

# Google Summer of Code

## Sugar Labs

Music Blocks Project Blocks Reorganization

---

### Basic Information:

<b>Name</b>	Daksh Doshi
<b>Github</b>	<a href="#">daksh4469</a>
<b>LinkedIn</b>	<a href="#">Daksh Doshi</a>
<b>Email</b>	<a href="mailto:ddaksh.2711@gmail.com">ddaksh.2711@gmail.com</a>
<b>Phone</b>	+91-7874034708
<b>IRC Nickname</b>	daksh4469
<b>Location</b>	Ahmedabad, India
<b>Time Zone</b>	India (UTC + 5:30)
<b>Languages</b>	Gujarati/Hindi, but I am proficient in speaking, reading, writing, and understanding English.

### University Information:

- University: Indian Institute of Technology, Roorkee
- Majors: Computer Science and Engineering
- Current: II Year (Graduation expected in 2023)
- Degree: Bachelor of Technology (4 Year Program)

## Contact Information:

Typical working hours include:

- UTC 0400 - UTC 0700 hrs (IST 0930 - IST 1230)
- UTC 0830 - UTC 1400 hrs (IST 1400 - IST 1930)
- UTC 1600 - UTC 1930 hrs (IST 2130 - IST 0100)

I can start my day 2 hours early or late and will be reachable anytime through my Mobile No. and Email.

## Skills:

Programming Languages and Frameworks:

- Fluent in HTML, CSS, SCSS, Javascript, Typescript, C/C++, Java, Python
- Sound knowledge of OOPs
- **Web Frameworks:** React, Django, Django-REST, Nodejs, Express, Mongoose
- **Libraries:** Bootstrap, EJS, Material UI, jQuery, Socket.IO, CUDA Library, C++ Standard Library
- **Databases:** MySQL, MongoDB
- **Utilities:** Figma, Postman, Firebase, MongoDB Atlas, Heroku, Docker

Development Environment:

- **Ubuntu** 20.04
- **Visual Studio Code** as IDE supported by a range of extensions.
- **Linux Shell**
- Chrome Dev Tools
- **Git** for version control

Apart from the technologies listed above, I have sound knowledge of MERN Stack and MVC Architecture. My other interests include Web Design and Prototyping, and Competitive Programming.

## About Me:

I am Daksh Doshi, a sophomore at the **Indian Institute of Technology, Roorkee**, where I am pursuing Computer Science and Engineering. I was introduced to software development and programming in my freshman year. Since then, I have explored various fields such as Cryptography, Web Development, Data Structures, Algorithms, and Computer Architecture. I developed a passion for Web Development and Design since my first semester at college. I have been learning new technologies and their applications every day since. I have been contributing to open-source for about three months now and am adoring it. I have been working with a team to refactor the entire website of UBA-IIT Roorkee. [UBA](#) (Unnat Bharat Abhiyan) is an organization that aims to aid the rural areas of India by addressing the issues and solving them through appropriate and sustainable technologies and by organizing suitable events.

## Past Projects:

- EndGem: ([Git Repository](#))
  - A full-stack Web Application built on NodeJS through Express.js Framework and MongoDB Atlas on the back-end and EJS templating on the front-end along with Passport Authentication for users.
  - Built to organize different types of documents according to their courses. Features include downloadable content, top downloads to date, and the ability to upload and delete documents.
- CovidWelfare: ([Frontend Git Repository](#) | [Backend Git Repository](#))
  - A full-stack Web Application, built on React(JS) for front-end and Django on the back-end with a REST API built using Django REST Framework.
  - Built to enable people to remotely help people in need of resources in the trying times of Covid-19. Connects the users by SEEK and PROVIDE functionality.

- Provides the real-time locations of users with the help of Google Maps API. Incorporates a Notification System to notify the users in need of resources and the provider users.
- Sorting Visualizer: ([Git Repository](#))
  - A React(JS) based Web Application to enable visualization of Bubble Sort Algorithm.
  - Includes the feature of manually setting the array size for incorporating a better understanding of the algorithm.

This is not an exhaustive list of all the projects. Some of my other projects can be found on my GitHub profile [here](#).

## Contributions to Sugar Labs:

I am an active contributor to Sugar Labs for the past 2.5 months now. This has helped me understand the codebase of [musicblocks](#) better, and I now feel comfortable working with it. Contributing to musicblocks has helped me comprehend how it works internally and understand the interactions between various components. I am working with React (Javascript) for the past year and have built several projects using it. Recently, I have been learning Typescript and its applications with React library. Now, I feel pretty comfortable using Typescript with React.

My contributions in [sugarlabs/musicblocks](#) can be widely classified into three categories: Porting to ES6/Linting/JSDoc/Documentation/Refactoring, Bug/Regressions Fix, and Enhancements/Features added. I have been contributing to Music Blocks, and the statistics of my contribution are given below:

- Pull Requests (PRs): 43 (27 merged, 8 open, and 8 closed)
- Commits: 58 (1564++, 1185--)
- Issues: 1 (1 closed)

