USER GUIDE

- How to use and best practices

Version 1.0.1, last updated 25.09.2015

Made by
NordicEdu Ltd.

Table of content

Overview

Simply SVG supports a subset of the SVG 1.1 standard. It supports most relevant features needed for importing crisp flat shaded vector graphics to Unity.

Features

- Drag'n'Drop
- Automatic reimport after modifying the SVG file
- Basic shapes and compound paths with holes
- Simple single color fill and stroke with opacity
- Transforms
- Groups and layers
- Custom pivot points
- Quality level settings
- Collider support
- Unity 5.2 UI Ready
- Particle System support
- Render sorter

Unsupported features

Some SVG 1.1 features are unsupported at the moment. In many cases the same effect can be achieved through a workaround. This section lists some of the SVG features that may appear in your imported documents that are not supported or are unreliable. Tips on how to get around these limitations are given where possible.

CSS styling and color names

The style-attribute, style element or linked external CSS are not supported. Use "Presentation Attributes" as defined in the SVG standard instead. Check your SVG exporter settings.

Use hex values for color instead of the CSS color names. Illustrator does this by default, while other products may have it as an option in their SVG exporter.

Text

Font rendering support is not included. If you need text, collapse your fonts into outline shapes.

Illustrator does this for you automatically, if the appropriate exporter setting has been selected. See the "Exporter settings" section for more details.

Blending modes

Blending modes are not supported at this time by the importer nor the default Simply SVG shader. Only effects that produce flat colors can be achieved with flattening in Illustrator and/or opacity.

You may have success by creating a custom material in Unity.

Animations

No workaround exists. Animations should be done in Unity.

Fill effects (gradients, pattern textures etc.)

Use a custom material in Unity.

Stroke effects

Basic strokes are supported but anything more fancy is not. If you wish to use such effects, convert the stroke effect to a basic shape object.

Instanced shapes

Simply SVG may import the instanced shape, but won't display it multiple times. Also the location of the imported instanced shape will be different. Do all duplications and positioning in Unity.

For example the Symbol Spray tool in Illustrator produces instanced shapes.

Clipping masks

Bake clipped shapes into basic shape objects.

Content creation

Simply SVG is intended for use with Adobe Illustrator and Inkscape. Other graphics packages may work as well, provided their SVG exporters support all the relevant features.

This section covers how to get compatible SVGs out of Adobe Illustrator and Inkscape. This includes tips on how to get best results and how to avoid some known problem cases.

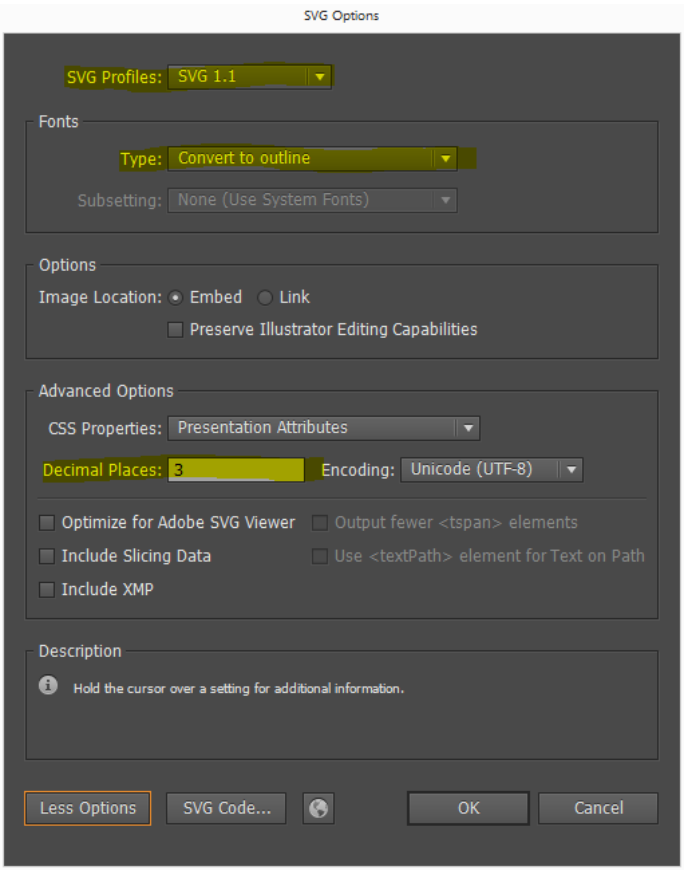
For most exporters, the most important settings are the following:

- Use Presentation Attributes for shape
    - Do not use CSS style attributes, elements or linked external CSS stylesheets.
- Use hex values for colors
    - Do not use CSS color names
- Use at most three decimal places worth of numerical precision
- If your graphics include text, convert them to outline shapes

Exporting SVGs from Adobe Illustrator

Settings for the built-in SVG exporter

Set Illustrator's SVG export settings as shown in the picture below.



