Geometrize Haxe Web Documentation

Comprehensive Project Guide and Reference

1. Project Overview and Goals

Geometrize Haxe Web is an open-source web application for recreating photos using simple geometric shapes. It runs entirely in the browser, using the Geometrize Haxe library (a Haxe port of Michael Fogleman's primitive algorithm) to transform raster images into a collection of optimized shapes. Given a target image, the app repeatedly generates random candidate shapes and uses a hill-climbing optimization to pick and add the best-fitting ones, gradually approximating the original picture. The result is an abstract, stylized output composed of semi-transparent rectangles, triangles, circles, ellipses, lines, and more, which can be exported or further edited as vector or raster artwork. As Sam Twidale (the project author) notes, this demo "uses the Geometrize Haxe library to transform images into shapes", and it faithfully reproduces Sam's desktop Geometrize algorithm in a portable web format.

2. Real-Life Use Cases

1. Generative Art & Design: Artists and designers can use Geometrize Haxe Web to create stylized posters, logos, and illustrations by converting photos into geometric abstractions. The ability to select shape types and parameters makes it a creative tool for digital art and graphic design experimentation.

2. Creative Coding & Visualization: Educators and students in computer graphics or algorithm courses can explore how simple primitives and optimization build complex images. The live, step-by-step generation makes it a great demonstration of computational art and hill-climbing techniques.

3. Data-Driven Graphics: Developers can incorporate the exported SVG or JSON shape data into other apps - for example, animating the shape data or integrating it into generative visuals. (The Geometrize project even includes a 'shape data tweening' demo to animate between geometry scenes.)

4. Education & Outreach: By rendering familiar photos (e.g., of animals, landmarks, or people) in geometric form, this tool can introduce geometry concepts and computational art to students. It also serves as a fun entertainment or outreach demo for museums or science exhibits.

5. Social/Online Tools: The Geometrize technology is used in automated services like the Geometrize Twitter bot, which takes user-submitted images and geometrizes them into shareable shapes. This shows how the web app could underpin interactive media applications.

3. Detailed Workflow

1. Load an Image: Click 'Open Image' to upload your own photo or 'Random Image' to try a sample. The target image appears on the canvas as the background.

2. Adjust Settings (Optional): Open the Settings panel to customize parameters. For example, use checkboxes to include/exclude specific shape types (triangles, ellipses, rotated rectangles, etc.) and sliders to set Shape Opacity, Background Opacity, Shapes per Step, and Mutations per Shape. These control the speed vs. quality trade-off.

3. Start Geometrizing: Click 'Run/Pause' to begin the automated process. The app launches a Web Worker that repeatedly tries random shapes to match the image. Each iteration adds one optimized shape. You can also click 'Step' to add exactly one shape at a time (useful for manual pacing). While running, the canvas is updated in real time with each added primitive.

4. Monitor & Refine: As shapes accumulate, the approximation improves. The Shapes Added counter shows how many shapes are in the output. You can Pause at any time to tweak settings or compare results. To restart, hit 'Reset' to clear all shapes and begin anew.

5. Export Results: When satisfied, use the export buttons. The 'Save Image' button saves a raster PNG snapshot of the current canvas. 'Save SVG' exports an infinitely scalable vector image (combining all shapes as one SVG). 'Save JSON' downloads the raw shape data as a JSON file. These outputs allow you to further process or share the geometrized artwork.

4. Technical Architecture

Under the hood, Geometrize Haxe Web is built with Haxe and HTML5 technologies. The core algorithm is written in Haxe and compiled to JavaScript for the browser. The app uses an HTML <canvas> for drawing; a utility class (CanvasTools) handles drawing shapes onto the canvas. To avoid freezing the UI, the computationally intensive shape-search runs in a browser Web Worker thread. When you click Run, the main UI thread sends commands to a worker (instantiated from GeometrizeWorker) which performs the random shape mutations and evaluation. Results (best shapes, progress updates) are passed back via message events, and the main script renders them on the canvas.

In summary, all processing is client-side: the Haxe code is transpiled to JS, the image analysis happens in-browser, and rendering uses standard Web Canvas. No server is needed. The project's Haxe build configuration (.hxml) targets JavaScript, and the final /js folder contains the compiled output. This architecture means Geometrize Haxe Web works in any modern browser, and the entire app can be hosted as static files (HTML/JS/CSS). Key components include Main.hx (glues the UI to the worker), GeometrizeWorker.hx (defines the worker's algorithm loop), and message/interfaces files (GeometrizeWorkerInterface.hx, GeometrizeWorkerMessages.hx). The system leverages the Geometrize Haxe library for shape definitions, mutation logic, and exporting routines.