## Pull Requests:

### sugarlabs/musicblocks:

#### Porting to ES6, Linting, JSDoc, Documentation and Refactoring:

- ❖ [#2760 \(merged\)](#): Update a function to ES6 Arrow Function in js/toolbar.js
- ❖ [#2764 \(closed\)](#), [#2773 \(closed\)](#): Add class to js/widgets/temperament.js and port ES6 to syntax.
- ❖ [#2810 \(closed\)](#), [#2811 \(merged\)](#): toolbar.js: Prettify, Linting and JSDoc documentation
- ❖ [#2812 \(merged\)](#): widgets/jseditor.js: Linting and Prettify
- ❖ [#2814 \(closed\)](#): utils/musicutils.js: Prettify, linting and JSDoc Documentation
- ❖ [#2817 \(merged\)](#): utils/platformstyle.js: Prettify, Linting and JSDoc Documentation
- ❖ [#2818 \(merged\)](#): basicblocks.js: Prettify, linting and JSDoc Documentation
- ❖ [#2819 \(merged\)](#): mxml.js: Prettify and Linting
- ❖ [#2821 \(merged\)](#): notation.js: Linting and prettify
- ❖ [#2824 \(merged\)](#): widgets/statistics.js: Add JSDoc Documentation
- ❖ [#2827 \(merged\)](#): FAQ/README.MD: Fixed some typos and grammatical mistakes.
- ❖ [#2830 \(merged\)](#): turtleactions/DictActions.js: Linting and Prettify
- ❖ [#2831 \(merged\)](#): Linting and Prettify: All files in js/turtleactions
- ❖ [#2833 \(open\)](#): blocks.js: Linting, prettify and removed debug logs
- ❖ [#2835 \(merged\)](#): blockfactory.js: Add global locations and constructor JSDoc
- ❖ [#2905 \(open\)](#): palette.js: Linting and Prettify
- ❖ [#2915 \(open\)](#): musickeyboard.js: Add ES6 Class

#### Bug/Regressions Fix:

- ❖ [#2807 \(merged\)](#): Fix Bug in Arbitrary Edit Tab and improvements in Tempo Widget.
- ❖ [#2837 \(merged\)](#): pitchstaircase Bug Fix: Sound keeps playing even after closing the widget.
- ❖ [#2845 \(merged\)](#): Bug Fix, Temperament Widget: Play and Stop not working properly
- ❖ [#2848 \(open\)](#): phrasemaker.js: Bug Fix, Linting and Prettify.
- ❖ [#2854 \(merged\)](#): Bug Fix: Pitch Staircase Widget

- ❖ [#2863 \(merged\)](#): Fix regressions in MusicKeyboard widget
- ❖ [#2878 \(merged\)](#): Bug Fix: Tooltip of Collapse Icon
- ❖ [#2891 \(open\)](#): Bug Fix: phrasemaker widget plays when no notes are added
- ❖ [#2900 \(open\)](#): statistics.js: Add global locations and bug fix
- ❖ [#2902 \(open\)](#): BugFix: rhythmruler widget does not render individual pause buttons

### **Enhancements/Features added:**

- ❖ [#2776 \(merged\)](#): Update UI of Temperament Widget
- ❖ [#2832 \(merged\)](#): WidgetWindows: UX Enhancement
- ❖ [#2838 \(merged\)](#): widgets/status.js: Improved UI of status widget
- ❖ [#2855 \(merged\)](#): Oscilloscope: Error Fix and UI modification
- ❖ [#2841 \(closed\)](#), [#2857 \(merged\)](#): Update the MusicKeyboard widget on maximizing.
- ❖ [#2874 \(merged\)](#): Enhanced the UI of search-bar and its suggestions
- ❖ [#2903 \(open\)](#): pitchdrum-mapper: Implement Stop Functionality and enhance UI

### **sugarlabs/musicblocks-v4-lib:**

- ❖ [#27 \(closed but helped in #28\)](#): Port musicutils.py to Typescript.
- ❖ [#29 \(merged\)](#): Add testcases to musicutils.test.ts
- ❖ [#33 \(closed\)](#): Update Scale section in README.md
- ❖ [#34 \(open\)](#), [#49 \(merged\)](#): Add documentation for musicUtils.ts
- ❖ [#42 \(merged\)](#): Add error testcases to musicutils.test.ts

### **Issues:**

- [#2872 \(closed\)](#): Blocks lose their color on hovering over them

## **Commits:**

While contributing to Sugar Labs, I have made a total of 61 commits (till the date of writing). All of these commits can be found [here](#).



(Contribution Statistics of sugarlabs/musicblocks repository from Jan 17 - Mar 27)

---

## Project Details:

**Title:** Music Blocks Project Blocks Reorganization

**Coding Mentors:** [Anindya Kundu](#), [Walter Bender](#)

**Assisting Mentors:** [Peace Ojemeh](#)

Music Blocks is being refactored to [musicblocks-v4](#). This project aims to address the issues related to the presentation of the blocks in the canvas in the current version of musicblocks(version 3) and build upon them to design and implement the blocks' organization in musicblocks-v4.

As Music Blocks is going through a complete overhaul, blocks graphics and the overall UI is also bound to change for good. Hence, this project might need some coordination with other projects i.e., blocks graphics refactoring, music blocks menus and palettes, and music blocks debugging aids.

### Project Tasklist:

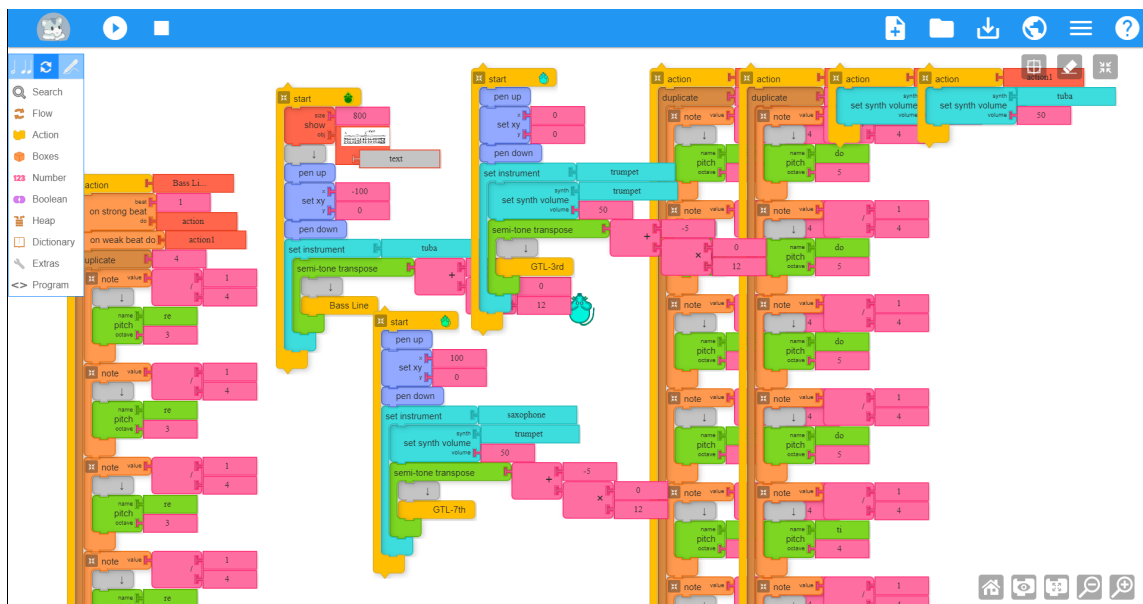
- Familiarize yourself with the current implementation.
- Come up with a framework for better project structuring.
- Design a flexible guidance layer on the top of the canvas.
- Implement the above in React (Typescript).

