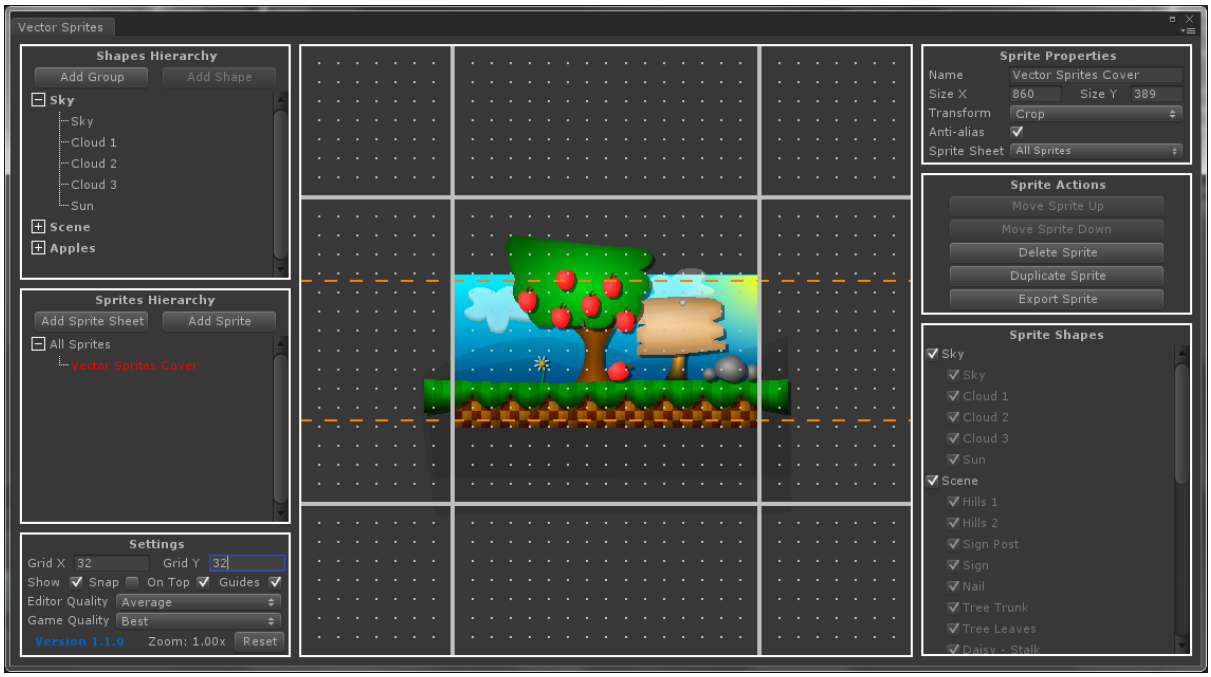
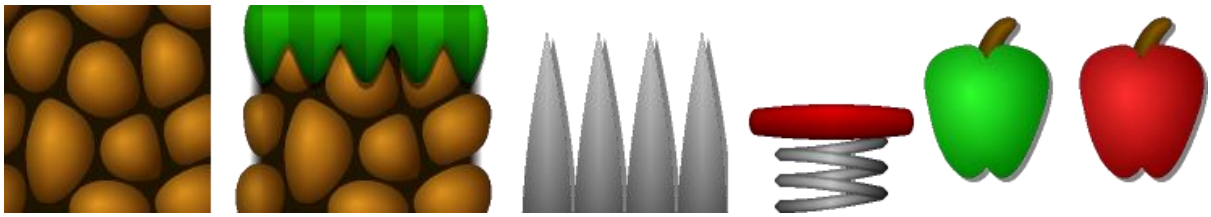
Vector Sprites (Version 1.1.0)

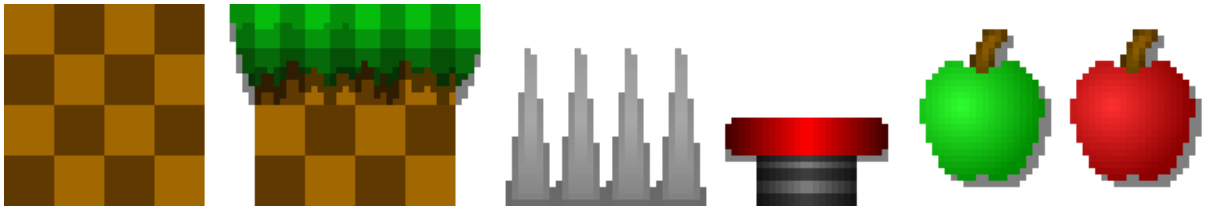
Vector Sprites is a powerful Unity GUI editor extension for generating high-quality 2D vector-based sprites. Because sprites are created using vectors rather than individual pixels, they can be exported at any size (up to 2048 by 2048 is supported) without loss of quality. Multiple sprites can be created from reusable shapes, which can then either be generated at runtime or exported as sprite sheets for which packing and slicing information is automatically generated.



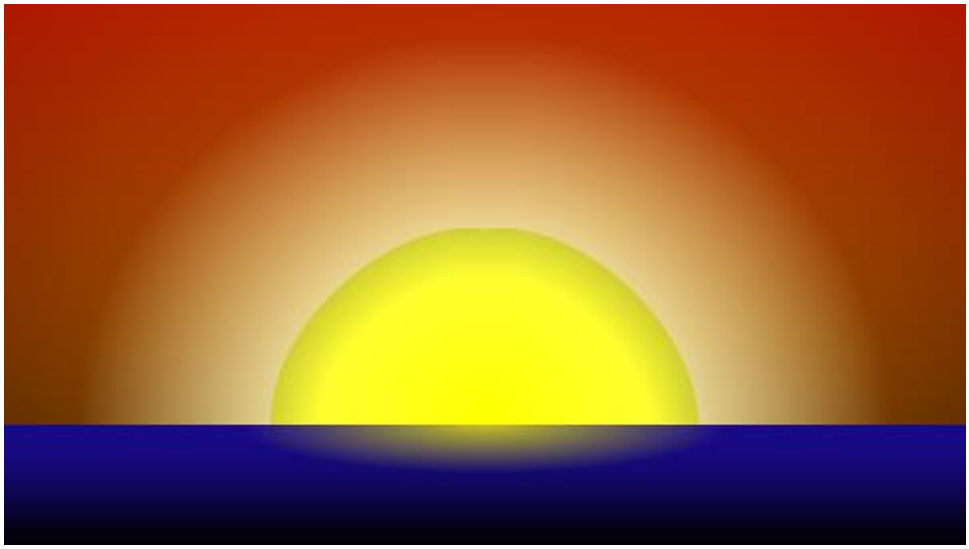
Vector Sprites is incredibly versatile and can be used to create a wide range of 2D sprites. To demonstrate its diversity, here are a few example sprites that have been generated using Vector Sprites. First of all it's great for creating tile sets for 2D games…

