



Music Blocks Lesson Plan

Debugging

Age:

7-12 years

Lesson duration:

60 minutes

- Introduction: What is “Debugging”? (15m)
- Part 1: Looking inside (20m)
- Break (5m)
- Part 2: Slowing it down (15m)
- Performance/Critique (10m)

Number of students:

Up to 10.

Rationale:

Students will learn about different approaches to debugging their programs.

Objectives:

Students will understand how to use a collection of different debugging techniques that are applicable to Music Blocks and other problem spaces.

LESSON

Introduction:

Begin by asking students to sit in a circle and explain that in today's lesson they are going to learn about “debugging”.

Did your Music Blocks program ever do something different than you expected or wanted? Or not work at all? How did you figure out what was wrong? How did you fix it?

Debugging is about figuring out what is wrong (and why something doesn't work) so that you can fix it. There are many ways to debug. We'll explore several different ways in this lesson.

Programming is hard. Composing music is also hard. Both programming and composing involve some trial and error and serendipity. Inevitably you will make mistakes along the way. Music Blocks provides a number of mechanisms, reviewed below, to help you explore ideas and find mistakes.

Some thoughts on debugging:

Learning is hard fun.—Marvin Minsky

Make the complicated comprehensible—Arthur Miller

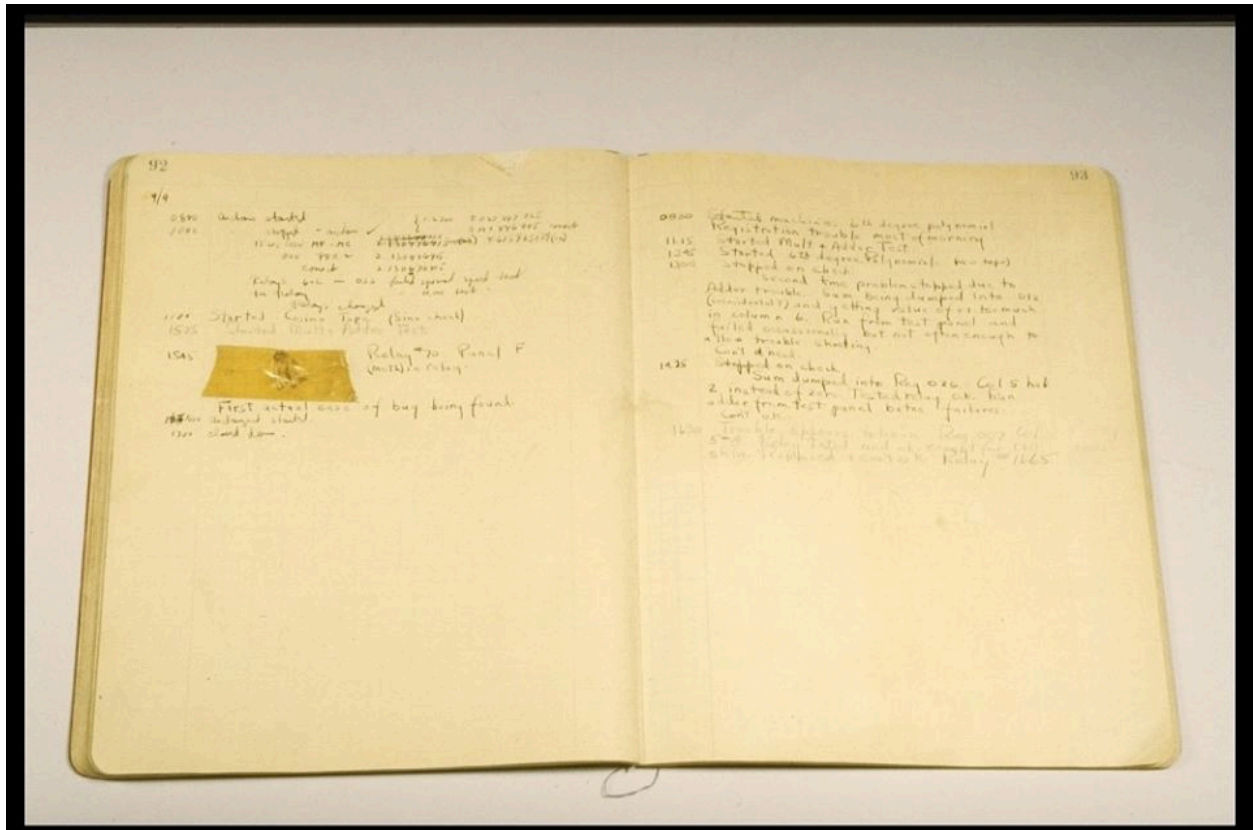
Debugging is the learning opportunity of the 21st Century. — Cynthia Solomon