- SVG profile must be set to “SVG 1.1” as this is the standard that SimplySVG follows.
- For text to work, the font type setting has to be set to “Convert to outline.”
- Numerical precision should be set to the maximum of three decimal places.
- Make sure CSS properties is set to “Presentation Attributes”

Multi-SVG exporter script for Illustrator

The multi-SVG exporter is a utility script for Adobe Illustrator that is included with Simply SVG. It’s purpose is to make it easy to split an Illustrator document into several SVG documents. This is useful for example when an artist wants to keep all game graphics in a single Illustrator document for easy editing and ensuring style coherence, but they should be appear in the game as separate entities.

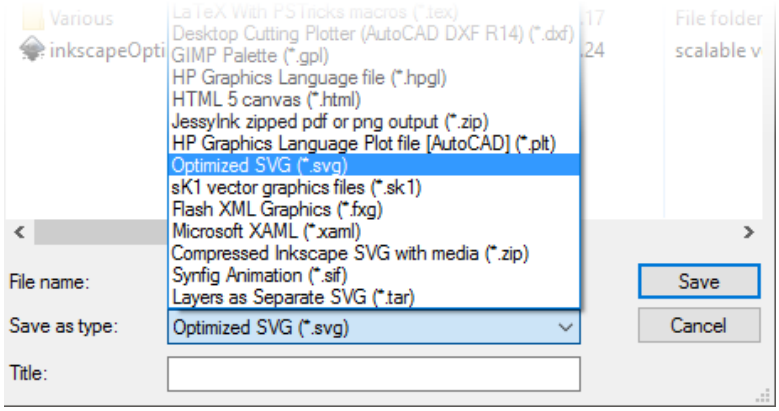
To use it, you must first structure the Illustrator document so that each separate graphic is in their own layer. Each layer will be exported as a separate SVG file.

The exporter script is included in the package downloaded from Asset Store. You should find it in the “SimplySVG/External Tools/Illustrator Plugin” directory.

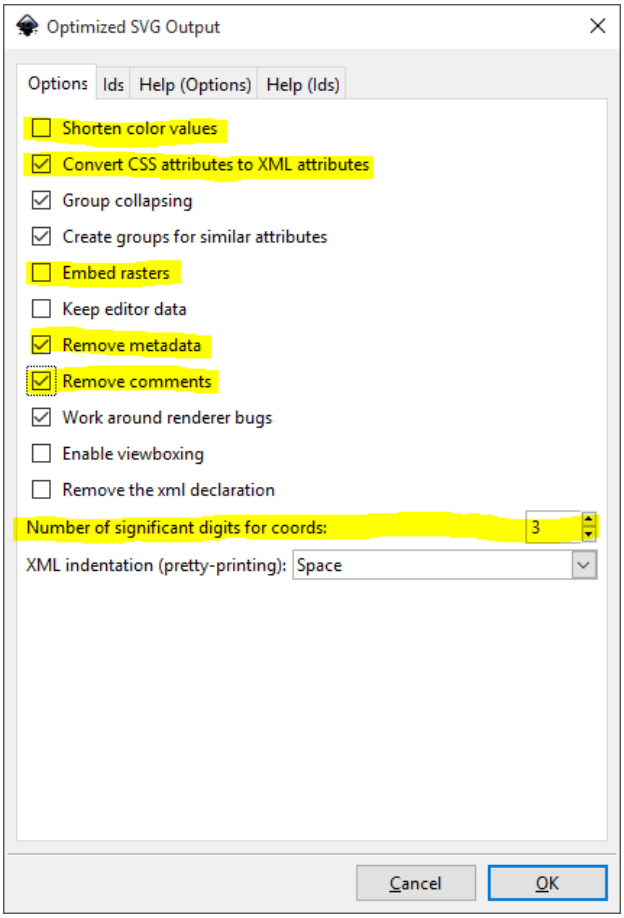
If you are unfamiliar with using custom scripts with Illustrator, see the included ReadMe file for more detailed instructions on how to use it.

Exporting SVGs from Inkscape

Inkscape includes many different SVG exporters with varying capabilities and settings. To export compatible SVGs from Inkscape, you have to use the “Optimized SVG” exporter. To do this select File -> Save as… and set the filetype to “Optimized SVG”. Do not use the other SVG options as they do not offer the needed export settings.



Set the exporter settings as shown in the following picture. The most relevant options are “Shorten color values” (off), “Convert CSS attributes to XML attributes” (on), “Number of significant digits for coords” (3). The other highlighted options should not affect importing, but help keep the file size to a minimum.



Best practices

Avoid excess control points and nonsensical shapes

Clean up your shapes before exporting. The importer honors control points and will not remove them unless they are in a straight line.

Some of Illustrator's tools produce a high amount of control points. There are several reasons these may appear while you work. For example Pathfinder and Shapebuilder sometimes produce a high amount of control points in a small area to better match the clipping curve's shape. Be careful when you use these tools and check the results before exporting.