5. Benefits

- High Performance (Client-Side): Because the app runs in the browser with computation in a Web

Worker, the interface stays responsive even while thousands of shapes are evaluated. Simple optimizations and low-level drawing make it relatively fast on modern hardware, and work scales with image size.

- Portability: The tool is purely client-side and built in Haxe (which compiles to JS), so it runs on any platform with a compatible browser. The Haxe library also supports all targets, meaning the same codebase can be compiled for desktop, mobile, or other environments.

- Customization & Extensibility: Since it's open-source (MIT-licensed), developers can modify shape types, optimization strategies, or UI features. The Settings panel exposes many parameters to fine-tune the output. New features (like additional primitives or color modes) can be added by extending the Haxe code.

- No Server Dependencies: Deployment is simple - just serve static files. There is no backend or database, so user images stay local to the browser. This also means offline or self-hosted usage is straightforward.

- Scalable Outputs: The ability to export to vector (SVG) allows creation of high-resolution graphics. As noted in the Haxe documentation, "there is no jagginess when saving images as vector-based SVG". This makes the tool useful for producing print-quality art or zoomable graphics.

6. Export Options and Formats

Geometrize Haxe Web provides three export formats, each with its use cases:

- PNG (Raster Image): The 'Save Image' button generates a standard PNG. Use this when you need a simple bitmap output - e.g. for sharing on social media, including in slides or web pages, or quick previews. The PNG captures exactly what's shown on the canvas at that moment.

- SVG (Vector): The 'Save SVG' output produces a scalable vector graphic containing all the primitives. SVG is ideal when you need resolution-independence (for poster-size prints or responsive web graphics) or when you want to edit the shapes further in a vector editor. Because shapes are drawn with opacity, the SVG preserves the exact look of overlapping translucent shapes. (As documented, saved SVGs have 'no jagginess' and maintain quality.)

- JSON (Shape Data): Selecting 'Save JSON' downloads a JSON file listing all shape parameters (type, coordinates, color, etc.). This is not directly viewable as an image, but it preserves the generative data. You can reload this JSON in another Geometrize session, or use it for animations, data analysis, or programmatic remixing. For example, Geometrize includes demos that tween between JSON shape sets. JSON is useful for developers who want to manipulate the raw output.

7. Deployment Recommendations

Because Geometrize Haxe Web is a static web app, it can be hosted on any static-file server. The simplest deployment is via GitHub Pages or any CDN/static host. The official demo is live on Sam Twidale's site (https://www.samcodes.co.uk), and the source repository can be cloned to serve the files elsewhere. No special server-side technology is needed. Developers can build their own version by installing Haxe and running the included HXML build file (GeometrizeHaxeWeb.hxml), then deploying the result.

8. Code Examples

```
var targetBitmap = new Bitmap(imageData);
var backgroundColor = new Rgba(255, 255, 255, 1.0);
var runner = new ImageRunner(targetBitmap, backgroundColor);
var options = new ImageRunnerOptions();
options.shapeTypes = [Circle, Triangle]; // only use circles and triangles
// Add one shape
runner.step(options);
// Export results
var svgString = SvgExporter.export(runner.shapes);
ShapeJsonExporter.exportToFile(runner.shapes, "output_shapes.json");
```

9. Screenshots (Placeholders)

Figure 1: Geometrize Haxe Web output of a seagull photo using only 50 random circles. Each circle has partial opacity and a color chosen to best match the underlying pixel data. The result is very abstract - the bird's form is only roughly suggested by overlapping blobs.

Figure 2: The same image approximated with 500 circles, showing much finer detail. With more shapes, the algorithm captures colors and features more accurately (the beak and eye of the seagull become clearly defined). As expected, increasing the shape count yields a more recognizable and higher-fidelity result.

Figure 1 and 2: The Geometrize Haxe Web demo progressively builds up a picture by adding shapes. (Higher shape counts yield more refined images.) [Replace with actual screenshots when available.]

10. Licensing & Acknowledgments

Geometrize Haxe Web is distributed under the MIT License. The core library (Geometrize Haxe) is credited to Sam Twidale (samcodes) and was released in 2017; it is based on Michael Fogleman's Go primitive library. This demo and all its source code are open-source and freely available, as reflected by the MIT licensing text. We acknowledge Sam Twidale for developing and maintaining the Geometrize tools, and Michael Fogleman for the underlying algorithm. The example images used in the demo are public-domain/CC0 stock photos to avoid any copyright issues. We also thank contributors and community members who have helped refine the code and documentation.

11. References

- Project documentation and source repositories
- Geometrize Haxe library page
- Sam Twidale's Geometrize resources
- Geometrize Haxe README
- MIT License text