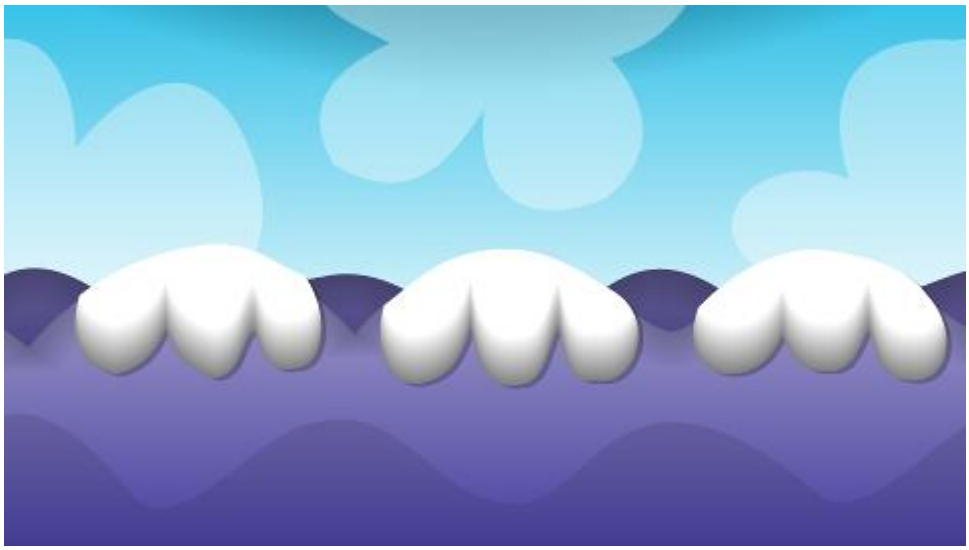


…note that the grass and soil sprites can be tiled. Or maybe the same thing but with a more retro feel…

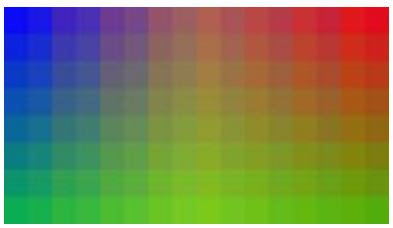


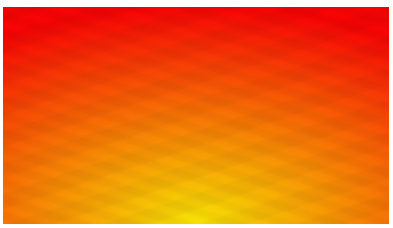
…or for creating high definition-sized, level backgrounds…





…or just some gradient-style backgrounds (scaled down here, but again can be exported at high-definition)…



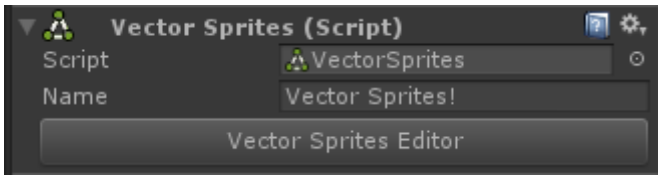


…or even for creating icon sets…

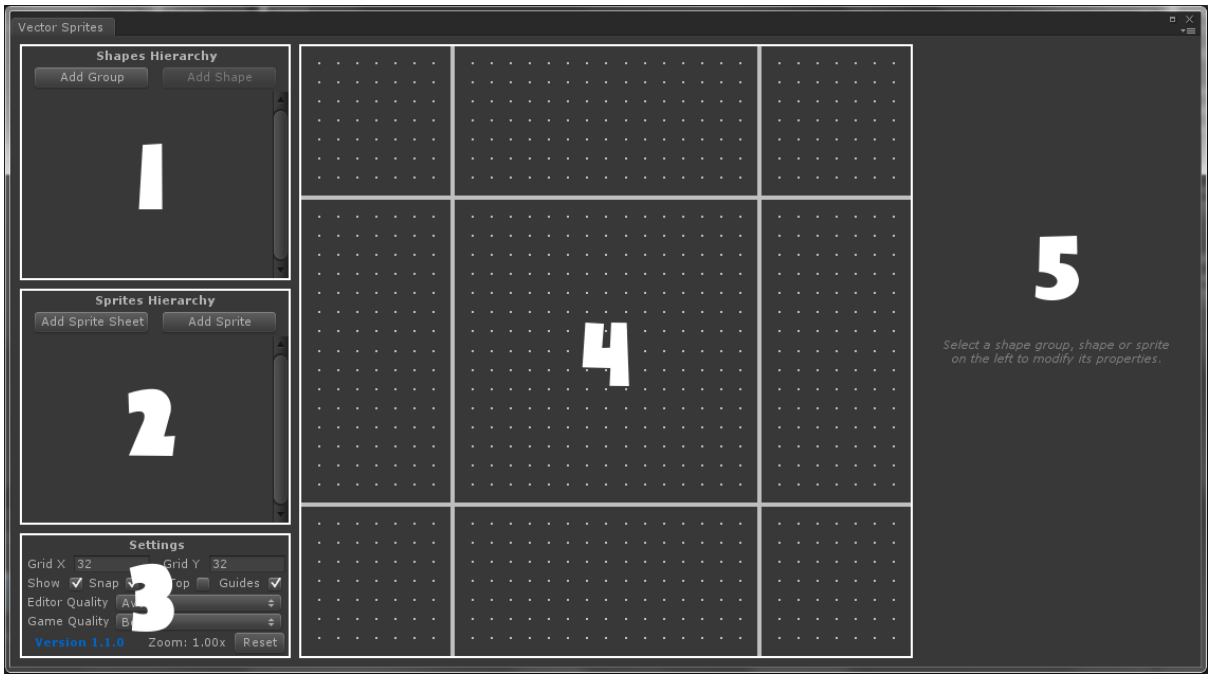
…the possibilities are endless!

Adding a Vector Sprites Component

To use Vector Sprites, add a Vector Sprites component to any game object within your scene. Vector Sprites does not require any other components, and you can add as many of them as you like – for example in a 2D platform game where each world has a different style of platforms, you could have one Vector Sprites component for each world.



