Support Shader-Based Filters in p5.js

2023 Google Summer of Code Proposal

Justin Wong
https://wonger.dev
jkwongfl@yahoo.com
github.com/wong-justin

Project task: Support shader-based filters in p5.js (link)
Project size: 350 hrs
Potential mentors: @davepagurek, @aceslowman, @aferriss

Context

What's the need for shader-based filters? The main motivation is that image filters in p5.js are too slow with their current pixel-by-pixel loop implementation. It frustrates students and other users who have limited computer resources when some sketches run at a few frames per second. An alternate implementation of image filters using shaders will significantly improve performance. (See 3)

Moreover, implementing shader filters will touch on accessibility in a few other ways:

- The filter() function will be expanded to include a shader parameter instead of only accepting image filter flags. This gives users another entry point to shader programming in their sketches. (See 2)

- It's a good opportunity to improve documentation. I found that the current docs are a bit inconsistent and opaque for users to understand the role of WebGL. Shader-based filters will add more complexity to this space. Rounding out documentation about WebGL mode and its methods will give users some clarity and help them transition from the default 2D drawing mode into more advanced features. (See 1)

- Performance is not the priority for p5.js, but bad performance ruins accessibility if the sketches can hardly run. Users and contributors would benefit if there was a system to benchmark other user-facing methods and target the slowest ones. Profiling these new shader-based filters is a good opportunity to start such a system, or update an existing one. (See 4)

Me

Why am I personally excited for this project? It covers some of my primary interests and goals:

- I love p5js. It was my introduction to javascript and the library I used to make a venn diagram highlighting tool - and that tool happens to share a performance issue related to the slow image filters covered in this project. I'm going to investigate a rewrite of the venn diagram with shaders after GSOC because of this proposal. I've also enjoyed dabbling in shader programming, between working through the book of shaders and tinkering in the minimalist playground tixy.land.

- I love teaching. I worked as a math tutor for several years since high school and through the pandemic (see the whiteboard I made for remote tutoring, or at least the demo gif). Part of tutoring involves good communication and simple explanations. GSOC is a great opportunity to hone these skills, especially on a library like p5js designed in part for learners.

- I want to be a better developer. I've been making solo projects and reading about software development since finishing my two years of programming courses, free to explore any topic and do as I please. But I have no experience with the collaborative and real-world aspects of development. The prep work for this proposal has already taught me a lot about navigating a large codebase and what it takes to work with other people on software. GSOC will be a great launching point for a career in software. I'm working part-time in a different field to support myself for now, so I'll be contributing half days throughout the week for the duration of the project.

The Plan

0. Become fully onboarded

I've gotten familiar with the project and codebase from a distance, but I haven't executed much code.

Timeframe:

April 4 - May 3: Continue independent prep; start development environment, fork repos, and sketch proof of concepts. Explore past PRs and commits on related topics like how the 8 default image filters were chosen, and the commit history of documentation pages for drawing functions. Explore tests. Understand framebuffers and textures more fully.

May 4 - May 28: Community bonding; looking forward to communicating closely with a mentor and closing my knowledge gaps. Finishing setting up my development environment.

1. Start improving docs

Adjusting docs seems like a gentle start into the codebase. I want to be able to explain p5js WebGL features to myself and others before coding a new one.

Currently:

1. The reference pages for shared 2D/3D drawing methods are a bit inconsistent about documenting their 3D parts.

   Some don't mention 3D at all, like circle();
   Some only mention it in a parameter/function signature, like point();
   Some mention it in the description, like quadraticVertex();
   And some mention it everywhere, with examples, like vertex().

The inline 2D docs seem to override the competing inline 3D docs. For example, when comparing point() in 3D to point() in 2D, only the 2D content is found in the reference page.

2. Is P2D vs WEBGL mode clear? I think beginners assume WEBGL just means 3D shapes, which misses some nuance like shaders and textures and 2D drawing and text rendering and the coordinate system. The WEBGL doc page is bare even though it's the first search result for "p5js webgl".

   There's a lot of good content about WebGL mode, but it's scattered around: A great tutorial in the p5.js github wiki by Stalgia Grigg (link), a series of articles from Austin Slominksi in the p5js.org "Learn" section (link), a booklet of p5js shaders (link), and a bit more across the internet.

Proposed:
- Complete documentation by adding WebGL parameters, descriptions, and code examples where appropriate for shared P2D/WEBGL drawing functions.
- Link to existing WebGL resources in the WEBGL mode reference page, along with a helpful summary of differences / features to expect.

Timeframe:
May 29 - June 11 (2 weeks)

2. Implement new parameter filter(shader)

Currently: filter() only takes FILTER_TYPE parameters, not shaders. This is incomplete compared to desktop Processing's filter() function, which works with both types of calls. The missing part of the filter() function is waiting at RendererGL.js.

Proposed:
- Implement filter() function for the WebGL renderer
    - Accept a p5.Shader that's been loaded earlier in the setup
- Consider exposing default shader constants, like a basic.vert that just passes on gl_Position.
- Add tests.

Timeframe:
June 12 - July 2 (3 weeks)

3. Replace implementation of filter(FILTER_TYPE) with shaders

Currently: The filter() function in the P2D renderer uses pixel-by-pixel loops (source), which is why it's so slow. There's a proof of concept sketch that demonstrates a way to replace this and integrate WebGL shader filters, which I'll use as a starting point.

Shaders for other WebGL functions are located in p5.js/src/webgl/. Existing shader filters are located in Adam Ferriss' repo.

Proposed:
- Make the filter() function in P2D mode use a secondary WEBGL renderer behind the scenes.
    - Investigate the best way to create this secondary renderer (eg. at the beginning of every sketch? Not every frame). Include performance tests.
    - Investigate framebuffers.
- Create OPAQUE, POSTERIZE, ERODE, DILATE shaders, and use from existing GRAY, INVERT, THRESHOLD, BLUR shaders.
    - Investigate if BLUR, DILATE, ERODE even need to be replaced (they're 20x - 100x faster than other filters, according to 1.3.1 benchmarks).
    - Consider adding a few new filters, like INVERT_OPACITY.
- Let the filter() function in WEBGL mode accept these filter constants
- Add tests.

Timeframe:
July 3 - July 30 (4 weeks)

Note: July 10 - 14 is midterm evaluation

4. Performance benchmarks

Currently: It would be nice to give other contributors some performance measuring tool, but it's a low priority (the slowest drawing functions in p5js are usually identified from user feedback). This is more of a stretch goal as time allows. The only system in place is Kenneth Lim's inactive benchmark repo. There's also a one comparison of javascript vs WebGL in this glitch.com demo.

Proposed:
- Investigate the existing benchmark repo and figure out how to apply it to new code or p5js versions. Consider how it could be incorporated into CI / build process.
- Investigate a system using other profiling tools like jsperf or chrome dev tools in the meantime. Consider how it could be generalized or deployed for others to use.
- Search for user sketches in the wild, or curate my own, to use as before/after demos to showcase performance improvements from this project.

Timeframe:
July 31 - Aug 20 (3 weeks)

Note: Aug 21 - 28 is project submission week.
And from Aug 29 onward: Post-project writeup (blog post?) Contribute to p5js again in the future!