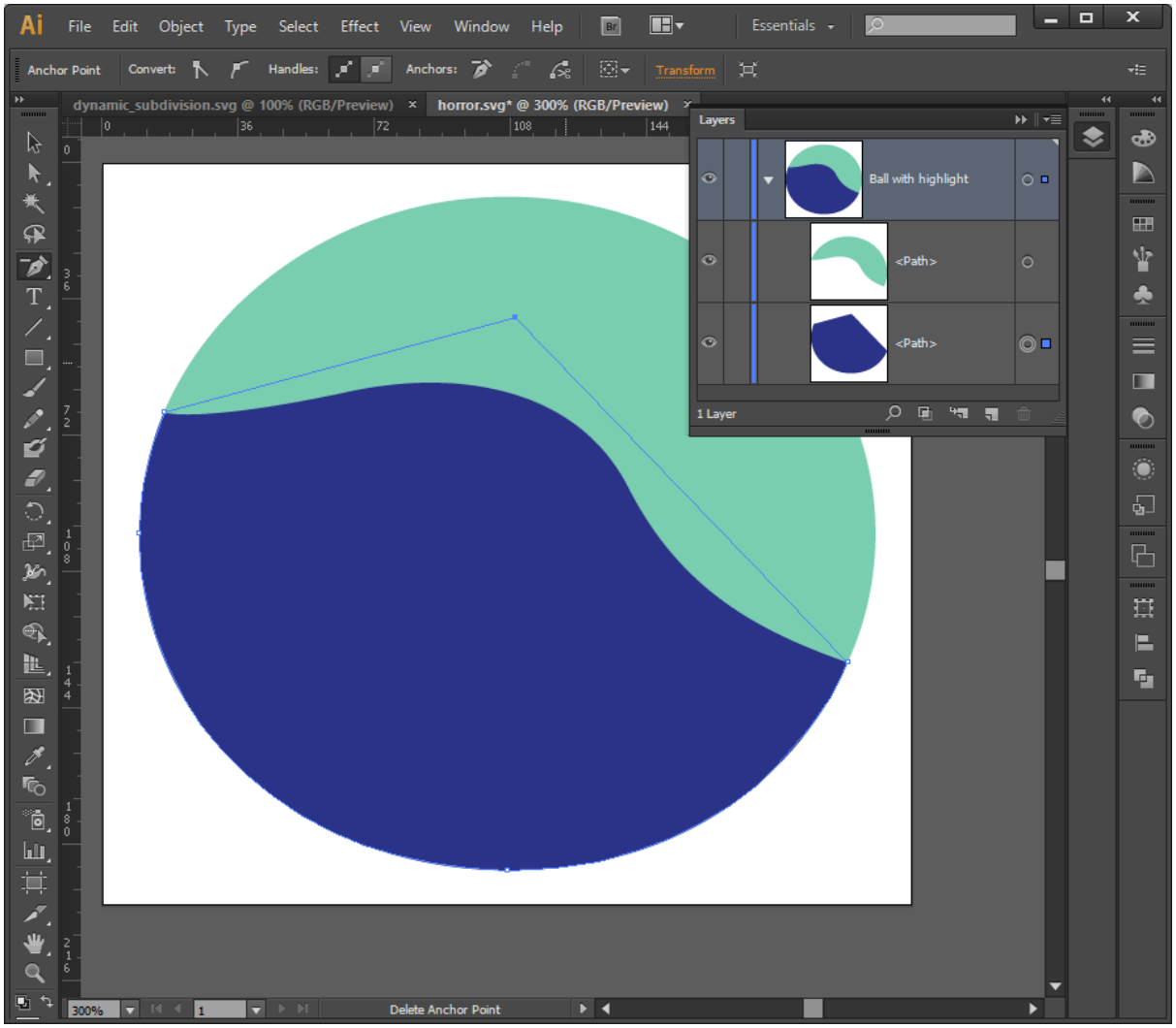
Illustrator allows one to create all sorts of nonsensical shapes, like for example paths with only one control point or filled paths with all control points in a straight line, without stroke and no curvature. They might look "ok" and not interfere in Illustrator when you don't look too closely, but they do come knocking when they should be built into a triangle mesh. The importer may warn about failing to build these shapes.

Sometimes building a mesh may fail due to too high control point amount in a small area. This is mostly due to numerical rounding errors producing all sorts of weird shapes. This is usually easily avoided by cleaning your shapes before exporting. Not only does it make the final mesh better, it also makes the source document more maintainable. So just do it!

In most cases, the importer will just skip such problematic shapes and complain about them in the console view.