The important message that comes from ideas about debugging is that we learn from our mistakes; that the intricate process of making things work or learning new skills has to do with hypothesizing, testing, revising, etc.—Cynthia Solomon

Sometimes bugs are serendipitously adopted as features worth perpetuating, sometimes procedures must be constructed to deal with the phenomena caused by their appearance, and sometimes the bugs and their side effects need to be removed. But in this pursuit, children become creative researchers studying behavior, making up theories, trying out ideas, etc.—Cynthia Solomon

6 Stages of Debugging—Anonymous

1. That can't happen.
2. That doesn't happen on my machine.
3. That shouldn't happen.
4. Why does that happen?
5. Oh, I see.
6. How did that ever work?



"American engineers have been calling small flaws in machines "bugs" for over a century. Thomas Edison talked about bugs in electrical circuits in the 1870s. When the first computers were built during the early 1940s, people working on them found bugs in both the hardware of the machines and in the programs that ran them.

"In 1947, engineers working on the Mark II computer at Harvard University found a moth stuck in one of the components. They taped the insect in their logbook and labeled it "first actual case of bug being found." The words "bug" and "debug" soon became a standard part of the language of computer programmers."

From https://americanhistory.si.edu/collections/search/object/nmah_334663

Part 1: Looking inside.

A. Clicking on an individual stack of blocks

The Play button will run all of the Start blocks simultaneously. But you can also run an individual stack of code by clicking on a stack. This lets you test and debug small sections of code or play a single voice of a multi-part composition by clicking on one of the Start blocks or single phase by clicking on one of the Action blocks.

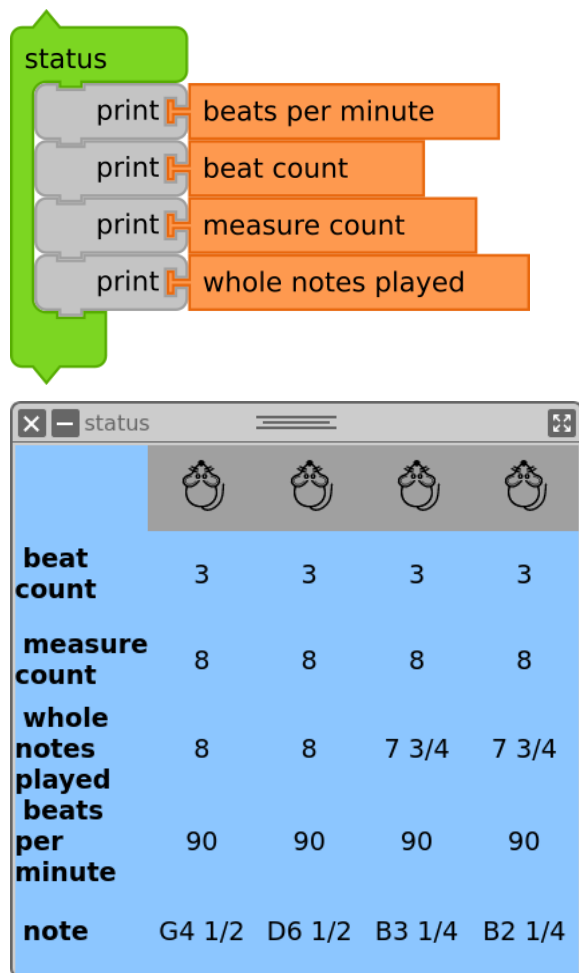
B. Print and Comment blocks

The Print block can be used to print a message while running a program. It is useful to determine if a section of code is being executed when expected or if a box or parameter contains an expected value.

