



# GSoC'21 Proposal - Processing

*p5-teach.js*

## Personal Details

Name: Aditya Siddheshwar

Course: B.Tech - Electronics and Communication Engineering

University: JK Institute of Applied Physics and Technology, Allahabad, India

Email: [adityasiddheshwar.adisid@gmail.com](mailto:adityasiddheshwar.adisid@gmail.com)

Github: <https://github.com/two-ticks>

LinkedIn: [www.linkedin.com/in/aditya-siddheshwar](https://www.linkedin.com/in/aditya-siddheshwar)

Discourse: <https://discourse.processing.org/u/two.ticks/>

Phone: +91 9511492788

Current Country: India

Time Zone: IST(UTC+5:30)

## Project Abstract

This project would involve developing tools for teaching STEM through p5.js, adding functions to animate shapes, and animating maths symbols. The main focus is to introduce a simple, easy-to-use library to animate repetitive methods. It would provide educators tools to make interactive animated sketches that can support learning in the remote environment. It will take advantage of the p5.js core library - p5.sound.js for sound effects and KaTeX for rendering symbols.

# Project Description

## Overview

- Scenes and transitions
- Transform functions
- TeX editor
- Create a player that can tell progress from 0% to 100% over time and control the scene
- `play()`, `pause()` and `stop()`
- Typing effects
- `fadeOut`, `fadeOut`, `blink`, `wipeIn`, `wipeOut`, `zoomIn`, `zoomOut`
- Sound/Audio support
- Camera controls (like [manim.scene.moving\\_camera\\_scene](#))
- Feedback from educators
- Unit tests (using mocha and chai)

## Detailed

### Why *p5-teach.js*?

#### Problems faced by the teachers of high school and professors of STEM:

- The main problem for STEM teachers is installation and setup.
- Teachers of high schools use tools such as Desmos and GeoGebra for visualization but can't be used for explaining derivations involving a lot of math symbols.
- Professors use MATLAB for visualizations and it is not easy to learn a new language considering their time constraints.
- They don't have access to proper documentation for animation engines for mathematics.
- They need transition and scenes but during installation and setup of animation engine (like manim), they get demotivated and get back to MATLAB
- The visualizations they are using are not pretty.
- STEM teachers need LaTeX for work but it is not flexible. For example, they want to label graphs with maths symbols but it is not easy to do so.
- All STEM teachers are not necessarily good coders. They don't want to use software that is very tough to learn, as it will take some time and a learning curve to get familiar with.
- It is not easy to tweak the library because of the complex coding concepts used in the library.

p5-teach.js will try to resolve these problems.

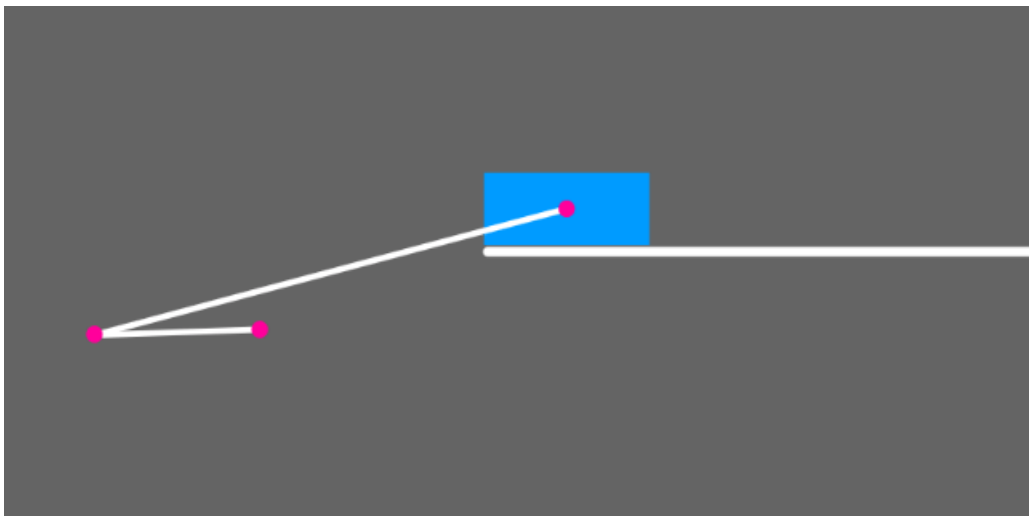
**Key-features:**

- **Easy to set up and ready-to-use examples**
- **Getting started guides and cheat sheets for beginners**
- **It would not be tough for beginners to learn**
- **It will be pretty and will support transitions**
- **Documentation and examples**

**p5-teach.js** will broaden the community by connecting with more people from the STEM field. Bringing people from STEM fields will be a step to follow our commitments towards adding features to p5.js that increase access. We focus on making p5.js available to and approachable for people who are excluded from the p5.js community (intentionally or not) and from similar tools and communities. The community will benefit from the involvement of more STEM teachers and can explore together more art, science, and technology.