### The Problem:

In the current version of [musicblocks](#), there exist numerous issues concerning the presentation and organization of the blocks on the canvas. These issues are as follows:

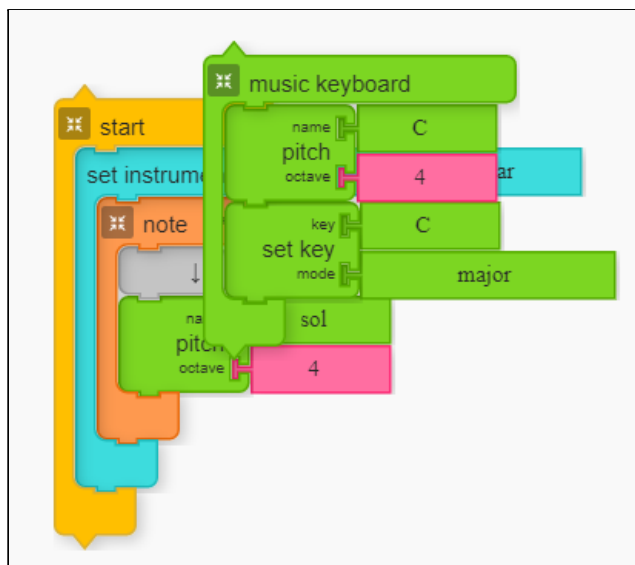
1. Currently, a user can arbitrarily place any block in any position, and thus, it gets quite **overwhelmingly crowded** when a considerable amount of blocks are present in a project. This gets quite noticeable and hampers the project management experience of a user. An example project where this is visible is shown below (from Planet > Global):





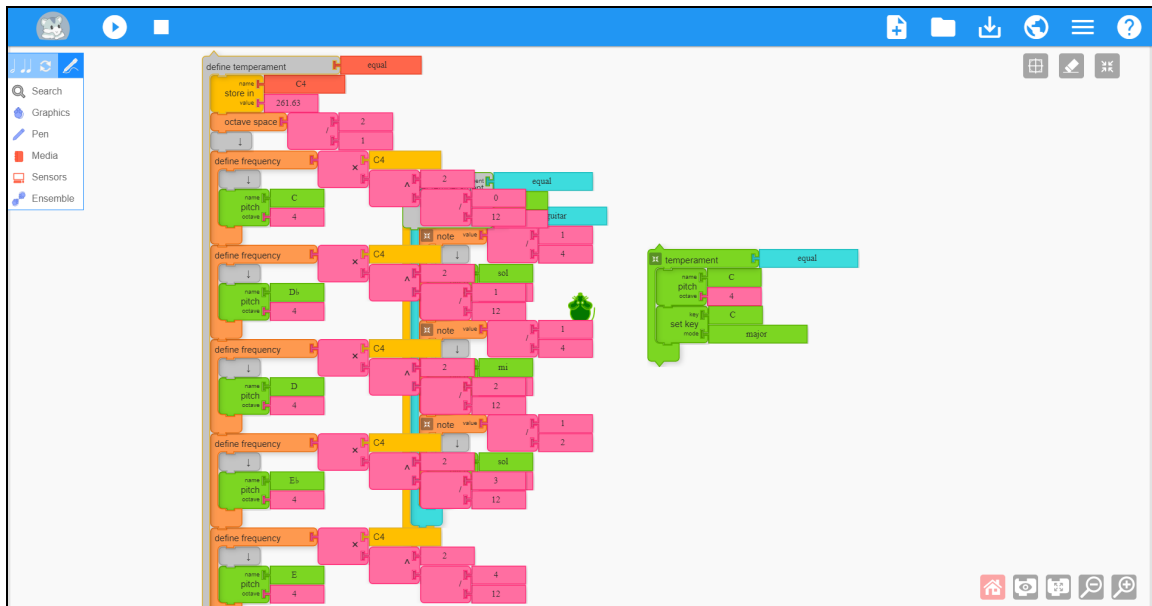
(Project Reference: 12 Bar Blues with a Guitar Tab)

- The **blocks can overlap each other** without any constraints whatsoever. In the current version of musicblocks, the block or block-stack that a user has been more recently engaged with overlaps the other blocks around it. A simple example depicting this is shown below:



(Here, the **music keyboard** widget block overlaps the **start** block as it has been more recently worked with.)

- There is a feature in musicblocks to 'clamp' some blocks. However, this feature is only present in some of the blocks. The issue mentioned in the 2<sup>nd</sup> point affects these **clamped blocks** heavily as they can become **almost entirely invisible** to a user if it gets overlapped by another block or block-stack.
- As mentioned above, the feature to clamp is only available in some of the blocks. Now, sometimes the **width of a block-stack** can get congesting, and sometimes, there is no option to inline-clamp these blocks. This results in large parts of the blocks in the project getting overlapped by a single block-stack. A simple example representing this is generating the action block through the temperament widget. This is shown in the below image:



(In this musicblocks project, the **start** block and the **set temperament** block are not visible due to overlapping by the **define temperament** block, majorly due to its large width.)

