



Sugar Labs:

Music Blocks Block Graphics Refactoring

Meta Info:

>Personal Details:

Name: Jaikishan Brijwani

Github: [ricknjacky](#)

IRC/Matrix nick: @ricknjacky

Slack-MB: @ricknjacky

Email: jaikishanbrijwani@gmail.com

Language: English

Location: Mumbai, India IN

Time Zone: IST (UTC + 5:30)

>Education and Background:

University: Manipal University Jaipur, India

Majors: Computer & Communication Engineering

Current: Junior Year (expected graduation in 2022)

Degree: Bachelor of Technology (4 Year Program)

>Contact and Working Hours:

Reachable anytime via **Email**, **Matrix** or **Slack**.

- 09:00-13:00, 16:00-19:00, 22:00-02:00 (IST)
- 03:30-07:30, 10:30-13:30, 16:30-20:30 (UTC)

Personal background:

I am a 20 year old junior, currently pursuing my BTech(Bachelor of Technology) in **Computer and Communication Engineering** at **Manipal University Jaipur**. I first got acquainted with computers and aspects of software development in high school. The great and ever-increasing utility of the same continued to fuel my desire to learn more. In pursuit of exploring different fields of computer science, I began learning web development in my freshman year and worked on a couple of personal projects. I also learnt different languages pertaining to Software Development like C++, Python, Dart etc.

I was introduced to the world of open source in my sophomore year. Since then, most of my time went into learning new technologies and developing software using them. I have **been contributing to open-source regularly since about six-seven months** now.

I have experience working closely with a team courtesy of a couple of internships I have done. I have been an active member of the Web Development team of [E-Cell MUJ](#) and [FOSS MUJ](#). While working on my projects I adhere to very strict timelines and make sure to follow the best practices to create impacts of the highest significance.

Programming Skills:

<u>Languages</u>	JavaScript, TypeScript, HTML, CSS, SCSS, Python, GoLang, C/C++, PHP, SQL.
------------------	---

<u>Web Frameworks</u>	React, Redux, Node, Django, Django REST, Flask
<u>Testing</u>	Jest
<u>Libraries</u>	Socket.IO, JQuery, Bootstrap
<u>Utilities</u>	InkSpace SVG, Adobe Xd, Docker+Compose, CI+CD Pipeline, Postman, Firebase

Development Environment :

- **Windows 10**
- **VSCode as IDE supported by a range of extensions.**
- **Windows PowerShell**
- **Chrome Dev Tools**
- **Git for version control**

Apart from the technologies listed above I have sound knowledge of Object-Oriented Programming, architectures like MVC, MTV and MERN/MEAN stack.

My other interests include **Information Security, Blockchain, DevOps** and **Machine Learning**. Along with contributing to open-source projects, I am working on some personal projects, exploring and learning WASM.

My projects:

[Dev Jobs](#)

- React based WebApp that uses [GitHub Jobs API](#)
- Used React Hooks and advanced state management
- The app is responsive and works fine on all devices, differing in pixel dimensions
- Used SCSS partials for styling individual components

[Chat-App](#)

- Developed a real-time, bi-directional chat engine
- Used React and Express for Front-End
- Used Socket.IO and Node.js Backend for implementation

[To-Do App](#)

- A personalized to-do app using nodejs
- Used ejs templating engine for UI/UX.

I am not listing all the projects here. My other projects can be viewed on my [GitHub profile](#).

Open Source Contributions:

- **Sugar Labs:** I have been working with and **actively contributing to Sugar Labs** projects i.e. **Music Blocks** and **Sugarizer** namely for the past 6 months. Accentuated below is the catalogue of my contributions for the same in the past.