The project can be divided into three stages:

- Development of animation methods, scene control, and tools
- Adding support for KaTeX and sound in animations
- Testing and Documentation



This is an example of animation ported from the manim (python) library into a p5.js sketch. In manim and other animation engines, it is easy to export high-quality video but they are very tedious to set up. Maths require symbols that we usually make from LaTeX tools which are not easy to integrate into animations. The power of p5.js lies in its beginner-friendly nature which will help to solve these issues. We can build a library to solve these problems. The primary aim is to provide listed tools for teachers who just started with p5.js and javascript

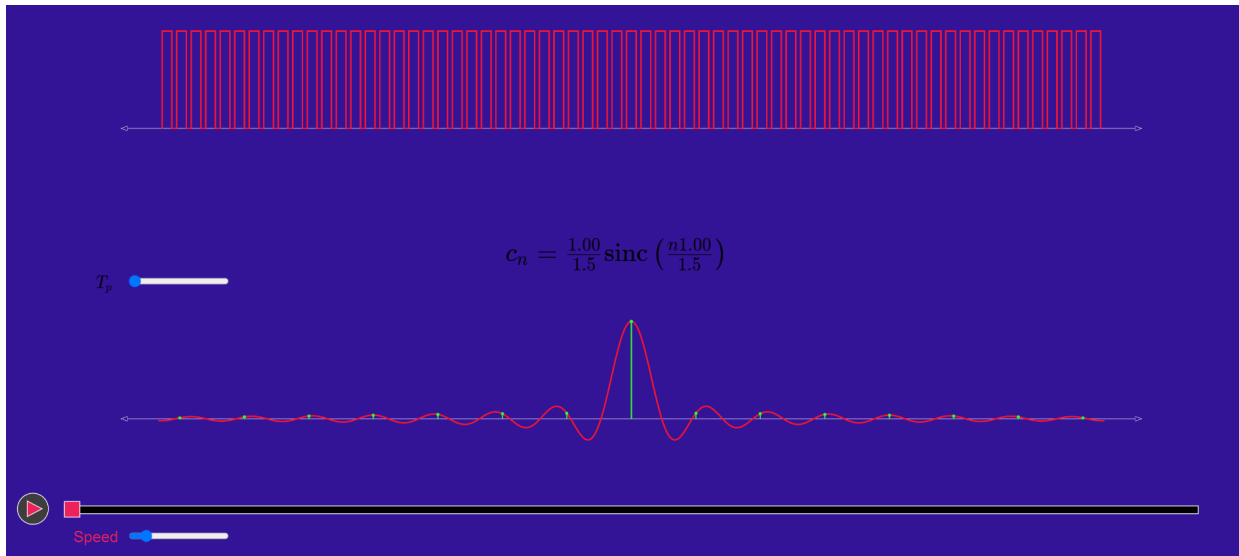
- Animation through browser
- TeX editor

## Animation through browser

This feature includes animation playing within the browser but also allowing users to interact while the animation is playing, such as a player that can tell progress from 0% to 100% over time and control the scene. We can add sound and speed controls to make the user experience more interactive. Educators can embed their sketches in webpages or can share links to their students to make the distance learning experience more interactive.

Such as

- [Visualizing quaternions - An explorable video series](#)
- [Ractive-Player](#)



This is an example of animation through the web browser.