Simplify hidden geometry

Avoid having smooth curves in places that will not be visible anyway. The following picture demonstrates a simple case where a shape's edge is covered by another shape and can thus be simplified without any negative effect on the artistic properties of the graphic:



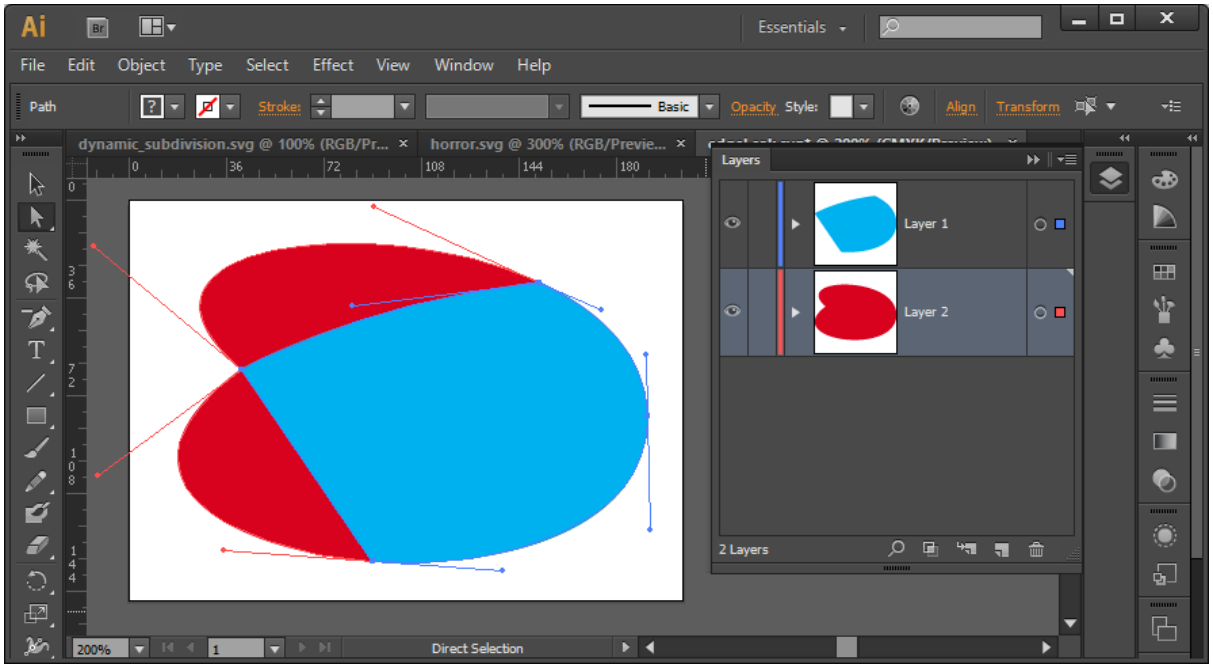
The top side of the blue has been simplified as it is not visible. This has a positive effect on the complexity of the final mesh.

Avoid touching curves and elliptical arcs

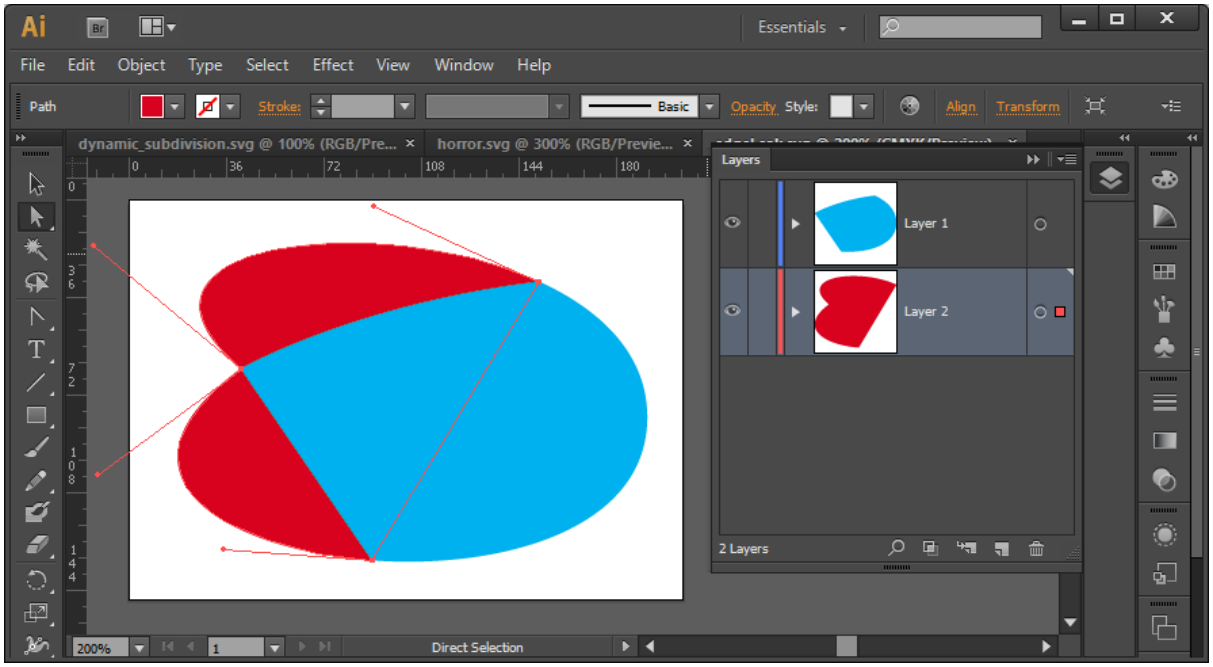
Leaks may appear if several curves touch as their shapes are built individually and the curves' subdivision points may not match exactly. This means that several curves may follow a slightly different path. Having such areas in your document leads to unpredictable results.