Once the Vector Sprites component has been added, you can optionally give it a name so that you can identify it. After that, simply click the **Vector Sprites Editor** button to get started! Doing so will present you with the following window, henceforth known as the **Editor Window**:



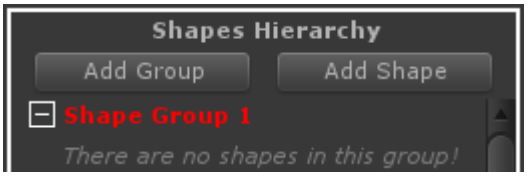
This where you will create your shapes and sprites. The sections have been numbered above and are as follows:

1. **Shapes Hierarchy** – This is where you manage shape groups and shapes. Shape groups and shapes are the basic building blocks of sprites.

2. **Sprites Hierarchy** – This is where you can create sprite sheets and sprites to be used in your game. Sprite sheets are collections of sprites, and sprites are collections of shapes and/or shape groups.
3. **Settings** – These frequently-used settings are global to the entire Vector Sprites component and are always visible.
4. **Canvas** – This is where you'll create the shapes that make up your sprites by dragging round end points and tangents with the mouse to create smooth Bezier Curves. The canvas is actually twice as big as the final sprite that will be generated from it – the centre square represents the sprite itself and the excess area around the edges allow for creating shapes that exceed these bounds, and optionally wrap.
5. **Properties** – The properties section allows you to edit the currently-selected object, be it shape group, shape, sprite sheet or sprite. The view changes depending on which of these entities is selected.

Adding Shape Groups

Before you can create shapes, you'll need to add a **Shape Group**. Click **Add Group** in the Shapes Hierarchy, and a new group named **Shape Group 1** will be automatically created and selected (selected entities are always highlighted in red).

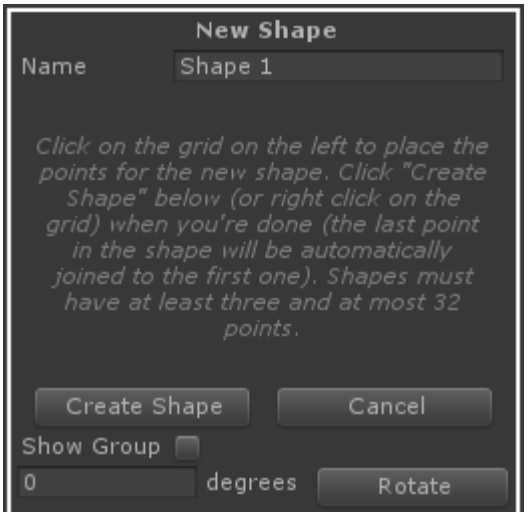


You will also notice that the properties section will have changed to display the properties of the new shape group:

Shape groups have only a few simple properties. You can give the shape group a name, you can move it up or down in the hierarchy (these buttons will be greyed out if there are no other shape groups to move the current shape group above or below), you can delete the shape group entirely or you can transform (move, rotate or scale) all shapes within the group. For more information on transforming shapes, see the **Transform Shape** action in the **Shapes** section below. The order of shape groups is important because they are drawn from top to bottom, so the last shape group will be drawn on top of everything else. Now we have a shape group, let's create some shapes!

Adding Shapes

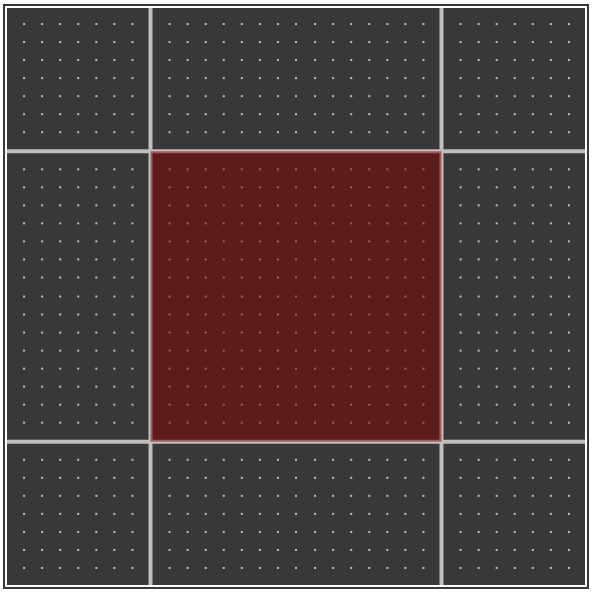
When a shape group is selected, the **Add Shape** button in the shapes hierarchy will be enabled. Click this button to add a new shape, which will display the **New Shape** properties on the right-hand side of the editor window:



Shapes are added by clicking the desired position of each point in the shape in the canvas area. Each time a point is added, a line is drawn between that point and the previous one, thus creating a polygon with straight edges. Once this polygon has been created, its edges can be dragged into curves via tangents. Shapes can have between three and 32 points.

You can cancel the creation of a new shape at any time by clicking **Cancel**. You can also give the shape a name at this point, although this can always be changed later. To finish a shape, click **Create Shape** or right-click on the canvas. This will join the last point of the shape to the first point to create a closed polygon.

Note that the visible area of the shape when the final sprite is created will be restricted to the centre square of the canvas (the area shaded in red in the screenshot below). Shapes can be still created outside these bounds, but shape points and tangents must be kept within the entire canvas area.

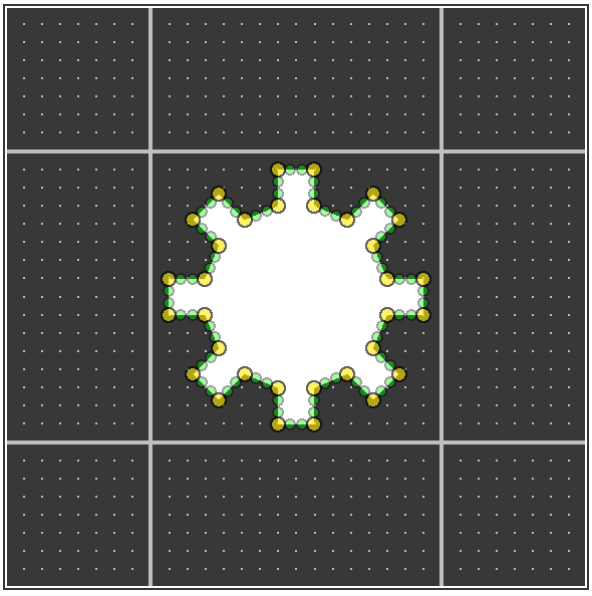


There are a couple of other settings for creating new shapes:

- **Show Group** – Shows all other shapes in the group when creating a new shape. Useful if a new shape needs to be positioned relative to another existing shape.
- **Rotate** – Rotates any points already created around the centre of the canvas. This is useful for creating uniform shapes whose points might not necessarily snap to the grid points, such as a star shape.