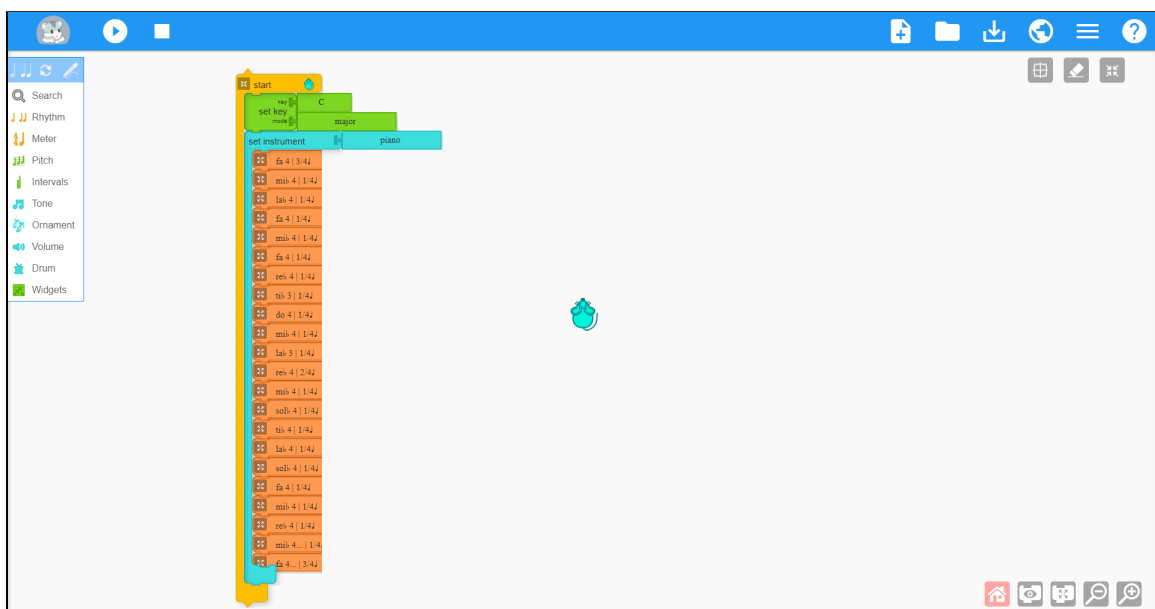
- Adjusting the **placements of the blocks can be quite tedious**. If the project is quite large, complex, and utilizing many blocks, the task of dragging a block/block-stack through the entire project can be pretty overwhelming.

## The Solution:

The initial idea is to introduce **swim lanes** to guide a user to encourage better project management. Swim lanes can help a user arrange the blocks in columns and help **better project management** and much cleaner projects. I believe that a user should have a significant role in choosing how he/she wants to organize the blocks as each project and each user is different. Thus, this utility should be optional for a user to choose as, most likely, this feature would be much more helpful in more extensive and more complex projects than in beginner projects. We can include this as an **Advanced Mode or High Shelf feature** (depends on the new UI/UX). This utility has the following deliverables:

- **Provide the user the choice to use this feature or not:**

A user can choose to utilize the feature and can also disable (stop using) it. This feature is **more likely to be used in more extensive and complex projects** than in small beginner projects. For example: In the below-shown image of a musicblocks project, there is no such need to organize the blocks in swim-lanes. It uses just a single block-stack that can be managed comfortably with the standard organization.



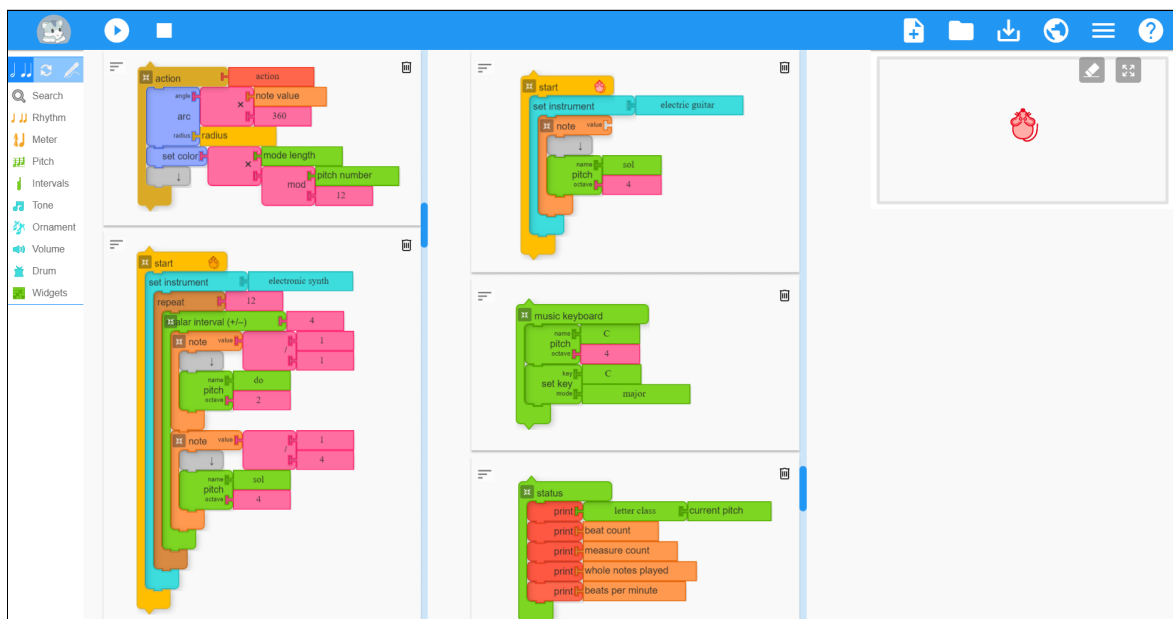
(Project: SandyChopinBerceuse from Planet > Global)

- **Set the number of swim-lanes:**

If the feature is enabled, a user can set the number of swim-lanes according to the project. However, there may be an **upper limit to the number of swim-lanes**, which can be calculated by the maximum width of a block/block-stack present in the project. However, we can also pre-define this upper limit based on . We may need to manage one issue: *the variance of the number of swim-lanes to the block-size*, which can be increased or decreased using the “Increase Block Size” and “Decrease Block Size” buttons, respectively.