In the following picture, the red and blue shapes should overlap exactly, but due to differences in curve approximation, a sawtooth leak pattern emerges:

The following picture demonstrates another potential situation where a similar sawtooth pattern may appear:

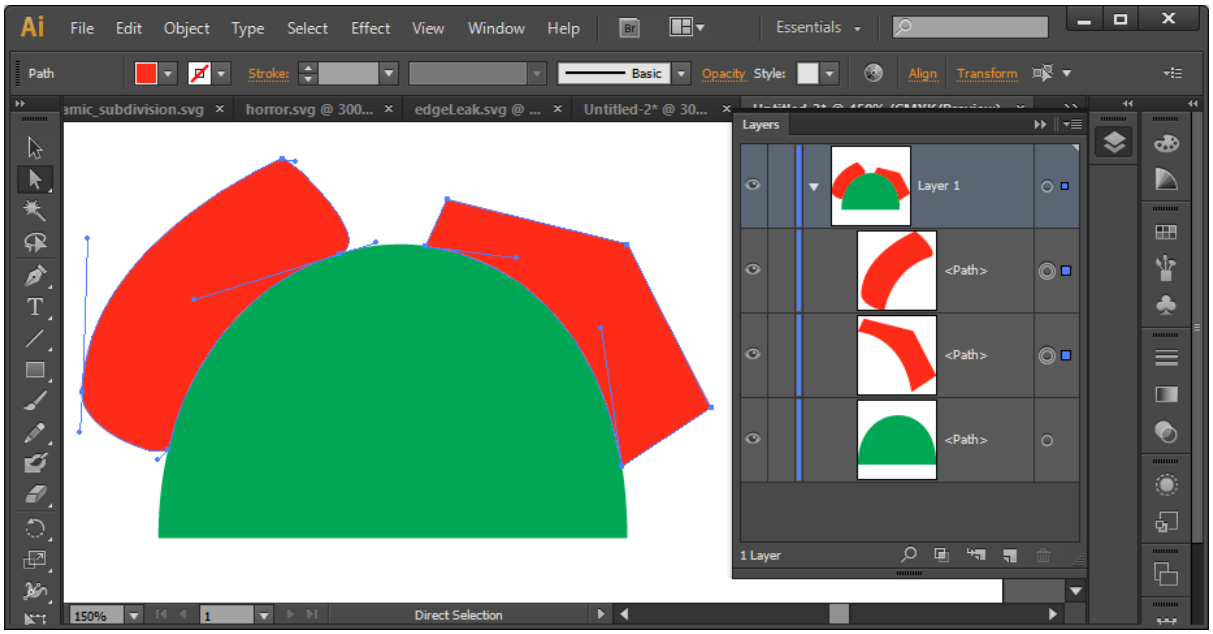


To avoid leaks, the underlying red shape is simplified so that it’s curves no longer touch the blue shape:



The resulting picture looks exactly the same as the the previous one. It will also have much simpler runtime geometric complexity and thus better performance.

Here is another example of a very bad shape structure that would be very easy to fix by extending the orange parts so that they are properly under the green shape:



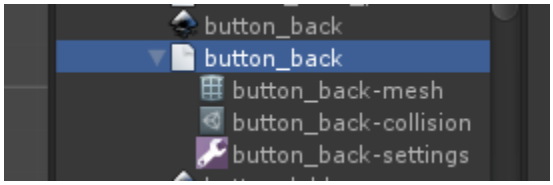
If left unfixed, it is likely that holes will appear at the seam where the orange shapes touch the green shape.

Practical applications of Bézier curves are always mere approximations of the mathematical model, as exact reproduction would require infinite number of vertices along the path's length.

Importing and use in Unity

SVG Assets

Importing .svg files to Unity has been made as simple as possible. When you add an SVG file to the Unity's Assets folder, after which the SVG file is imported automatically and several assets for different purposes are generated and added to the project. The SVG file itself is left unmodified so the user can modify and save the file and any changes are automatically re-imported.



The generated assets are stored right besides the original SVG file. An asset container has been created which stores the mesh, collision data and import settings. Descriptions of their purpose and instructions on how to use these generated assets can be found in the Inspector when you select them.

The SVG file can be renamed normally and the generated asset container and it's contents will update accordingly. If either the SVG file or the asset container is moved to another directory, the other one will be moved as well. The asset container cannot be renamed directly. This is a restriction imposed by Unity.

Problematic documents

Simply SVG reports most of the encountered errors through the Unity's console window, so it is best to keep it open at all times.