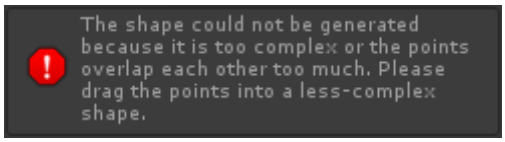
Editing Shape Points

Once you've created your shape, it will be filled in solid white by default. We'll leave it this colour for now and concentrate on how we can alter the points and tangents of the shape's edges to create the curved edges we want. The shape's points are represented by yellow circles, and the tangents by smaller green circles. Either can be dragged around the canvas to modify the shape.

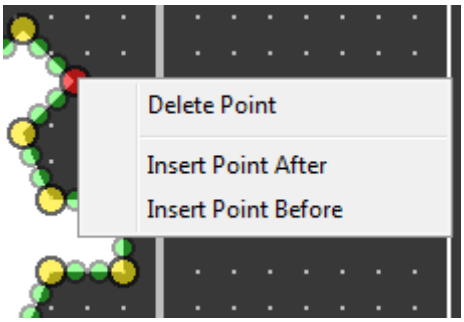


Each edge of the shape is what is known as a **Bezier Curve**. Such curves can be specified by a start and end point and two tangents (actually any number of tangents, but Vector Sprites shape

edges always use two). The best way to work out how the tangents affect the shape is to experiment – just about any shape can be created if the points and tangents are in the right place. You may also notice that certain arrangements of points and tangents will cause an invalid shape, which will cause the white filled in area of the shape to disappear. Such arrangements cannot be successfully filled in by Vector Sprites, but if this is the case, an error message will be shown and you will have the opportunity to adjust the shape points to correct this:

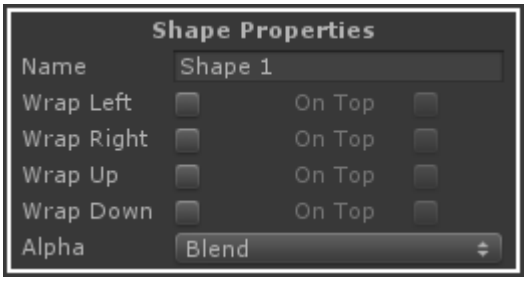


You can also add or delete shape points by right-clicking on the points themselves, which displays a context menu with these self-explanatory options:



Shape Properties

Shapes have more properties than any other Vector Sprites entity, and the properties displayed can depend on what is selected within them. The top section, however, is always displayed:



Like all other entities, shapes have a name to allow you to identify them. Below this are the wrapping and alpha blend properties as described below.

Shape Wrapping

Wrapping allows an identical copy of a shape to be displayed at the opposite end of the sprite to allow the shape to be tiled without you having to create a second, identical shape. The wrapped shape will be displayed on the canvas and will automatically update as the original shape is modified. Wrapping can be set on the left, right, top or bottom of a shape, and Vector Sprites will automatically wrap in the corners when two adjacent sides are wrapped – for example if you choose to wrap the shape upwards and to the left, the upper-left corner will also wrap. This allows you to place a shape overlapping a corner, and for it to be copied in all four corners of the sprite.

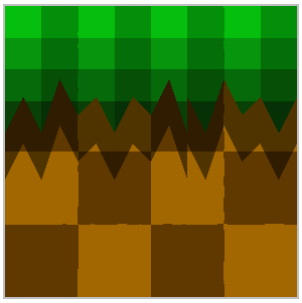
The **On Top** flags simply specify whether the wrapped copy of the shape should be drawn on top or below the original, which is only of any relevance when the shape and its wrapped copy overlap. For shapes wrapped in corners, the left/right **On Top** flag will take precedence over the up/down **On Top** flag.

Alpha Blending

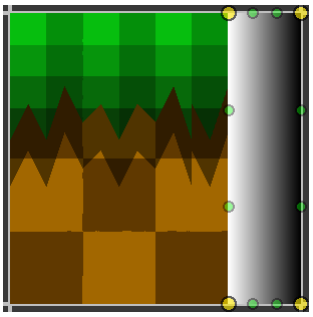
Alpha blending is an advanced technique to allow a shape to overwrite the transparency of any shapes beneath it. The two settings are:

- **Blend** – The default setting and the one that will be used most of the time. The alpha level of a shape is blended with the shapes below it, so for example a semi-transparent black shape will darken any colours beneath it.
- **Overwrite** – This setting allows a shape to be created that overwrites the alpha levels of any shapes below it, thus fading them in certain areas, regardless of how many shapes are underneath. For shapes that have overwrite alpha blending, the colour of the shape is irrelevant – only the alpha level is used. When the itself shape is selected, it is shown with alpha levels going from white (opaque) to black (transparent), otherwise the shape on its own would be invisible. When the shape is shown as part of a group or a sprite, the alpha overwriting takes place.

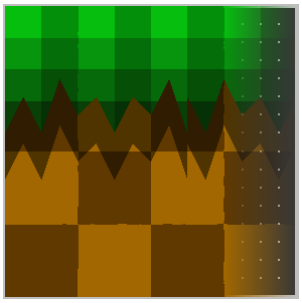
An example of how overwriting alpha levels can be used is shown in the screenshots below. Firstly, a basic sprite for a platform game is created:



This sprite can obviously be tiled, but what if we want to create a sprite using the same shape, but that fades out at the edge? We can achieve this by adding another shape that overwrites the alpha. Here is the same sprite but with an additional shape with alpha blending set to **Overwrite**. Note that because it is selected, it is showing the alpha level as black and white:

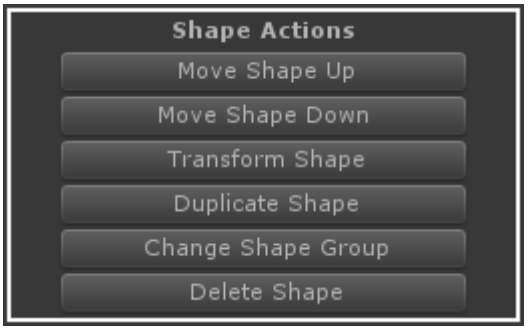


When the entire Shape Group or a sprite containing it is selected, the alpha is overwritten as you would expect, and we have another Sprite with a faded edge:

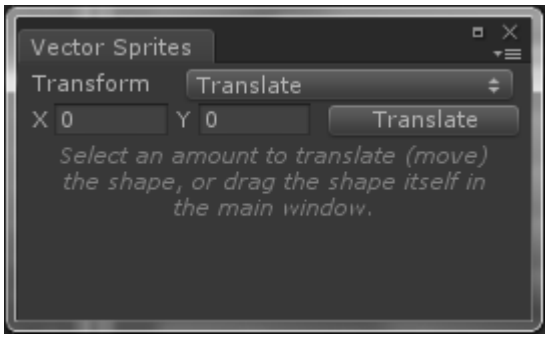


Shape Actions

Underneath these properties are a series of buttons that represent actions that can be performed on the selected shape. A description of these is below:



