a manual for the audio processing system and supporting programs with tutorials*.

Acknowledgments

The authors would like to thank Google for their generous support of this work through Summer of Code (GSoC) and Google Code-in programs and the students who participated. Marnen Laibow-Koser helped advise the GSoC intern. Glen Larsen advised on Lilypond. The Sugar Labs community has been very supportive of the project, as has been Eduardo Marturet. We would also like to thank Cynthia Solomon for encouraging us to write this paper.