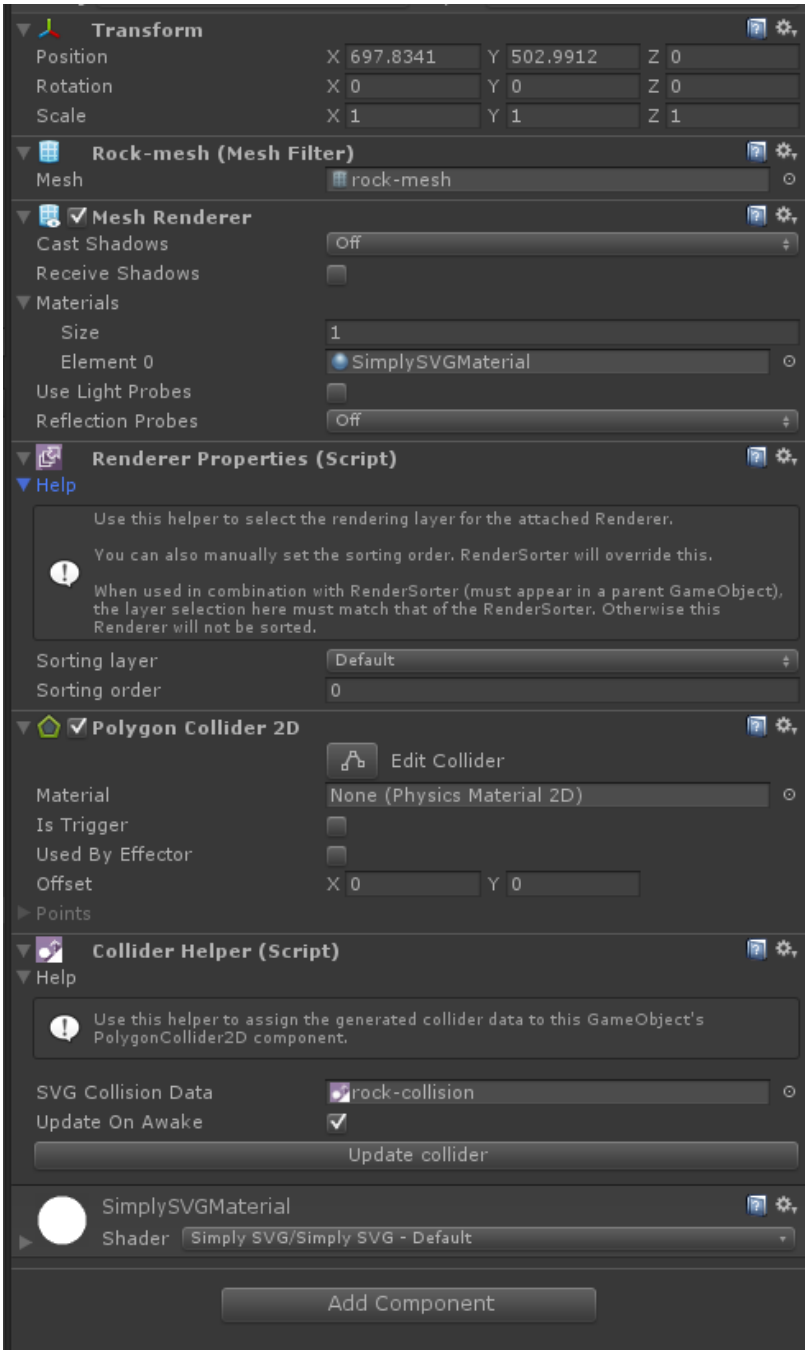
Error reporting verbosity can be controlled through the GlobalSettings object (located at Assets/SimplySVG/Settings/GlobalSettings).

See the section Content creation - Best practices for tips on how to avoid most common problems and avoid using unsupported features.

Using the generated assets

Drag & drop the SVG file to the scene

The easiest way to get the graphic into the scene, is to drag & drop the SVG file to the editor's scene view. This automatically creates a new GameObject with a typical Mesh Renderer + Mesh Filter + Collider setup. Their settings are also automatically set to sensible default values. A few utility components are also added.