- **Individually scrollable swim-lanes:**

One potential advantage that swim-lanes can provide is the freedom to individually manage each swim-lane. Each swim-lane can be scrollable which gives the user the freedom to manage individual swim-lanes intensively rather than scrolling through the whole project altogether. **This resolves the tedious job of dragging** a block/block-stack through the whole project. In the case of individually scrollable swim-lanes, a user can simply drag a block/block-stack from one lane to another without holding and dragging it through the entire canvas. An inceptive prototype of implementing swim-lanes:



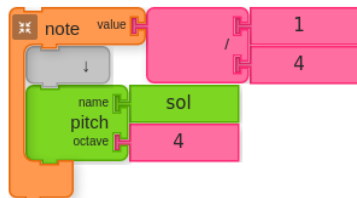
(Note: The above design is made considering the current block design, and could be modified after refactoring the block design in musicblocks-v4.)

- **Collapse the turtle stage:**

One aspect to keep in mind while developing the swim-lanes is the placement of the mouse in the canvas. In the presence of swim-lanes, the mouse could adequately be placed using the `Collapse` feature. **Also, on clicking the Play button, the swim-lanes would hide just like the blocks in the current version of musicblocks.**

- A significant aspect in developing swim-lanes is each swim-lane's width since every swim-lane will have a unique set of blocks or block-stacks. The upper-bound to the number of swim-lanes is also determined by the width of block-stacks. Thus, extensive widths can result in a low number of swim-lanes. One way of handling this is by providing the user **the option to set the widths**(probably by dragging) of each swim-lane. Another way of handling this is by **adding the "inline-clamp" functionality** to some more blocks. In the current version of musicblocks, there are only three **INLINECOLLAPSIBLES** blocks:

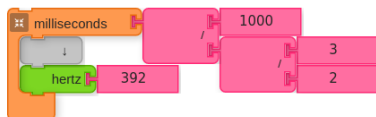
- newnote (note value block)



- interval (scalar interval (+/-) block)



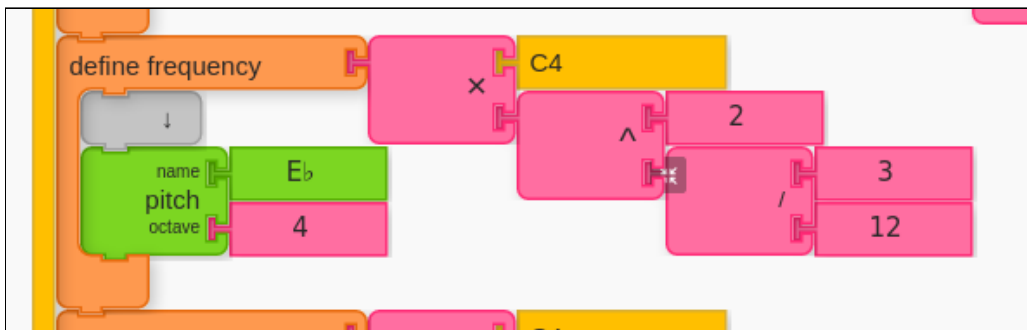
- osctime (milliseconds block)



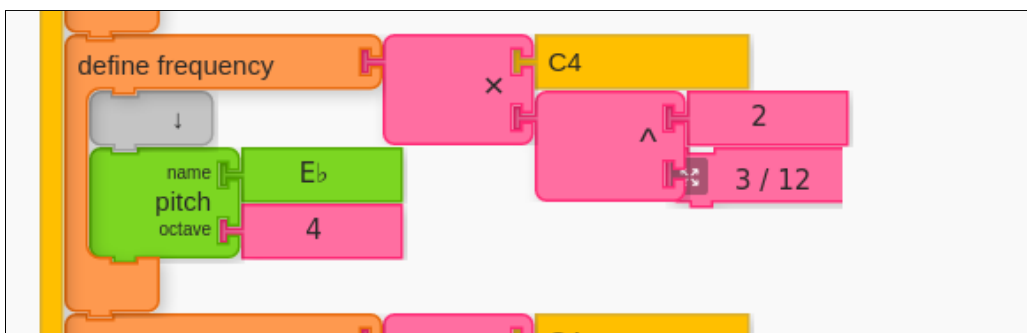
However, in most of the cases (including the Define Temperament one mentioned on Page 10), the width of a block-stack is increased due to nested number-blocks (mainly add(+), subtract(-), multiply(x), divide(/), mod and power(^)).

One idea is to **make these number blocks inline-collapsible**. This would help reduce the overall width of a block-stack and thus, fewer cases of overlap in the regular block organization. A very basic implementation of this would involve representing the answer, after operation between the connected numbers, as the name of the clamped block.

Before clamping the divide block:



After clamping the divide block:



Above shown is a very initial example of this idea. The label "3 / 12" can be calculated using the following function in the `block.js` file of the current MB.

```

// function to allot the text to be presented on clamping the "divide" block
_divideLabel() {
  const number1 = this.connections[1]; // first block connected to the divide ("/") block
  const number2 = this.connections[2]; // first block connected to the divide ("/") block
  const num = this.blocks.blockList[number1].value; // value of the first block
  const den = this.blocks.blockList[number2].value; // value of the second block

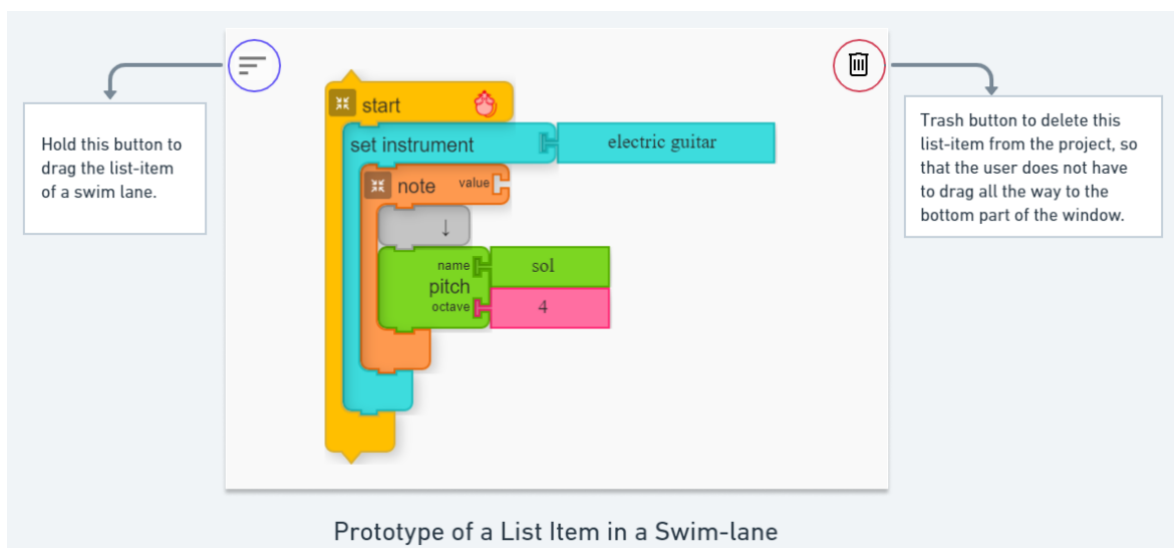
  //by default, this.collapseText.text = "divide";
  if(number1!=null && number2!=null) {
    this.collapseText.text = num + " / " + den; //text to be appeared on clamping
  }
}
}

```

Similarly, if all the number blocks are made inline-collapsible, widths of the block-stacks could be reduced significantly. Value of these number blocks is already being stored as the `value` field of a block in the `blocklist` in the current version of musicblocks. Thus, this idea would work similarly in case of *nested number blocks*.

- **Transfer of blocks/block-stacks:**

We can maintain a block-list and their positions of each swim-lane and a user can comfortably drag and drop a block/block-stack from one swim-lane to any position in some other swim-lane after scrolling to the desired position. This also helps in **solving the overlap issue**, as now, on dropping (“onmouseup”) event, the block/block-stack will get positioned in the swim-lane accordingly.



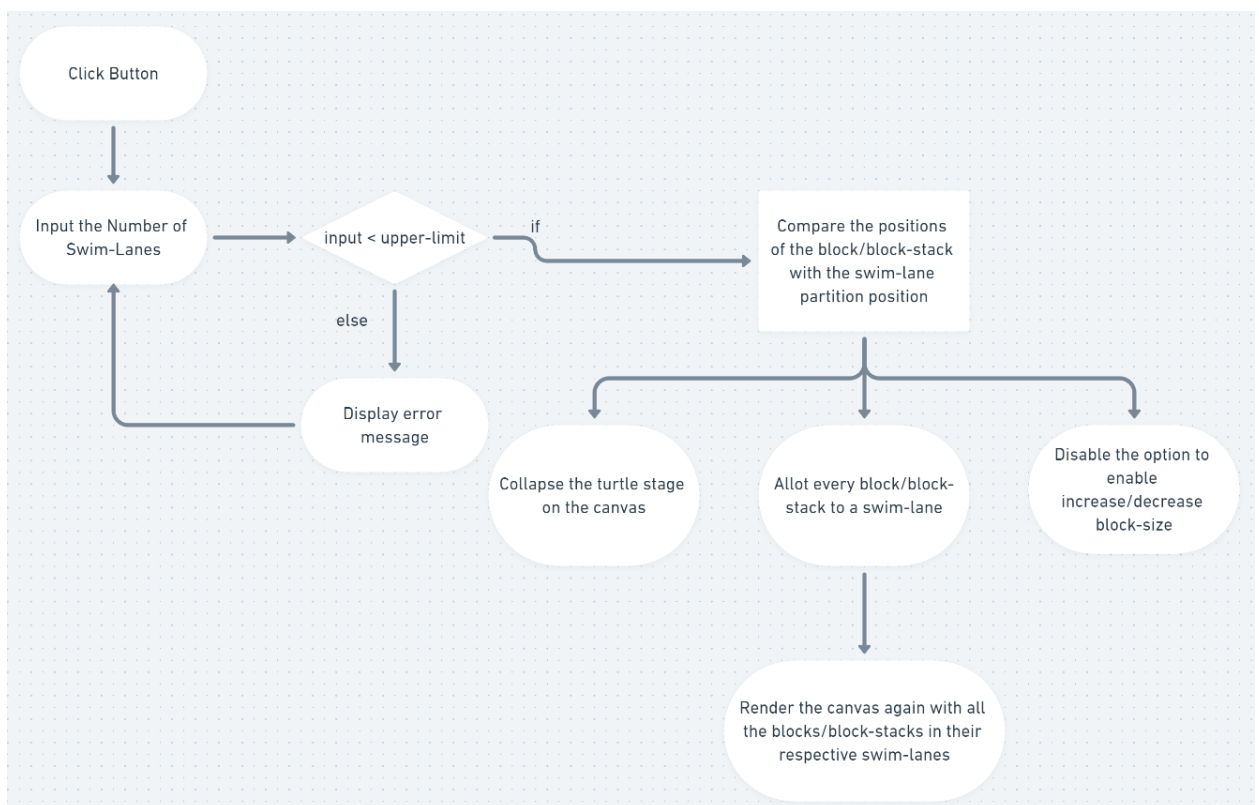
- The **Trash functionality** would also be needed to manage in presence of swim-lanes as currently, one has to drag a block/block-stack to the bottom of the screen and hover over the displayed trash icon. However, since the blocks/block-stacks are presented as list items in a swim-lane, it would be **preferable to add a Trash icon for every list item** (just as in any ToDo list, but not as checkbox) containing a block/block-stack.
-