[https://editor.p5js.org/radium.scientist/present/aOz6FS2j\\_](https://editor.p5js.org/radium.scientist/present/aOz6FS2j_)

## Typing Effects

$$F = G \frac{m_1 \cdot m_2}{r^2}$$

```

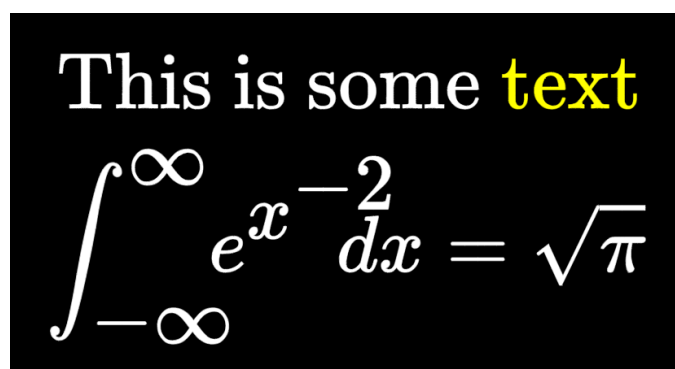
let currentIndex = 0;
let tex;
let katexScript = "\\Huge F , = , \\Huge G, \\frac{m_{1}\\cdot m_{2}}{r^{2}}, \\n";
let parts = katexScript.split(",");
let content = "";
function setup() {
  createCanvas(400, 400);
  frameRate(4);
  tex = createP();
  tex.position(width / 5, height / 3);
}
function draw() {
  background(50);
  fill(255);
  textSize(144);
  textAlign(CENTER, CENTER);
  content += parts[currentIndex];

  katex.render(content, tex.el);
  currentIndex++;

  if (currentIndex >= parts.length) {
    currentIndex = 0;
    content = '';
  }
}

```

Typing effects can be manually produced but are difficult to do for large sentences and will be monotonous. We can inspire from manim and can develop text animations such as the following:



This is some **text**

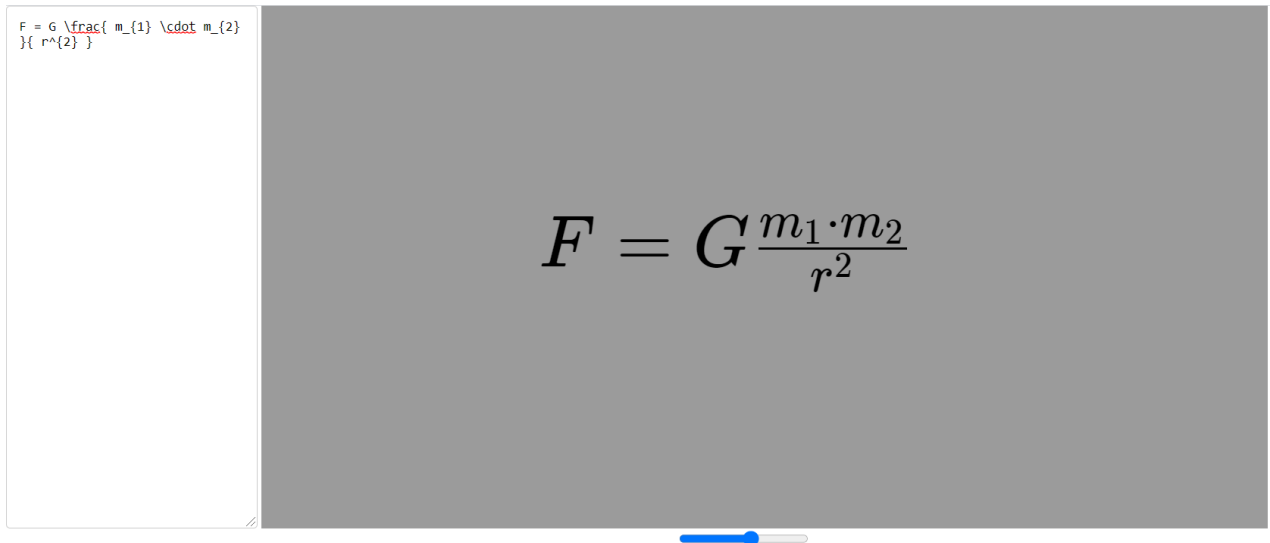
$$\int_{-\infty}^{\infty} e^{x-2} dx = \sqrt{\pi}$$

## TeX editor

### Why *TeX editor*?

TeX editor is required for rendering KaTeX to the canvas in creative ways so that educators (specifically those who post mathematical problems through the online medium in form of Google quiz, Instagram post, or Twitter) can screenshot (with inbuilt tools such as snipping tool in Windows 10 or open-source tools such as ShareX) and share them online. TeX editor can change the background, font color, load fonts, resize, drag and drop TeX. We can try to build features to export as SVG or PNG by using [canvas-latex](#) library that renders KaTeX to canvas and then we can use [toDataURL\(\)](#) methods to create an image to export SVG or PNG from TeX editor, but it can take a lot of time which we can invest on other features this summer.