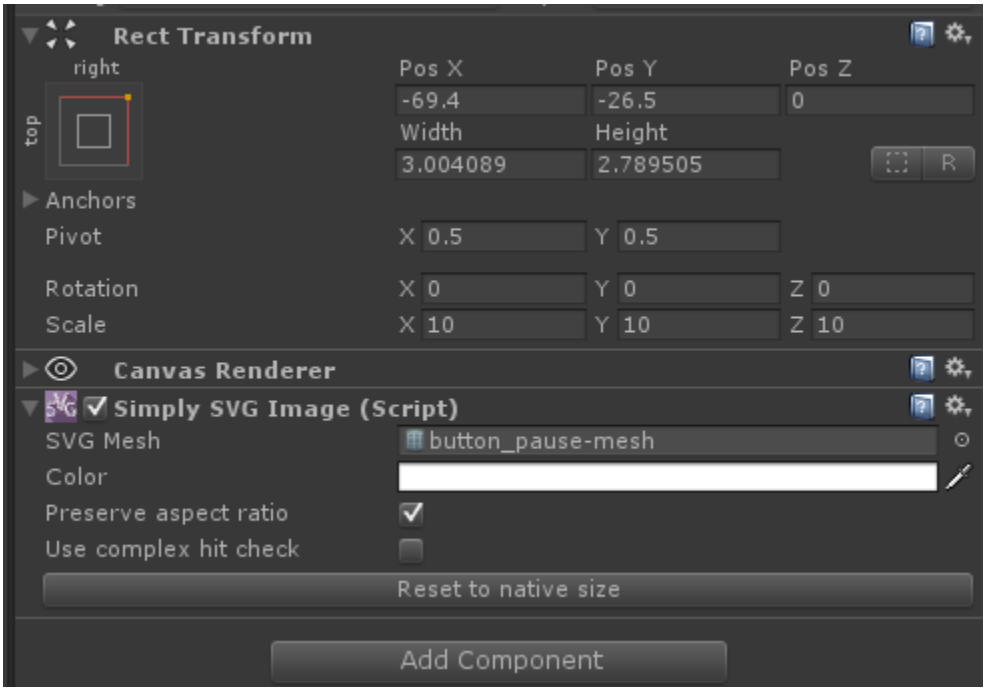


The mesh assigned to the mesh filter is the generated mesh asset of the SVG file and a material “SimplySVGmaterial” is automatically assigned to the Mesh Renderer. This is a minimal setup required to draw meshes generated from SVG files.

UI image

First create a regular UI Canvas setup. After this, you can add a SimplySVGImage GameObject to it by selecting “GameObject/UI/Simply SVG Image” from the main menu or the Hierarchy view.

The following picture shows an example of the structure of an UI element with a SimplySVGImage component attached:

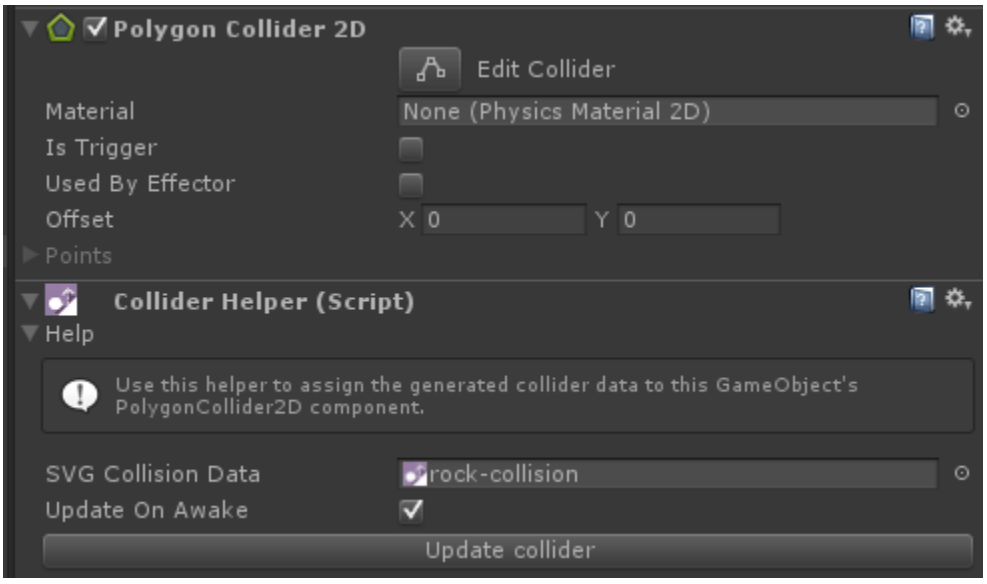


“Preserve aspect ratio” controls whether or not the graphic is allowed to stretch to fill the whole RectTransform.

“Use complex hit check” is a more precise click hit check method that accurately check the click against the actual graphic geometry. Turning this off will cause click hits to be checked against the graphics RectTransform area.

ColliderHelper component

The following picture shows an example of the structure of a GameObject with the ColliderHelper component attached.



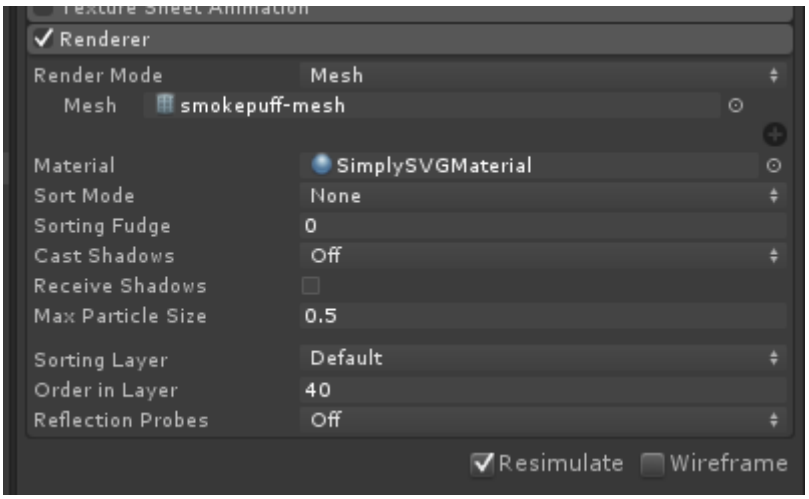
The Collider Helper component helps you create a collider for your SimplySVG GameObject. Adding the component also adds Polygon Collider 2D with it. You can drag the Simply SVG collision asset to the SVG Collision Data property field. Hit Update collider and you are done.