## Summary of the Proposed Solution:

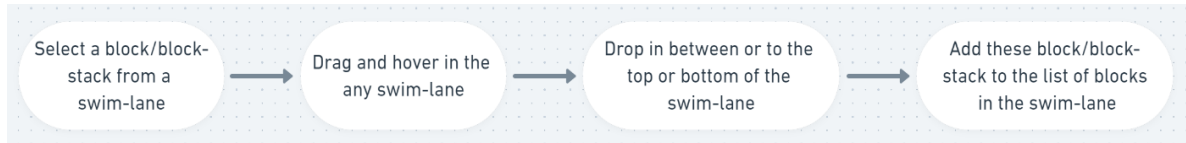
1. Add a feature (in advanced mode/high-shelf) to **organize the blocks in swim-lanes**. (*optional for user to enable/disable*)
2. **Every swim-lane will be scrollable**, hence, providing the user the advantage of managing each set of block/block-stacks individually.
3. **The user can set the number of swim-lanes** up to an upper limit based on the width of the block-stacks, and the turtle stage would be collapsed.
4. Each block/block-stack will be arranged in a swim-lane as a **draggable list item**. Also, the turtle stage would need to be collapsed so as to incorporate these swim-lanes. One idea is that maybe, the collapsed turtle stage could appear by a pop-up.

Below shown is the workflow activity diagram describing the process of rendering the swim-lanes organization of a project.



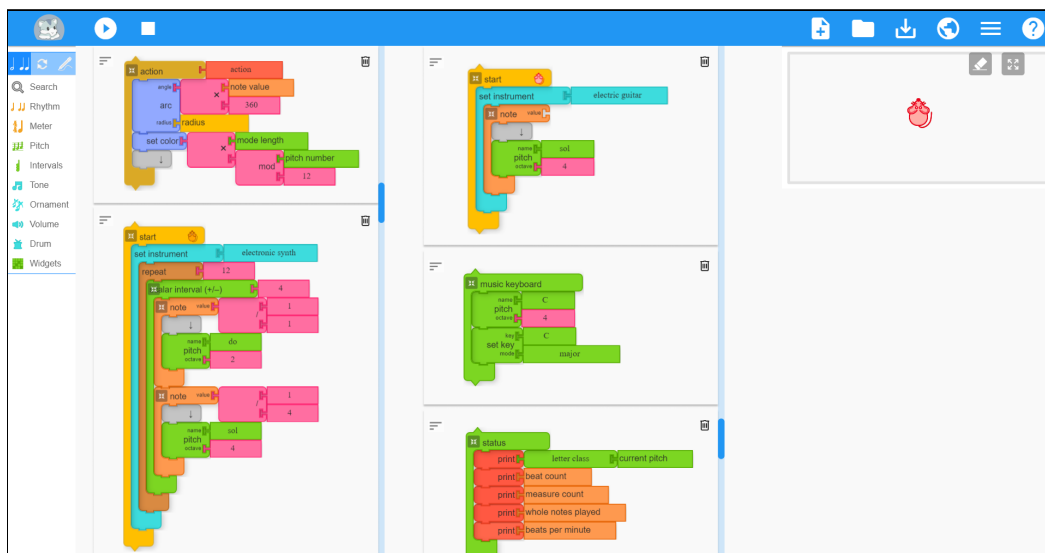
(The given flowchart shows the workflow of the proposed swim-lane feature)

5. The order of blocks/block-stacks can be arranged in a swim-lane by dragging and dropping.
6. A block/block-stack can also be added by “click” in palette menus. In this case, it will be **rendered on the top of the active swim-lane**. A block/block-stack can be **moved from one swim-lane to another by dragging and dropping**.