- **Move Shape Up/Down** – Moves the shape up or down within the shape group. If either one of these buttons is greyed out, it is because the shape is already at the top or bottom of the group. The order of shapes in a group is important because shapes are drawn from top to bottom in a group, with each shape being drawn on top of the previous one.
- **Transform Shape** – Opens up a dialog window allowing the selected shape(s) or shape group(s) to be translated (moved), rotated or scaled:

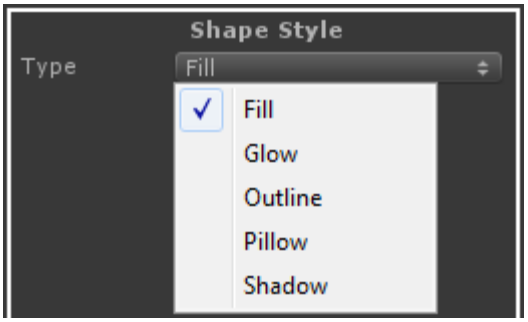


  Shapes can be transformed either be entering absolute values in the text boxes provided, or, as long as this window remains open, dragged within the main canvas. An icon in the centre of the shape on the canvas indicates what operation – translating, rotating or scaling – is selected. Rotating and scaling can be performed with either the centre of the shape or the centre of the sprite as the origin. When you have finished transforming the shape and wish to return to modifying the individual shape points, simply close the transform window. Alternatively the transform window will automatically close if another entity is selected.
- **Duplicate Shape** – Creates a copy of the selected shape in the same group, then automatically selects the copy.

- **Change Shape Group** – Use this option to move the selected shape(s) to another group. As long as there is at least one other shape group, a dialog window will be displayed when the button is clicked containing a list of shape groups. Simply click on one of these shape groups to move the selected shape(s) into that group.
- **Delete Shape** – Deletes the current shape(s).

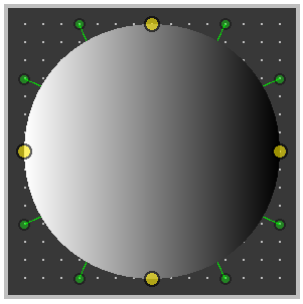
Shape Style

This section is where you will decide how to decorate the shape. There numerous options available, each one of which is described in detail below. Shapes have five configurable areas, which can be selected from the **Type** dropdown at the top of this section:

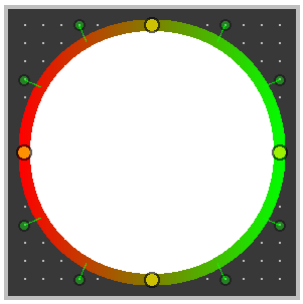


Note that any combination of these types can be used to make up a shape – the selection only refers to the active one as there isn't space for all of them in the properties area. Below is a description of each of these types, along with a screenshot showing the visual effect of each one.

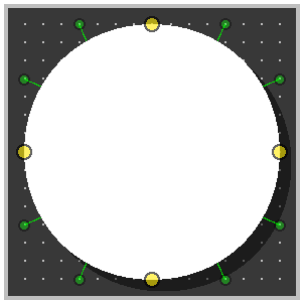
- **Fill** – The style for the centre of the shape. This shape has a gradient fill from white to black, left to right:



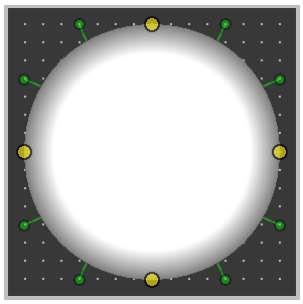
- **Outline** – The style for the outline of the shape, the width of which is configurable. This shape has a solid white fill, with a gradient outline from red to green, left to right:



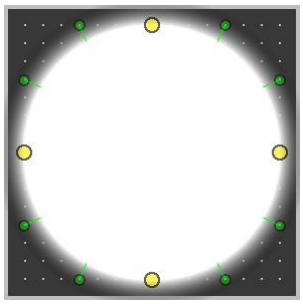
- **Shadow** – Draws a copy of the shape, offset a configurable distance from the original and in one solid colour. By making this colour semi-transparent black, a shadow effect can be created:



- **Pillow** – Pillow shading is a technique used to give 2D shapes a more 3D effect by shading round the inside edges, usually in a dark colour that fades as it gets to the centre of the shape:



- **Glow** – Glow shading is the opposite to pillow shading – it creates a similar effect but on the outside of the shape. Setting the colour to a semi-transparent white will create an effect that looks as though the shape is glowing:



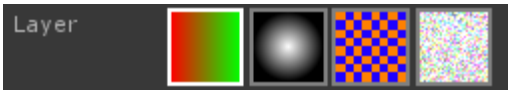
The properties displayed below the type will change depending on what type is selected. These are summarized below:

Fill/Outline Properties

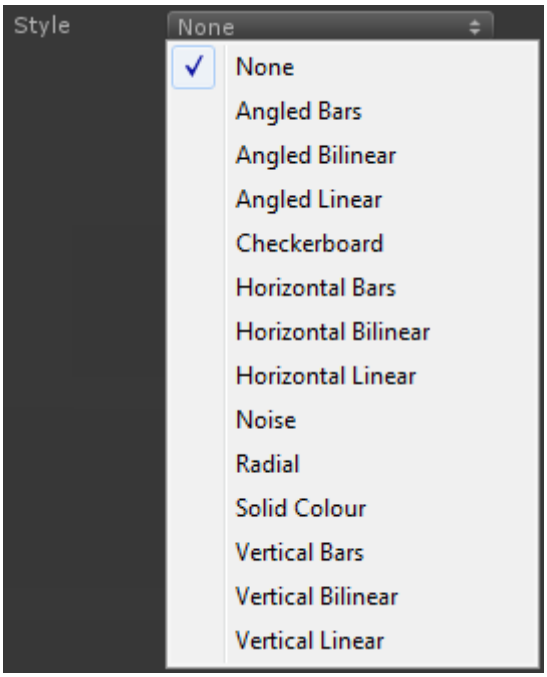
The properties for defining the style of a shape's fill or outline are almost identical. The only difference is that outlines must specify their width:

Fills/outlines work on a system of layers, up to four of which are supported. Each layer is simply a different set of properties which represent a way to style the shape, and the layers are

drawn one by one on top of each other. By making later layers partially transparent, different styles can be blended together. Layers are selected by clicking on one of the four layer boxes, which themselves show a preview of the current layer:



Once a layer has been selected, a particular style for that layer is required. There are a wide range of styles available:

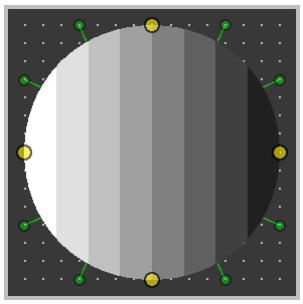


- **None** – Do not apply a style to this layer.
- **Angled Bars** – A configurable number of bars that alternate between two colours, at a specified angle.
- **Angled Bilinear** – A bilinear gradient shade that smoothly transitions from one colour to a second colour and back to the original colour at a specified angle.
- **Angled Linear** – A linear gradient that smoothly transitions from one colour to a second colour at a specified angle.
- **Checkerboard** – A checkerboard of configurable size consisting of two alternate colours, and optionally at a specified angle.
- **Horizontal Bars** – A configurable number of bars going horizontally across the shape that alternate between two colours.
- **Horizontal Bilinear** – A bilinear gradient shade that smoothly transitions from one colour to a second colour and back to the original colour going across the shape.
- **Horizontal Linear** – A linear gradient that smoothly transitions from one colour to a second colour going across the shape.
- **Noise** – Random noise – i.e. random changes in the colour of each pixel.
- **Radial** – A smooth circular gradient from one colour to a second colour radiating out from a centre point.

- **Solid Colour** – A single solid colour.
- **Vertical Bars** – A configurable number of bars going vertically down the shape that alternate between two colours.
- **Vertical Bilinear** – A bilinear gradient shade that smoothly transitions from one colour to a second colour and back to the original colour going down the shape.
- **Vertical Linear** – A linear gradient that smoothly transitions from one colour to a second colour going down the shape.

Each of these styles in turn has their own set of properties, many of which are shared between styles. Here is an explanation of each of these properties:

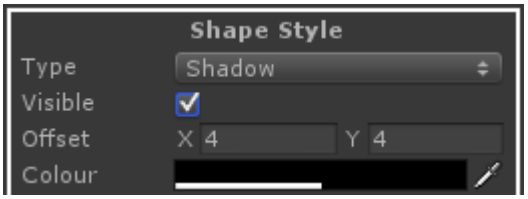
- **Colour 1** and **Colour 2** – The two colours required for the style. All styles require two colours except **Solid Colour** and **Noise**, which required only one.
- **Bias** – Gradients allow a bias between colour 1 and colour 2. The bias is a decimal number between 0 and 1 - the default is 0.5, meaning both colours are used equally. A bias of 0 means colour 1 is used entirely and a bias of 1 means colour 2 is used entirely.
- **Bands** – Gradients can be separated into colour bands to clearly show each individual shade of the colour. The default value, which is also the maximum value, is 255 which results in smooth shading where bands are not visible. The screenshot below shows a horizontal linear gradient with eight colour bands:



- **Area** – The area over which the style should apply – either the area of the shape or the area of the sprite. Select **Sprite** to keep patterns the same within the sprite bounds regardless of how much the shape overlaps the edges.
- **Angle** – For styles that can be rotated, specifies the angle between 0 and 360 degrees.
- **Centre X/Y** – An optional offset of for the centre position of the style. Particularly useful for the **Radial** style to allow it to be positioned somewhere other than the centre of the shape.
- **Bars/Squares** – For styles involving bars and checkerboards – specifies how many bars or squares to create.
- **Size** – For the **Radial** style only – specifies the size of the radial.
- **Noise Type** – For the **Noise** style only – noise can be either **RGB** or **Random**. Both are essentially random, but **Random** noise adjusts pixel values in a completely random way whereas **RGB** noise adjusts pixels based on the shape's colour.
- **Noise Level** – For the **Noise** style only – the amount of noise to add.

Shadow Properties

Shadows are relatively simple. They can be turned on or off via the **Visible** flag, and have an **Offset** and **Colour**:

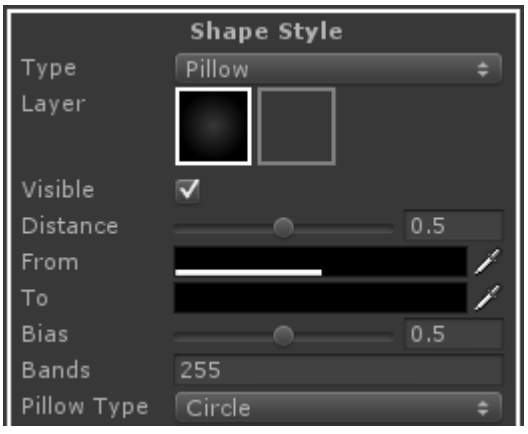


The offset specifies how far away from the original shape the shadow should be created, and both the X and Y directions can positive or negative. The colour represents the colour of the shadow – try a semi-transparent black as in the screenshot above for best effects.

Pillow Properties

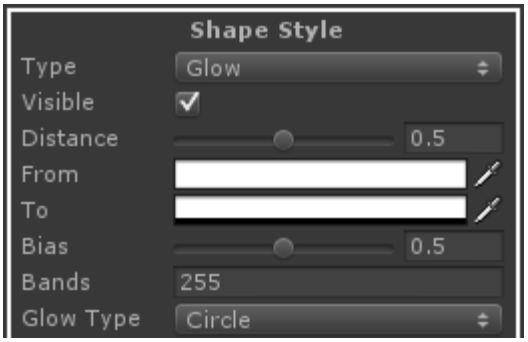
Pillow shading, like outline/fill styles can have multiple layers – in this case two. Layers that have their **Visible** flag set can have the following properties:

- **Distance** – The distance into the shape that the pillow shading should extend.
- **From** – The starting colour of the pillow shading – i.e. the colour nearest the edge of the shape.
- **To** – The ending colour of the pillow shading – i.e. the colour nearest the centre of the shape.
- **Bias** – The colour bias – see the outline/fill style for more details on bias.
- **Bands** – The number of colour bands to show – see the outline/fill style for more details on bands.
- **Pillow Type** – Whether the pillow shading should occur all the way around the shape (**Circle**) or at a specific angle only (**Angle**). If **Angle** is selected, an angle in degrees must also be specified. This is the main reason why pillow shading supports two layers – one for each type.



Glow Properties

Glow properties are almost identical to pillow properties, other than the fact that they don't support multiple layers. For glow shading, the **From** colour represents the colour at the edge of the shape, and the **To** colour represents the colour going away from the shape.



Sprite Sheets and Sprites

Now that we've covered how to create shape groups and shapes, we can use these to build sprites, and we can optionally place these sprites inside sprite sheets. Sprites are the final product of Vector Sprites – they're exported or created at runtime and can be used in your games.