The ColliderHelper copies the collider information from the CollisionShapeData object and puts it in the PolygonCollider2D component. This means, that the ColliderHelper component is actually not needed after the collider has been saved to the PolygonCollider2D component.

If the SVG is reimported and a new collider is generated, this new collider is not automatically uploaded to the PolygonCollider2D in the editor. As such, it is recommended to keep the ColliderHelper attached as it can then automatically update the PolygonCollider2D's collider data at runtime if the source SVG is reimported and a new collider is generated. The "Update On Awake" option controls this behaviour.

Particle System

Set the Render Mode option to Mesh in the Renderer section of the Particle System's settings. Then select a mesh generated by Simply SVG. Set the Material to SimplySVGMaterial.



Additional tools

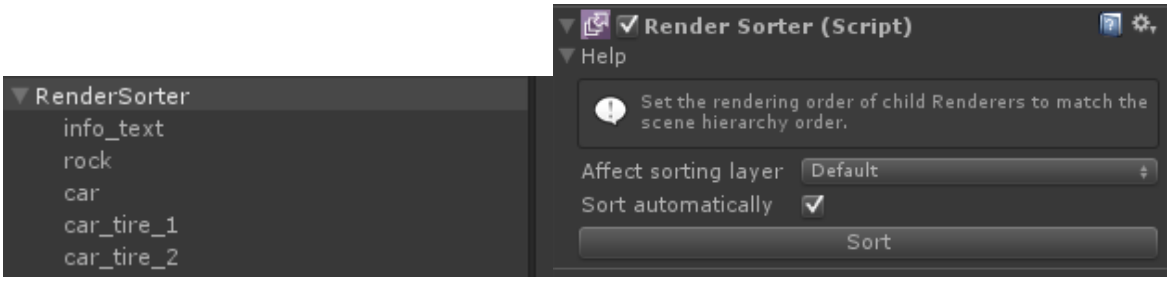
RenderSorter components

A single graphic sorts itself internally during mesh build and should thus render correctly without any user interaction.

For multiple separate graphics to render in a correct order, a sort order manager is required. Unfortunately Unity only has built-in support automatic hierarchy based draw order sorting for it's UI Canvas system, so a custom sorting script has been provided for more general use cases. The RenderSorter behaves very similar to Unity's UI sorting.

Usage

To use the RenderSorter script, you need to attach it to a GameObject as a component. After which you can add you graphics as child GameObjects. You should now have a set up similar to this:



The “RenderSorter” GameObject has several SimplySVG GameObjects as its children. By default, Unity assigns new Renderers to the “Default” layer (use the RendererProperties component to change the layer if needed) so the same layer is selected for the RenderSorter too.

The “Affect sorting layer” option controls which Renderers will be included in the sorting operation. **Only child Renderers with the same layer will be sorted.** Everything else will be ignored. You can use several RenderSorters that target different layers in the same GameObject hierarchy. To modify sorting layer order, use Unity’s project settings (Edit/Project Settings/Tags and Layers/Sorting Layers).

By pressing the “Sort” button in this example context, the graphics will be sorted such that “info-text” is at the bottom and “car_tire_2” is drawn on top.

If “Sort automatically” is ticked, the rendering order will be re-sorted every frame at runtime, thus ensuring correct ordering even if the hierarchy changes. Do note that this comes with a performance cost.

Unity’s UI system has it’s own sorting functionality, so RenderSorter should not be used with it.

RendererProperties component

The RendererProperties component is used to make some hidden settings in Unity’s Renderers easily editable. For now, its main purpose it so allow controlling draw order.

The “Sorting layer” property is used to select which sorting layer the this GameObjects Renderer should be associated with.

The “Sorting order” can be used to manually set the order of this GameObject’s Renderer in the assigned layer. If RenderSorter is used, this value will automatically be overridden.

If you are using RenderSorter, the layer selection of the RendererProperties should match that of the RenderSorter.

Trouble?

Should you experience problems when importing SVG documents to Unity, here's a few things you can try and remember:

1. Make sure the export settings in your graphics software are correct
    a. Illustrator sometimes silently forgets and changes the SVG exporter's settings, so double check them when saving.
    b. Hitting the "Save" button in Inkscape always uses the default SVG exporter rather than the "Optimized SVG" exporter (even if this was the last one used), thus skipping all the necessary steps to to export a compatible SVG file. Always use "Save as…"
2. There are no nonsensical shapes (such as single point paths) in the document
3. Check the SVG file to see if some unsupported features are being used
    a. Feature suggestions are welcome

Still having problems? We'd love to have a look at your SVG document and try to figure out a solution. Please send an e-mail to simplysvg@nordicedu.com with the SVG document attached.