# Dan's SVG Plotting Library

Convert curves from SVGs directly into physical, straight‑line plotting instructions and completely skip the idea of manual curve flattening!

{Line}

## Why keep fiddling with Inkscape?

If you use Inkscape or other popular SVG editors to prepare your output for plotters and physical work, you will already be familiar with resorting to multi‑step workflows, involving at least the following steps.

* Create a sacrificial copy of your file that will no longer be editable in the future.
* Convert everything to objects.
* Subdivide all curves until you reach a desirable interpolation resolution.
* Convert all curves to line segments.
* Export to DXF (or another format).

That’s time-consuming, a lot of extra effort, and potential for error, not to mention that if you make any updates or improvements to the original drawing and want to print again, you have to repeat the entire process.

{Line}

## The SvgPlotting Wheelhouse

With the SvgPlotting library, you can get the finished physical plotting data with a single call, and there is no need to create a sacrificial or un-editable version of your file. Just follow these easy steps.

* Load an SVG file as an **HtmlDocument**.
* Create an instance of **SvgImageItem** with a reference to the HtmlDocument and a preferred curve vertex count for precision.
* ...
* nothing else. That's it. Use the *svg*.**PlotPoints** collection to plot your points out in the physical world.

When you initialize the SvgImageItem above, the library generates a sequence of straight-line segments, with embedded pen-up and pen-down instructions, ready for plotting.

Curves? Handled. Precision? You decide via the **curveVertexCount** parameter when initializing the object.

{Line}

## Quickstart

Following is a quick start example of using the SvgPlotter library.

{CodeBlock,cs,Example1.cs}

From there, feed the paths into your plotter or convert to simple DXF, G‑code, HPGL, or whatever you need.

{Line}

## Features

Here are some of the main benefits of choosing the SvgPlotting library.

* **One-step flattening**. No more time-consuming, messy manual workflows.
* **Configurable precision**. You control how many segments are used to approximate each curve.
* **Complete plotting data**. Straight segments and pen state included.
* **Generic output**. This library serves as an ideal precursor to DXF, G‑code, pen‑plot instructions.

{Line}

## When to Use SvgPlotting

Here are only some of the scenarios in which the SvgPlotting library can come in handy.

* Prepping SVG art for pen plotters, CNC, laser cutters, vinyl cutters, and embroidery.
* Automating workflows that currently rely on GUI tools like Inkscape.
* Generating minimalistic plot‑ready SVGs without the GUI overhead.

{Line}

## Integration

This library is not only tiny but a lot of its value rests in that there are no commercial dependencies aside from Newtonsoft JSON.

* .NET Standard library. You can use this and its supporting libraries directly in Unity or other systems compatible with .NET Standard. Virtually slap it into any C#/.NET project.
* No OS or GUI dependencies. This library works either headless or server-side on any platform where .NET will run.
* Directly consume *svg*.**PlotPoints** to generate your own output format.

{Line}

## Tips

Following are a few tips to help you get started.

* All output is currently in **mm**. If you would like the option to select any unit of measurement, please create an issue in this repository.
* Higher **curveVertexCount** creates smoother curves but at the cost of more segments.
* Sample values of 8 to 32 are good starting points for ultra-low poly, while 50 approximates a clean curve in a cm scale. Step into 64 or more only when precision is critical or the size of a curve is larger than 10 cm.
* Post-process the **PlotPoints** collection as needed by merging paths, offsetting lines, and exporting to your desired format.

{Line}

## Other Notes

This is an early release and there are currently a few limitations.

* The only containers currently supported are **A** and **G**. Please create an issue if you would like support for ClipPath, Defs, ForeignObject, LinearGradient, Pattern, RadialGradient, Switch, or Text.
* The only shapes currently supported are **Circle**, **Ellipse**, **Line**, **Path**, **Polygon**, **Polyline**, and **Rect**. Please create an issue if you would like support for Image, TSpan, or Use.
* 3D transforms are not yet supported.
* Text plotting is not yet supported.