<https://two-ticks.github.io/p5-teach.js/katex-editor/>





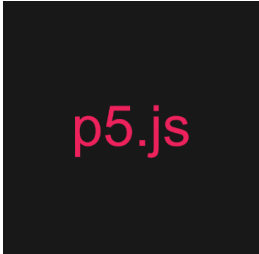
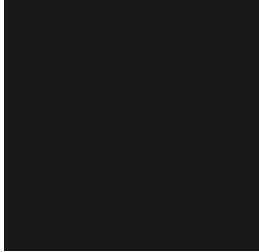
TeX editor is similar to [Maths in Motion](#).

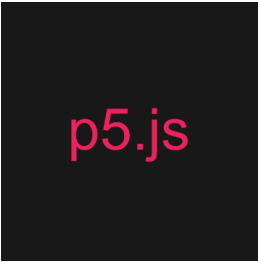

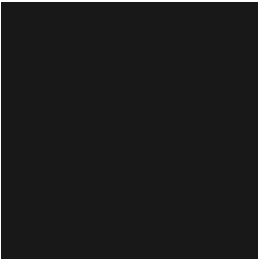

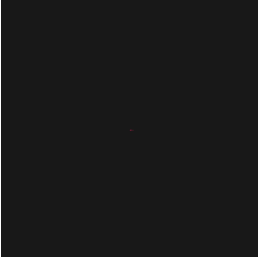
This project is more focused on educators using the online mediums to teach. Many educators post on Twitter, Instagram, and YouTube, TeX editor will help them build posts by providing them tools to work with KaTeX easily.

ShareX or OBS Studio can be used to record animations and capture screenshots for posting online (in the future we can add exporting features with help of [canvas-latex](#)). We can add examples for recording with the help of CCapture library (will not be able to record KaTeX elements because KaTeX is added as a DOM element but such libraries record only canvas).

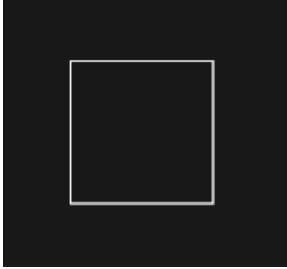
## API

(\*GIFs: <https://github.com/two-ticks/p5-teach.js/blob/gh-pages/README.md>)

write(object)	 <p>Writes the text with a blurry effect at each new character.</p>
typeWriter(object)	 <p>Write the text with the typing effect.</p>
fadeIn(object, duration)	 <p>Make a fade in effect.</p>
fadeOut(object, duration)	 <p>Make a fade out effect.</p>

<code>blink(object, duration)</code>	<div><p>Make a blinking effect.</p></div>
<code>wipeIn(object, duration)</code>	<div><p>Make a wipe in effect.</p></div>
<code>wipeOut(object, duration)</code>	<div><p>Make a wipe out effect.</p></div>
<code>zoomIn(object, duration)</code>	<div><p>Make a zoom in effect.</p></div>
<code>zoomOut(object, duration)</code>	<div><p>Make a zoom out effect.</p></div>
<code>moveAlongPath(object, path)</code>	Moves object along the defined path.



Transform(object1, object2)	Transforms one shape into another by interpolating vertices from one to another. Similar to <a href="#">this</a> example.
Shift(x,y)	Shifts shape and object by x and y
wait(duration)	Waits for the duration specified before playing the next animation. It is helpful in timing the animation.
Create(object)	 <p>Creates animation of object building up from scratch.</p>

### Additional Features (to be discussed)

- 3-D shapes and interactions using three.js or p5-WebGL
- Export as SVG or PNG from TeX editor

### Unit Tests

We will use chai for making our test cases and mocha for driving our test cases.

### Work done so far

- <https://github.com/two-ticks/p5-teach.js>
- <https://discourse.processing.org/t/discussion-and-review-of-proposal-addon-library-development-p5-teach-js/29065>

## Project Plan - Preliminary Plan

### May 17, 2021 - June 7, 2021 (Community Bonding Period)

- Understand the existing codebase of p5.js and p5.sound.js by working on existing minor issues and being active in the community.
- Discuss with the mentor the best way to go about the implementation
- Get familiar with animation and simulation JavaScript libraries such as scene.js, anime.js, and manim.js. Understand the inner working of projects such as canvas-latex.js, manim, and eulertour.