Pull Requests: -

➔ Music Blocks

- ❖ [#2569 \(merged\)](#) : fixed the grid does not fill the available space. Fixes issue described in [#2568](#).
- ❖ [#2576 \(merged\)](#) : fixed regression with camera block. Fixes issue described in [#2574](#).
- ❖ [#2578 \(merged\)](#) : fixed discrepancy with markdown file. Fixes issue described in [#2577](#).
- ❖ [#2589 \(merged\)](#) : Introduced a new feature via which the name of the project can be loaded from the planet in the browser tab. Fixes issue described in [#2584](#).
- ❖ [#2600 \(closed\) {PR was merged manually}](#) : Guide artwork SVG files were deprecated. Collaborated with Walter Bender in updating them accordingly. Fixed issue [#2591](#).
- ❖ [#2601 \(merged\)](#) : Stats widget's documentation needed an update; I proposed the same. Fixes issue described in [#2546](#).
- ❖ [#2604 \(merged\)](#) : Oscilloscope widget was generating spam console logs even after the stop button had been pressed. I fixed it here. Fixes issue [#2563](#).
- ❖ [#2613 \(merged\)](#) : Timbre widget displayed erratic behavior while navigating through different filters. I proposed a fix for the same here. Fixed issue [#2612](#).
- ❖ [#2691 \(merged\)](#) : For some blocks, the help widget window could not fit the SVG and the text below within the scope of the window. I proposed a fix for the same. Fixes issue [#2690](#).
- ❖ [#2709 \(merged\)](#) : Master guide on `sugarlabs/musicblocks` repository had deprecated artwork, I updated the file, concurrent to the current version of musicblocks.
- ❖ [2734 \(merged\)](#) : Tour window showcased regression that arose, as the method of calculating height of help widget was dynamic.

- ❖ The following PRs fixed issue [#2711](#)
 - ▶ [#2714 \(merged\)](#)
 - ▶ [#2715 \(merged\)](#)
 - ▶ [#2719 \(merged\)](#)
- } Added new artworks and information pertaining to widgets and blocks described in the issue above.
- ❖ [#2803 \(merged\)](#) : Some SVGs displayed erratic behavior upon being rendered in different browsers, fixed that issue here.
 - ❖ [#2862 \(merged\)](#) : Temperament widget’s export functionality was not working properly due to some bugs prevalent; I fixed those bugs here. Fixed issue [#2858](#).
 - ❖ Music Blocks is being rebuilt from ground up and updating its syntax to the ES6+ syntax was a task I contributed to, ranging from porting to class definition, eliminating occurrences of “var”, “that”, adding JSDocs, syntactically prettifying the code to make it more readable etc. Following are the PRs I proposed for the same:
 - [#2688](#)
 - [#2772](#)
 - [#2674](#)
 - [#2792](#)
 - [#2675](#)
 - [#2793](#)
 - [#2676](#)
 - [#2797](#)
 - [#2755](#)
 - [#2799](#)
 - [#2759](#)
 - [#2806](#)
 - [#2769](#)

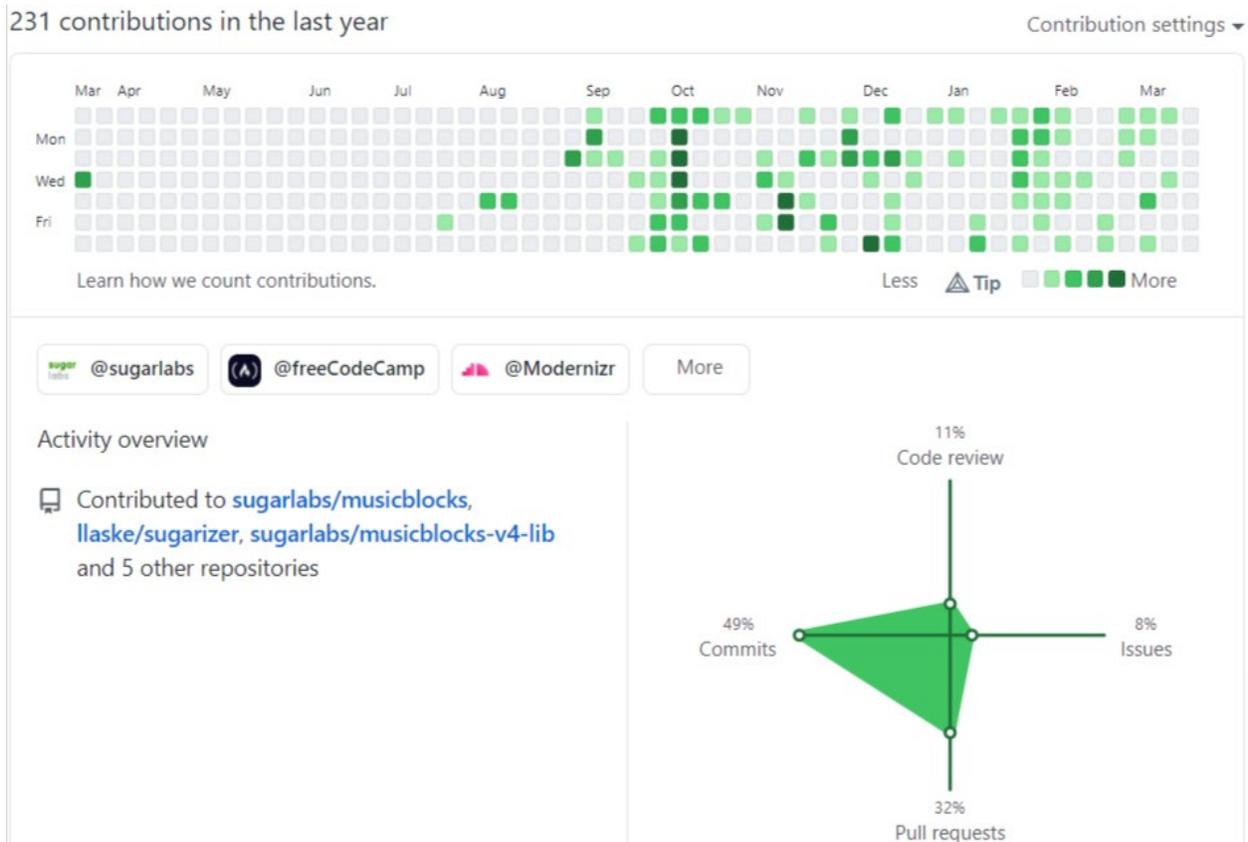
My contribution on Music Blocks repository (30 Sep 2020-15 Mar 2021)