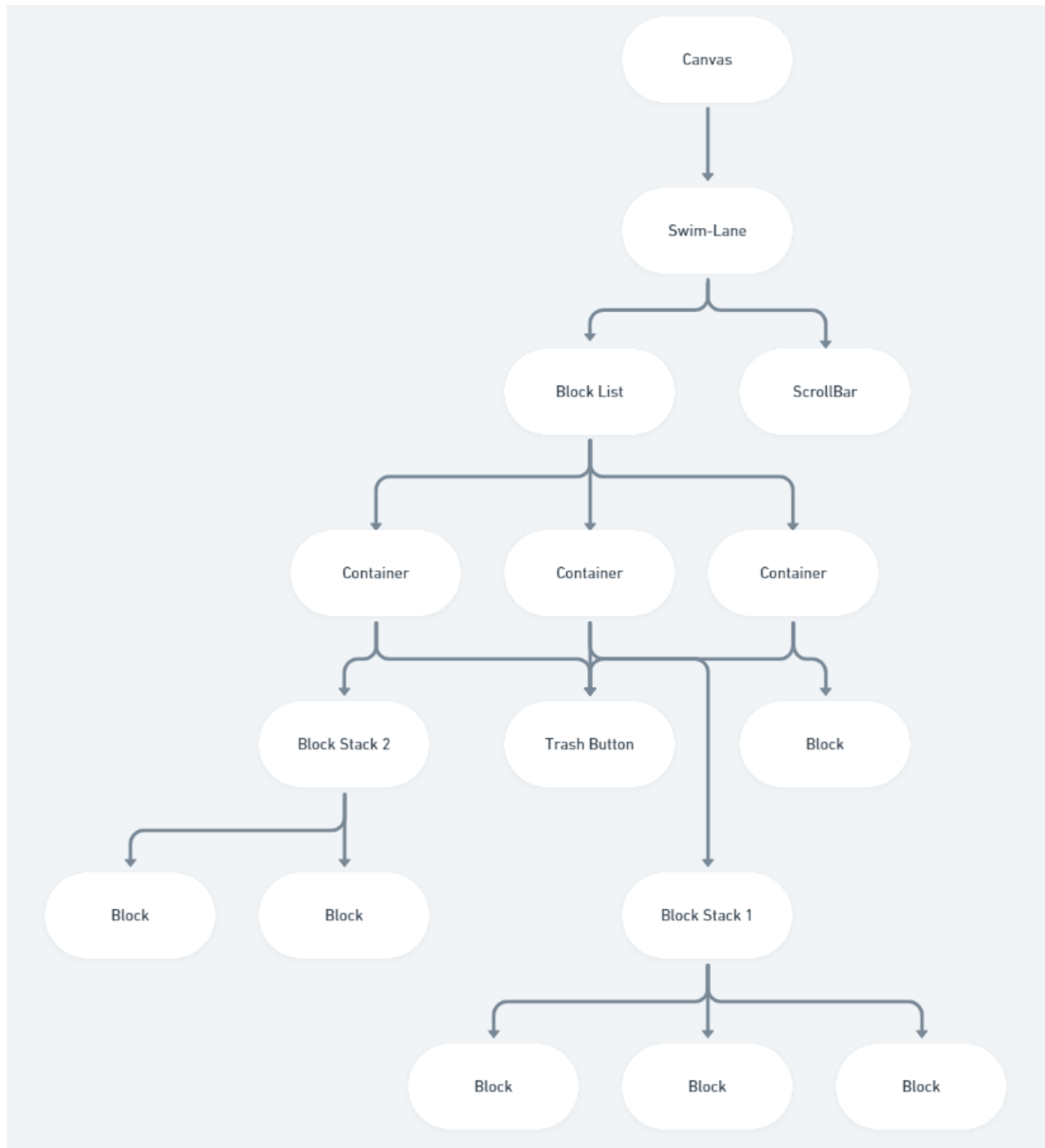


7. Issues that may need handling:
  - a. The feature of **increasing and decreasing the block size**.
  - b. The **width of certain block-stacks** may result in a decrease of the upper limit of the number of swim-lanes.
  - c. A user should not be allowed to enable **horizontal-scroll** while using this feature.
8. Utility ideas:
  - a. Some more blocks may be added to the **INLINESCOLLAPSIBLES** blocks so as to control the width of a block-stack.
  - b. **Resizable swim-lanes** by adjusting the width by dragging, and **enable horizontal-scroll to incorporate more swim-lanes without any constraints**.

An inception idea representing the idea, in the current version of musicblocks, is shown below (also shown on Page 12):



Using React can be a boon here as it would be assistive in developing every list-item as a component and every swim-lane as an individual component utilizing the list-item's components. Every swim-lane would be *mapped* in the canvas when the user opts to use this feature. The blocklist is each swim-lane can be managed using `useState()` and `useEffect()` hooks. Below is a sample component tree of a swim-lane in the canvas of musicblocks-v4.



(Component Tree of a single swim-lane in the canvas)

**Timeline:**

<b>Pre GSoC</b>	<ul style="list-style-type: none"> <li>Familiarize myself with the current implementation.</li> <li>Continue contributing to Sugar Labs.</li> </ul>
<b>Community Bonding</b> (17th May - 7th June)	<ul style="list-style-type: none"> <li>Get to know the community better and bond with the mentors and developers.</li> <li>Receive feedback on this project if something needs to be amended.</li> <li>Explore the tech stack required for the project.</li> </ul>
<b>Week 1</b> (7th June - 14th June)	<ul style="list-style-type: none"> <li>Discuss and finalize the ideas to be implemented concerning this project in musicblocks-v4.</li> </ul>
<b>Week 2</b> (14th June - 21st June)	<ul style="list-style-type: none"> <li>Work out the final framework, workflow and the structure for this project.</li> </ul>
<b>Week 3 - 4</b> (21st June - 5th July)	<ul style="list-style-type: none"> <li>Decide the title and icon of this feature after discussion with the mentors.</li> <li>Setup the re-render of canvas on enabling this feature with block/block-stacks allotted to a swim-lane.</li> </ul>
<b>Week 5</b> (5th July - 12th July)	<ul style="list-style-type: none"> <li>Review the work done with the mentors and make necessary amendments.</li> </ul>
<b>Evaluations</b>	
<b>Week 6-7</b> (12th July - 26th July)	<ul style="list-style-type: none"> <li>Work on the transfer of block/block-stacks between swim-lanes and the rearrangement of the block-list in a swim-lane.</li> </ul>
<b>Week 8</b> (26th July - 2nd August)	<ul style="list-style-type: none"> <li>Discuss and work on any new enhancements in UI or functionalities.</li> <li>Add the necessary guide/documentation for this feature.</li> </ul>
<b>Week 9</b> (2nd August - 9th August)	<ul style="list-style-type: none"> <li>Testing of the feature.</li> <li>Cleanup and wrapping up the work.</li> </ul>
<b>Week 10</b> (9th June - 16th June)	<ul style="list-style-type: none"> <li>One week buffer to compensate for any delay.</li> </ul>

## **How many hours will you spend each week on your project?**

My college summer vacations are scheduled to take place from 10<sup>th</sup> June to 2<sup>nd</sup> August, which is almost the whole of the coding phase. I will be able to devote around 40-45 hours a week efficiently. I have no other commitments for the summer vacations other than GSoC. So, I will be able to devote most of my time to GSoC. I am also free on weekends and will keep the community updated about my progress and maintain transparency about the project.

## **How will you report progress between evaluations?**

I will be active on GitHub as I will be continuously working on the project while interacting with the mentors. Thus, my progress will always be reported thoroughly on GitHub. I am also planning to write weekly or fortnightly blogs about my progress in the project. I will be reachable anytime through IRC, Email, or a planned video-session.

## **Discuss your post GSoC plans. Will you continue contributing to Sugar Labs after GSoC ends?**

After GSoC, I plan on continuing my contributions to Sugar Labs as I am amazed by the community relations and the work carried by this organization. I will contribute to the ongoing issues and the enhancements in the organization as there is always a scope of betterment on the web. I vision to hone my skills further and put them to use to give back to the community. I aim to develop mentorship skills and the ability to guide others and try to give back to the community by mentoring and guiding others. I hope to mentor future GSoC students.

**Looking forward to contributing to Sugar Labs this summer season.**

**Kind Regards.**