### Week 1

- Discuss with the mentor the best way to go about the implementation
- Finish implementation of typing effects

- Add documentation for typing effects.
- Create a player that can tell progress from 0% to 100% over time and control the scene (progress bar)
- Add automated test cases

## Week 2

- Add graphs and plotting
- Add animation for shapes and geometries.
- Documentation and examples of graphs and plotting.
- Documentation and examples for animation of shapes and geometries.
- Add automated test cases

## Week 3

- Finish implementing play(), pause(), and stop()
- Finish implementing scenes and transitions
- Add examples from high school topics and test functionalities :
  - Relative velocity
  - Inertia
  - Trigonometric Ratios
- Add automated test cases

## Week 4-5

- Start working on KaTeX support and sound.
- Improve the UI of the TeX editor and add buttons.
- **Animating the TeX element** - This milestone is one of the most difficult parts of the implementation, this may require more than two weeks. This will include discussion with mentors on various libraries to use and ways to animate TeX elements. The main goal is to animate derivations. If we would be successful in implementing this part we can look forward to document and add examples.
- Add automated test cases

## Week 6-7

- Implement fadeIn, fadeOut, blink, wipeIn, wipeOut, zoomIn, zoomOut, flip, flipX and flipY
- Add examples from University topics and test functionalities (may take more time) :
  - Sampling theorem
  - Thermodynamics
  - Nyquist Plot
- Take feedback from teachers
- Documentation and examples
- Add automated test cases

## Week 8-9

- Taking feedback from teachers and testing
- Documentation and examples
- The last three weeks are dedicated to documentation, testing, and feedback. In this period I will test and improve the library so it is easier to access and easy to use. I will take feedback from mentors, teachers from high school and professors, and will add more changes to the

library for better usage across diverse topics and fields.

## Week 10

- Documentation and examples
- Build tutorials
- Finalizing and presenting a demo to teachers and professors.

## Major Milestones

- Development of animation methods and scene control
- Adding support for KaTeX and sound in animations
- Demo to educators

## Commitment

My 5th semester will complete in mid-April leaving me enough time to get ready for my GSoC project. If I am selected, I shall be able to work around 35-40 hrs per week on the project, though am open to putting in more effort if required. My 6th end-semester exams are likely to be in July, I assure the organization I will try to put 20-30 hrs per week in that duration and will cover in subsequent weeks.

## Contributions so far

- PRs merged  
**p5.js-website**
  - [added symbols in description of math examples](#)
  - [added graphing 2d equations example](#)
  - [added statements and comments example](#)
- Issues, bugs found  
**p5.js**
  - <https://github.com/processing/p5.js/issues/5169>**CircuitVerse**
  - <https://github.com/CircuitVerse/Interactive-Book/issues/540>
  - <https://github.com/CircuitVerse/CircuitVerse/issues/2188>
- Other contributions  
**p5.js**
  - <https://github.com/processing/p5.js/issues/5070>
  - <https://github.com/processing/p5.js-web-editor/issues/1820>
  - <https://github.com/processing/p5.js-web-editor/issues/1838>

## My Projects based on **p5.js**

- **p5-projects** (<https://github.com/two-ticks/p5js-projects>)
- **line-encoding** (<https://github.com/two-ticks/line-encoding>)

## Additional Information

I have an Instagram page [two.ticks](#) where I post animations and sketches (mostly related to science). During lock-down I explored a lot of pages on Instagram about p5.js and animations, then started making animations myself. I am very interested in animating maths and have used manim to make animations such as [this](#), though the rendered quality was high in manim it lacked interaction (such as [this](#) in p5.js). This project is inspired by [eulerv2](#), [Dynamic Learning](#), [manim.js](#), and [MiM](#).

## After GSoC

I am interested in simulations and animations, therefore, will keep exploring and learning more about other open-source libraries such as matter.js, CindyJS, and manim.js. I will learn more about Verilog and contribute to [CircuitVerse](#) and [riscv](#). I will continue contributing to the Processing Foundation and work on building libraries for animation and education. I will add more features and examples to p5-teach.js.

I will also work on adding these features in [p5-teach.js](#):

- Export output (as GIF or MP4) feature (related issue: [#5118](#)) in the future after GSoC.
- TeX editor - Export as SVG or PNG. [canvas-latex](#) library renders KaTeX to canvas ( then we can use `toDataURL()` to create an image) to export SVG or PNG from TeX editor