➔ Sugarizer

- ❖ [#864 \(closed\)](#) **{PR was merged manually}**: Added functionality to allow frog to move with arrow keys in Food Chain activity. Fixed issue [#857](#).
- ❖ [#882 \(merged\)](#): Added functionality for a popup when an image is exported in Moon activity. Fixed issue [#867](#).
- ❖ [#883 \(merged\)](#): Added functionality for a popup when an image is exported in Shared Notes activity. Fixed issue [#868](#).
- ❖ [#884 \(merged\)](#): Added functionality for a popup when an image is exported in Labyrinth activity. Fixed issue [#869](#).
- ❖ [#889 \(merged\)](#): Added functionality for a popup when an image is exported in Abecedarium activity. Fixed issue [#870](#).
- ❖ [#888 \(merged\)](#): Event bubbling bug wherein the Fullscreen mode was no longer activated. Fixed it here, issue no: [#859](#).
- ❖ [#899 \(merged\)](#): Fixed bug in Color My World activity: erratically mis-interpreted North Korea to be South, vice versa, issue: [#898](#)

My contribution stats over the course of over one year are as follows:



In addition to contributing via Pull Requests that have obviated bugs, I have also **contributed** to projects by **discovering issues** and creating new issues on the projects' respective repositories.

□ **Issues:** -

- [#2598](#)
- [#2612](#)
- [#2711](#)
- [#2779](#)
- [#2839](#)
- [#2864](#)

I have also contributed by reviewing PRs at times to help maintainers. As already mentioned, Music Blocks' new core being built I am contributing to it by contributing to `sugarlabs/musicblocks-v4-lib` repository where new core "engine" of Music Blocks is being built. I have also discussed some enhancements for the same.

● **Discussions:** -

- [#2710](#) , [#2720](#), [#32](#), [#33](#)

Project Overview:

Title: **Music Blocks Block Graphics Refactoring**

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

Assisting Mentors: [Peace Ojemeh](#)

The task of block graphics refactoring is described in detail on [ideas page](#). The focus is mainly on the frontend blocks: i.e., to improve how they interlock and how they expand/resize and several UX issues. The proposal can be realized in an incremental manner, frequent pull requests safely merged into the master branch throughout the summer.

Project Plan

Deliverables:

- Come up with a framework for how the block interconnections will work.
- Design the class structure for the new block rendering approach.
- Implement all of the above in React (TypeScript).