Sprite Sheets

Sprite sheets are a collection of sprites that can be exported into a single sprite sheet for use in your games. A single sprite sheet is often favoured over many individual sprites because it allows for all sprites to be referenced from the same texture and improves performance. To add a sprite sheet, click the **Add Sprite Sheet** button at the top of the sprites hierarchy:

Clicking this button will add a sprite sheet to the hierarchy, which will be automatically selected. Sprite sheets have only a single configurable property – their name. They also have a handful of actions:

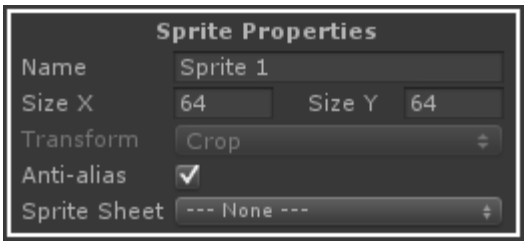
- **Move Sprite Sheet Up/Down** – Moves the sprite sheet up or down within the sprites hierarchy. The order of sprite sheets makes no difference other than to allow you to organise the hierarchy as you see fit.
- **Delete Sprite Sheet** – Deletes the sprite sheet. Any sprites associated with it will **not** be deleted – they will simply switch to having no parent sprite sheet.
- **Export Sprite Sheet** – Exports the sprite sheet to a PNG file in the assets folder with sprite sheet information attached to it. This button is only enabled if the sprite sheet has at least one sprite associated with it.

Note that sprite sheets are only of any relevance when they are exported. If you choose not to export your Vector Sprites sprites but rather to generate them at runtime (see **Using Generated Sprites** below for more information), there is no benefit, other than for your own organisational purposes, to using sprite sheets.

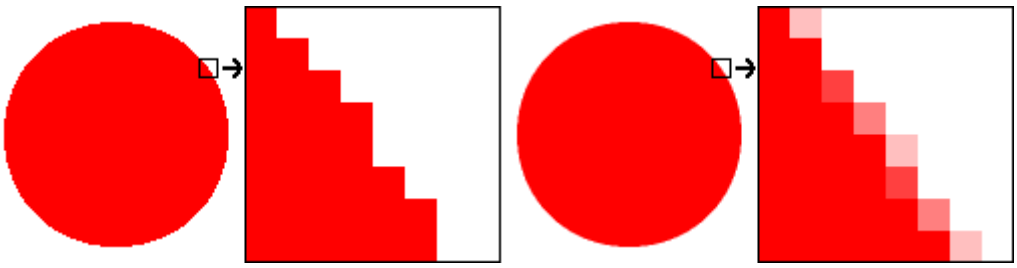
Sprites

Sprites are simply a collection of shapes and/or shape groups. Sprites can optionally be associated with a sprite sheet, which will allow them to be exported as part of that sprite sheet, or can be standalone. Click the **Add Sprite** button – next to the **Add Sprite Sheet** button in the screenshots above – to add a sprite.

By default, newly-created sprites are not associated with a sprite sheet. When they are created they are automatically selected, at which point the properties section on the right will display their properties. These are as follows:



- **Name** – The name of the sprite to allow you to identify it.
- **Size** – The final exported size of the sprite. This can be anything from 1x1 to 2048x2048, and does not have to be square.
- **Transform** – This is only relevant for non-square sprites and specifies how the sprite should be transformed so it fits within bounds. All shapes are created within a square area on the canvas, so if a non-square sprite is required, the transform options are:
    - **Crop** – The sprite is cropped to fit within the required size, so the top and bottom, or left and right, are cut off. To give you an idea of how much of the image is being cropped, the canvas window will show additional lines to indicate the crop area.
    - **Scale** – The sprite will be scaled to fit within bounds.
- **Anti-alias** – Anti-aliasing is a technique used to smooth out the edges of lines that are not horizontal or vertical, making them look less pixelated. In the image below, the circle on the left is not anti-aliased, whereas the one on the right is. The anti-aliased circle looks smoother because the pixels don't have to be either red or white – they can transition between the two colours around the edge. Unless you are specifically creating pixelated sprites, it is recommended to keep anti-aliasing turned on.



- **Sprite Sheet** – The sprite sheet that the sprite is associated, if any.

Sprite Actions

Below these properties are a set of actions that can be performed on a sprite:

- **Move Sprite Up/Down** – Moves the selected sprite up or down the list of sprites. It doesn't make much difference where the sprite is in the list, other than the fact that you can order them in a way that makes sense to you.
- **Delete Sprite** – Deletes the selected sprite(s).
- **Duplicate Sprite** – Creates a copy of the selected sprite and automatically selects it.
- **Export Sprite** – Opens up a new window containing the exported sprite at the exported size. From here, the sprite can be exported to a PNG file in the assets folder, or it can be viewed tiled.

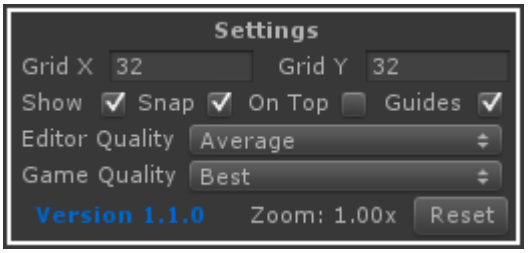
Sprite Shapes

Sprites are a collection of shape groups and shapes. Simply select the shape groups or shapes you would like to include in the sprite from the list and they will appear on the canvas and in the sprite when it is generated/exported.

Selecting an individual shape (e.g. **Shape 2** in the example above) adds that shape only to the sprite. Selecting a shape group (e.g. **Shape Group 2** in the example above) adds all shapes within that group, including any shapes that are added to the group in the future.