The Comment block (also found on the Extras palette) is similar to the Print block, except it only prints a message when the program is being run in Slow mode (See below). Comments are also written to the browser console.

Try using Print and Comment blocks in your favorite project to check on values.

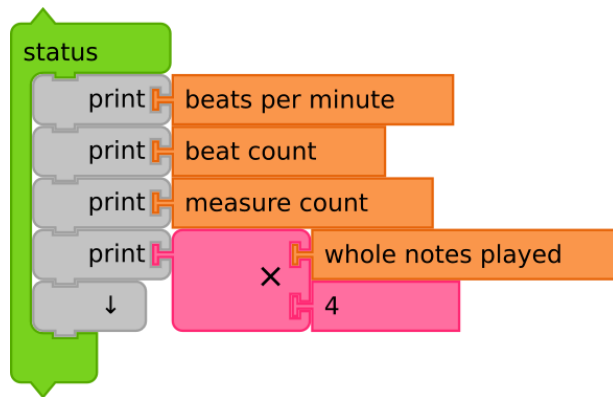
C. The Status widget is a tool for inspecting the status of Music Blocks as it is running. By default, the key, BPM, and volume are displayed. Also, each note is displayed as it is played. There is one row per voice in the status table.



The image shows a green status widget with four 'print' blocks connected to the following text: 'beats per minute', 'beat count', 'measure count', and 'whole notes played'. Below this is a screenshot of the 'status' window, which displays a table with four columns representing different voices. The table has five rows of data: 'beat count', 'measure count', 'whole notes played', 'beats per minute', and 'note'. The 'note' row shows musical notation for each voice: G4 1/2, D6 1/2, B3 1/4, and B2 1/4.

beat count	3	3	3	3
measure count	8	8	8	8
whole notes played	8	8	7 3/4	7 3/4
beats per minute	90	90	90	90
note	G4 1/2	D6 1/2	B3 1/4	B2 1/4

Additional Print blocks can be added to the Status widget to display additional music factors, e.g., duplicate, transposition, skip, staccato, slur, and graphics factors, e.g., x, y, heading, color, shade, grey, and pensize.



You can do additional programming within the status block. In the example above, whole notes played is divided by 4 (e.g. quarter notes) before being displayed.

Break

Part 2: Taking it slow

A. Play Buttons

Clicking on the Play button will play your program at full speed. (It will also hide the blocks while the program runs, which improves performance.) But there are two other playback modes.

On the Secondary Menu, there are two other Play buttons.

During Playback Slow mode the program will pause between the execution of each block and the block being executed will be highlighted. This is useful for following program flow, ensuring that the sequence of blocks being executed is what you expect. In addition, the value stored in any box or parameter is displayed on the block as the program runs, so you can "inspect" program elements as the program runs.

Run Step by Step advances one block per button press.

B. Show and Hide blocks

The Show and Hide blocks (found on the Extras palette) are useful for setting "breakpoints" in your program to debug a specific section of code. By putting a Show block at the start of a problematic section of code and a Hide block at the end of the section, your program can be run full speed until it gets to the Show block. Then the blocks are displayed and run in Playback

Slow mode. When the Hide block is encountered, the blocks are hidden and the program resumes running at full speed.

Try running your favorite Music Blocks program in slow motion. Can you follow the program flow?

C. Browser console

As Music Blocks runs, some debugging information is written to the browser console, such as the notes being played and comments (See the Comment block above). The console can be accessed by typing Ctrl-Shift-J on most web browsers.

```
running activity.js:514
overwriting session data activity.js:2632
notes to play: G4 1/4 logo.js:7368
notes to play: E4 1/4 logo.js:7368
notes to play: G4 1/2 logo.js:7368
fin logo.js:6721
overwriting session data activity.js:2632
>
```

Shown above is the console output from three notes: sol mi sol.

Performance/Critique:

1. Have each student talk debugging.
2. Engage in a discussion about their different approaches.

Key events:

- Introduction of key concepts: debugging, inspection, break points.

Materials:

- Music Blocks software

Assessment:

- Observe participation.
- Does the program perform as expected?