The Problem:

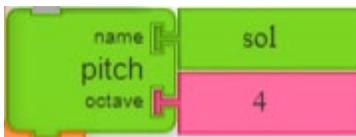
Music Blocks has been in development and being used for over 6-7 years to this date. Over the course of 20+ releases and 3 main version releases, the application has been used by an increasing number of users, a share of them being Professional Musicians, K12 kids, teachers etc. In this process, some of its underlying features have been challenged, the application is in a monolithic state with galore of “GOD” objects, UI/UX issues, regressions etc.

The current implementation of block generation is mainly handled by 2 files [blockfactory.js](#) and [protoblocks.js](#). These files handle blockfactory artwork generation and define the classes of block prototypes, respectively. Perusing through the current implementation it can be inferred that the code is overly complex and convoluted. Many parts of blockfactory are deprecated as well, making it prudent to integrate a new way of implementing how we define, generate, and use blocks.

- **Issue 1:** The design itself of the current blocks. The current design is unnecessarily cumbersome and results in poor UI experience once there are a lot of blocks in a particular project. For instance,



a) The divide block takes up a lot of space vertically and horizontally both when there are multiple note value blocks. A minimalistic simple design going forward can help us in placate this erratic behavior.



b) The pitch block as well showcases the same erratic behavior, here too we can apply horizontal minimalistic design that helps us save space.

- **Issue 2:** There should be a clear way to differentiate stale i.e., unused blocks and used one. Current implementation fails to capture this idea. The gist of what I try to convey: -



<== An implementation of such kind will help the user know which blocks are used and which ones are stale.

- **Issue 3:** Blocks can have an outline around them whenever they are selected or dragged around, added to workspace from the toolbar.
- **Issue 4:** When blocks are connected, they should produce a connection tone to let the user know of successful task performed. We can also incorporate “shadow-effect” UI behavior when the block is just about to connect.



All the above problems and consequent insights shared are primarily focused on UI only. Other issues as previously discussed for instance, convoluted codebase pertaining to block generation, artwork creation handling of multi-touch, issues due to rendering of all blocks on a single <canvas></canvas> element making the application monolithic will be at the epicenter of my work during summer.

 **The Solution:** One of the project task-list deliverables is to come up with a framework for how the block interconnections will work, make it more obvious how blocks interlock and use the interlocking to help define the schemas associated with some block logic. For this, **I propose we use BLOCKLY.**

Blockly is an open-source developer library for adding block-based coding to an app. Blockly provides a block editor UI and a framework for generating code in text-based languages. Out of the box it includes generators for JavaScript, Lua, PHP, Dart, and Python etc.

Custom blocks needed for our application can be generated using [blockly-developer-tools](#) and integrated into our app.

Current version of musicblocks has a feature to run midi device on it, blockly allows us to add blocks and categories dynamically as the scope of our app expands. If your app uses hardware (midi in this case), we can only show blocks for the currently attached hardware or let your user tell you which components they have.

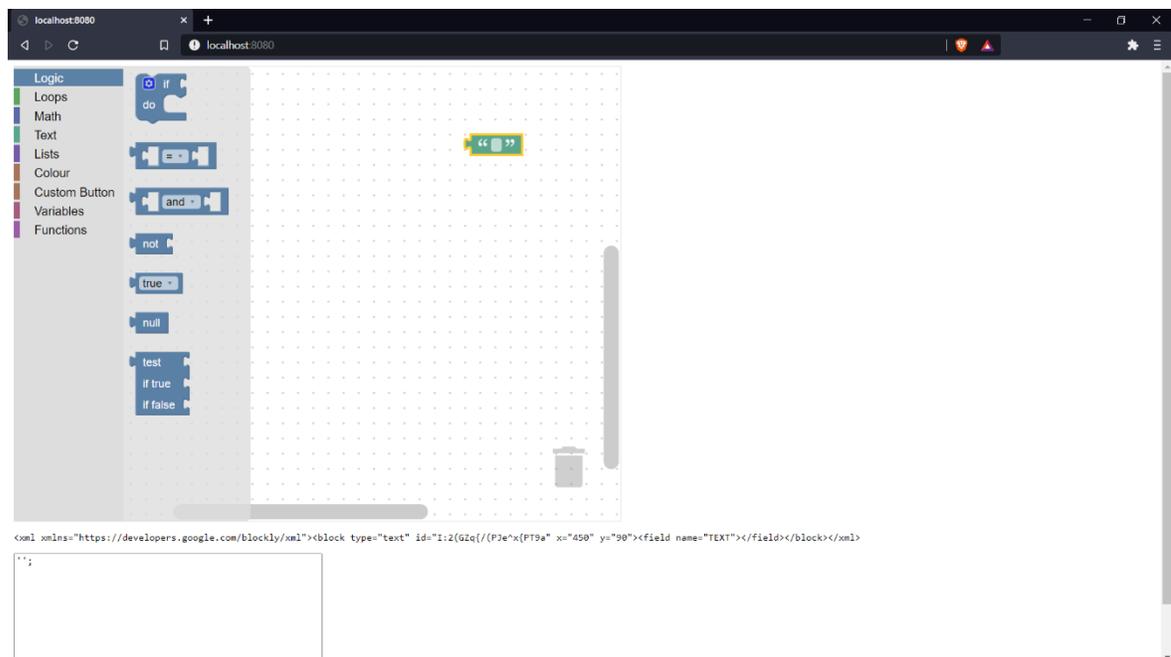
Users will have an easier time with our block language if there is consistency in our use of language and design patterns. I suggest that we:

- Use color to reinforce similarities between blocks.
- Use the same sentence structure across blocks.
- Use the same input order when different blocks use the same inputs.
- Use the same word everywhere when referring to the same thing.

Blockly is one of a growing number of visual programming environments. Deciding which one to use in your app is an important step, so here are a few of Blockly's biggest strengths to help you make the decision:

- **Exportable code.** Users can extract their block-based programs to common programming languages and smoothly transition to text-based programming.
- **Open source.** Everything about Blockly is open: you can fork it, hack it, and use it in your own sites and Android apps.
- **Extensible.** Tweak Blockly to fit your needs by adding custom blocks for your API or removing unneeded blocks and functionality.
- **Highly capable.** Blockly is not a toy. You can implement complex programming tasks like calculating standard deviation in a single block.
- **International.** Blockly has been translated to 40+ languages, including right-to-left versions for Arabic and Hebrew.

A react based blockly workspace sample is also available, upon which we can add our additional blocks. Once, the wireframe for the new musicblocks is decided, we can tweak up the workspace tantamount to it. As is, the workspace looks like this: -



The primary issue with current implementation of block generation in musicblocks using blockfactory and protoblock is the convoluted nature of it. By using blockly, that issue is obviated. I created a sample pitch block,



The block generator stub is reduced to just: -

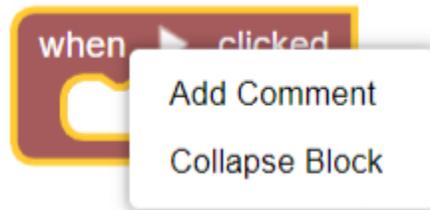
```
Blockly.JavaScript['pitch_block_demo'] = function(block) {
  var statements_pitch = Blockly.JavaScript.statementToCode(block, 'pitch');
  // TODO: Assemble JavaScript into code variable.
  var code = '...';
  // TODO: Change ORDER_NONE to the correct strength.
  return [code, Blockly.JavaScript.ORDER_NONE];
};
```

With corresponding json as: -

```
{
  "type": "pitch_block_demo",
  "message0": "pitch %1",
  "args0": [
    {
      "type": "input_statement",
      "name": "pitch",
      "check": "Number",
      "align": "RIGHT"
    }
  ],
  "inputsInline": true,
  "output": "String",
  "colour": 300,
  "tooltip": "",
  "helpUrl": ""
}
```

In addition to the aforementioned benefits of using blockly, I would also like to propose that we introduce comment feature on individual block stacks. When creating big projects with musicblocks like Fibonacci, the workspace gets cluttered and it's hard to keep a track of all the blocks, what do they do and when will they be triggered and so on. Also, after a user has saved their project work locally or even on planet, and if the project is visited after a long time, relying solely on

memory recollection as to which group of blocks does what and when they should be added and so on is quite cumbersome. Adding a feature to add comments on blocks will help us obviate this. This feature is already an integral part of all [blockly examples](#).