Settings

The bottom-left area of the editor window contains a few settings that apply to the entire Vector Sprites instance rather than individual shapes, groups, sprite sheets or sprites. These options are as follows:

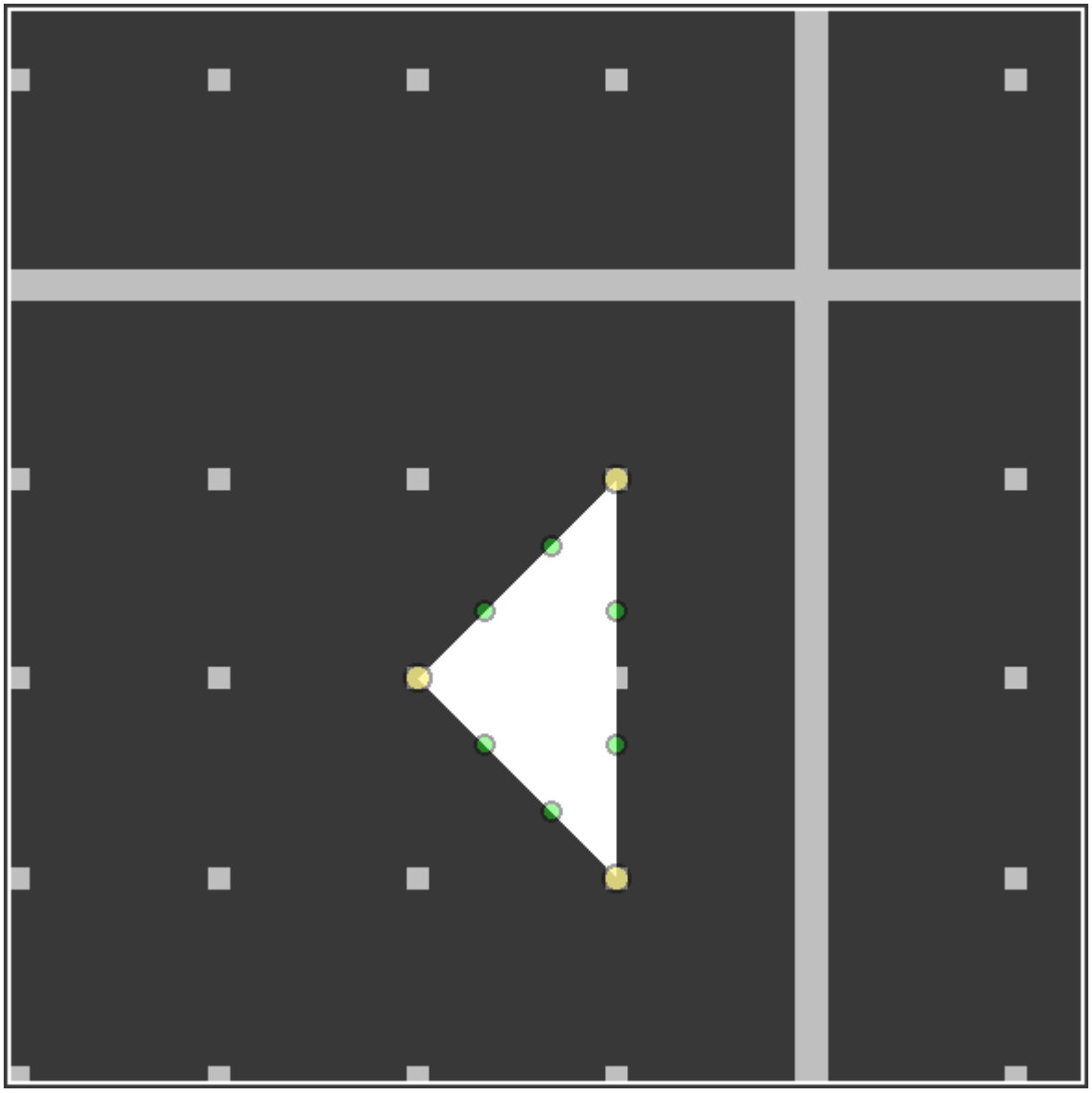


- **Grid** – The number of points going across and down the grid in the canvas.
- **Show** – Whether to show the grid points.
- **Snap** – Whether to snap to the grid points when creating shapes, dragging shape points and tangents, or transforming shapes.
- **On Top** – Whether to display the grid on top of all shapes.
- **Guides** – Whether to show the guides – i.e. the sprite boundaries.
- **Editor Quality** – The quality of shapes to render in the editor window. There are five quality settings and they relate to the number of points that the curves are split into. Fewer points will be quicker to generate, but will result in rougher-looking edges – so reduce the quality if you are experiencing slow down.
- **Game Quality** – Same as editor quality but for the exported sprites. It is recommended to leave this at **Best** to give the best results – certainly for larger sprites. The additional time taken is usually negligible for a one-off sprite generation.
- **Version 1.1.0** – Click this to display a version history of Vector Sprites, so you know what has changed since the last release.
- **Zoom** – Displays the current zoom level of the editor window (from 1x to 50x). Clicking the **Reset** button resets the zoom to 1x.

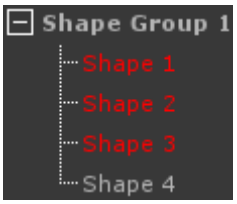
Advanced Editor Features

Once you've got the hang of the how the main building blocks of Vector Sprites work, there are a couple of additional features of the editor that are useful to know, and make editing shapes and sprites easier. These are:

- **Zoom** – Vector Sprites supports zooming the main canvas by up to 50x, which is incredibly useful for editing very small shapes. To zoom, simply scroll the mouse wheel up (to zoom in) or down (to zoom out) when the mouse is over the canvas. The zoom level is displayed in the **Settings** section and can be reset to its original size by clicking the **Reset** button next to it at any time. When zoomed in, you can drag the mouse over an empty space on the canvas to move the visible area of the canvas. The screenshot below shows a small shape that has been zoomed in to about 6x magnification:



- **Multi-select** – Multiple entities (shape groups, shapes, sprite sheets or sprites) of the same type can be selected at once by holding the **Control** button on the keyboard and clicking each entity. Selected entities are displayed in red.



In the screenshot above, three of the four shapes are selected. When multiple entities are selected, the properties and actions available are often cut-down as many don't apply to more than one entity at once. However multi-selection can be very useful to, for example, move multiple shapes by the same amount at the same time, set the size of multiple sprites to be the same, or to perform deletions of multiple entities at once.

Using Generated Sprites

You have two options for using sprites generated by Vector Sprites in your Unity projects. You can either have Vector Sprites generate the sprites at runtime (the Vector Sprites component will automatically do this when the game is run), or you can export the sprites to a sprite sheet in the project's assets folder, which can then be used like any other sprite sheet of images.

Which method you choose is up to you – but bear in mind that large or complex sprites may take a number of seconds to export, so if you are generating sprites at runtime it may not be desirable to add this onto your scene's loading time. The advantage of sprites generated at runtime, however, is that Vector Sprites components take up a lot less memory than the exported images they generate.

An important thing to remember when exporting sprites is that Vector Sprites components on active game objects will still generate sprites at runtime, even if those sprites have previously been exported. If sprite sheets are exported, it is recommended that game objects containing Vector Sprites instances are made inactive, or are placed in scenes that are not included in builds. Having said that, it is also recommended to keep the Vector Sprites components themselves in order to maintain the vector information, which can be changed and re-exported in the future if required.

If you do choose to generate sprites at runtime, they can be accessed in code by their name using an automatically-generated dictionary:

```
VectorSpritesComponent.sprites["name"]
```