As creating a Visual Programming Language with music as an integral part of it is our goal, we have [blockly.games/music](#) as a source of inspiration to peruse. Most of musicblocks's features like play a note, rest, set instrument, note-values(c4,d4 etc) work completely fine here.

Research Sources: -

- [Block Design Discussion](#)
- <https://developers.google.com/blockly/publications/papers/TipsForCreatingABlockLanguage.pdf>
- <https://developers.google.com/blockly/publications/papers/TenThingsWeveLearnedFromBlockly.pdf>
- [Blockly Documentation for developers](#)

Timeline: -

Since the entire project is about the blocks graphics refactoring, the timeline should be viewed more in terms of Phases rather than weeks. Since there is an overhaul of the entire application, I intend to align my project with other GSOC projects of this year namely: Menus and Palettes, Blocks Reorganization and debugging aids. I also intend to contribute in every way I can whenever needed.

Pre GsoC: -

- During this time, I'll try to dive deeper into blockly library and start developing some sample blocks, get their UI approved by mentors.
- Once, the UI of blocks is approved by mentors, I'll start integrating it into react-blockly sample workspace as shared above.

Phase 1: -

- **Week 1-2(June 7, 2021- June 21, 2021)**
Start with an extensive discussion with mentors and the community, especially those with a musical background about what behavior is expected from the new version of music blocks and blocks graphics.
- **Week 3(June 21, 2021 – June 28, 2021)**
Come to an agreement with mentors on features, design, and UI of all the blocks.
- **Week 4(June 28, 2021 – July 5, 2021)**
Implement the finalized features and get a feedback on the same. If any changes, implement them and then start working on the remaining blocks.

Milestone Reached: Phase 1 Evaluation

Phase 2: -

- **Week 5-8(June 5, 2021 – August 2, 2021)**
Discussion and work on the rest of the blocks that remain, integrate features like commenting, collapse etc. Add necessary plugins as well. Add generation of corresponding JavaScript feature as well.
- **Week 9-10(August 2, 2021 – August 16, 2021)**
Cleanup and testing of all the blocks mostly.

Milestone Reached: Phase 2 Evaluation

Post GsoC: -

- Mostly Complete working on the [idea](#).
- Contribute to other akin projects-ideas for musicblocks and continue contributing to the new application's both core engine and UI.

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

Amidst the pandemic, all my classes are being conducted online with option to view recorded lectures. This allowance allows me to have a flexible time schedule throughout the day. There will be only 1 exam this semester from **17 May 2021–5 June 2021 (probably online as cases are on the rise in India)**. After exams, I have 2 months of vacations and hence, I'll be able to devote a lot of time to GSoC. I can easily devote up to 30-35 hours a week.

Other than this project, I have no other commitments or vacations planned for the summer. I shall keep my status posted to all the mentors and community members on a weekly basis and maintain transparency in this project.

If you will be off-the-grid for a few days, then mention those in the timeline.

Since I am in my junior year, I don't have any commitments during the period and so I will be available for almost all of the time frame. Furthermore, I do not have any planned vacations or other engagements.

How will you report progress between evaluations?

In between evaluations, I am reachable anytime through Email, Matrix, Slack or a well-planned video session if required.

Further, the nature of this project is as such that I'll be continuously working on Github issues, interacting with one or more mentors, starting discussions whenever needed. I'll let them know of the progress then and there.

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

I have learnt a lot and picked up most of my skills by contributing to sugarlabs's projects over the winter and, I plan on continuing my contributions to this organization, by adding to my past projects and working on open issues. With the community growing continuously, I feel responsible for all the projects I'm a part

of. Having picked up a lot of developing skills, my major focus would be to develop mentorship skills so that I can give back to this community by helping other people navigate around and reviewing their contributions.

Till now, the highlight of my experience with Sugar Labs has been the active involvement of the mentors. With the community growing continuously, I feel responsible for the projects I contribute to. Having picked up a lot of development skills, my major focus after GSoC would be to enhance my mentorship skills so that I can give back to this community by helping other people navigate around and hope to mentor future GSoC students.

Comments: - Irrespective of the proposal being selected or not, I'd like to get some suggestions on my proposal, and on how I can improve it. Thank you!

****Looking forward to contributing to Music Blocks this summer****