

Sprite Shatter (Version 2.0.5)

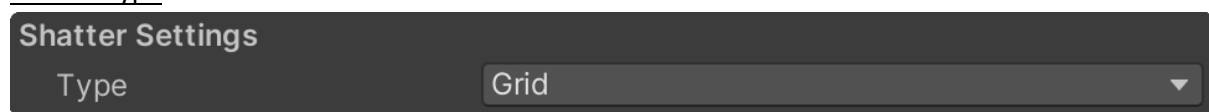
Sprite Shatter provides a quick and easy way to make your 2D sprites shatter into numerous configurable pieces. Using the simple user interface in the Unity Editor, any game object with an associated sprite renderer can be shattered into any number of pieces using varying patterns, with the pieces created as separate game objects at run-time. A single method call then shatters the sprite.

Sprite Shatter uses the source sprite's original texture – it does not create a copy so the memory overhead is low. It has support for circle, box and polygon colliders for the shattered pieces, the accuracy of which can be configured. Sprite Shatter comes with full source code (including editor code) and full prefab support. An example scene (the exploding circus!) has been provided which is ready to go – simply load this scene in Unity, click play and from there you can left-click to shatter the circus and right click to reset it.

To create a Sprite Shatter component of your own, first select a game object that contains a sprite renderer component for the sprite you want to shatter, and add a **Shatter** component to the game object. Note that the Shatter component requires a sprite renderer, so one will be added to the game object if it is not already present.

After a sprite is assigned to the sprite renderer, the Sprite Shatter properties will be displayed in the inspector window under the new shatter component. The Inspector view is split up into sections as follows:

Shatter Type



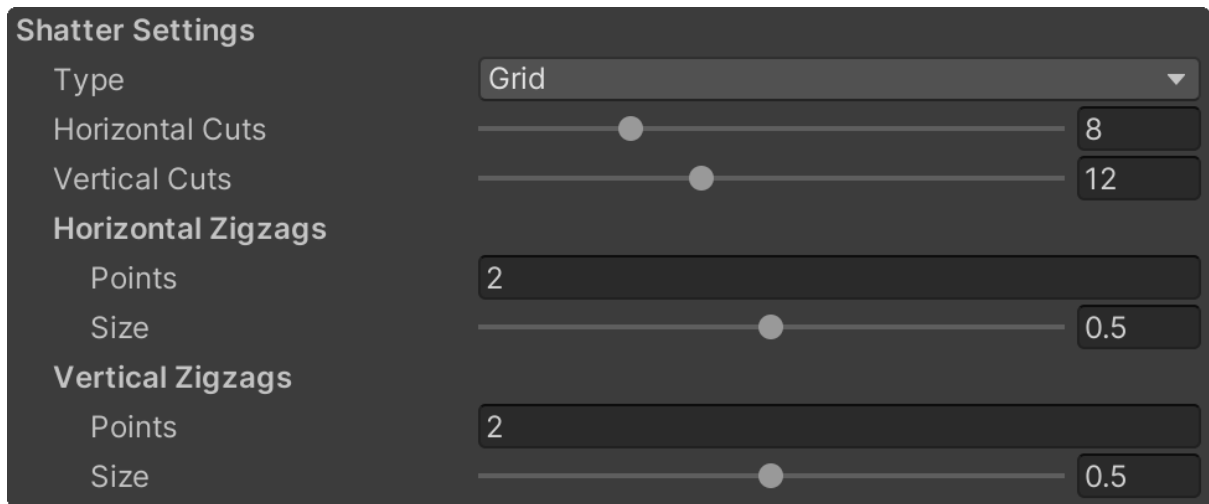
The Shatter Type property

- **Type** – this represents how you want to shatter your sprite. Currently four types are supported:
 - **Grid** – shatters the sprite in a grid formation by cutting it horizontally and vertically a specified number of times.
 - **Hexagons** – shatters the sprite into a customisable number of tessellating hexagonal shapes.
 - **Radial** – shatters the sprite outward from the centre, a bit like glass smashing from a single point.
 - **Triangles** – shatters the sprite into triangles in a horizontal or vertical orientation.

The properties for each of these shatter types are described below:

Grid Shatter Type

The grid shatter type shatters a sprite into a grid of square or rectangular pieces, optionally with zigzag edges. Randomness can be applied to make the pieces non-uniform in shape. The settings for the grid shatter type are as follows:



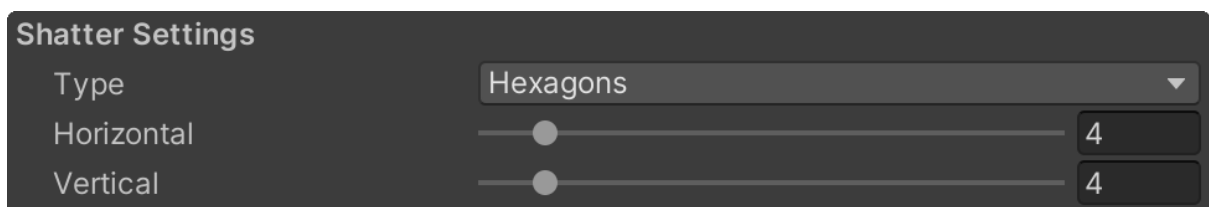
The screenshot shows the 'Shatter Settings' panel for the 'Grid' shatter type. It features a dropdown menu set to 'Grid'. Below it are sliders for 'Horizontal Cuts' (set to 8) and 'Vertical Cuts' (set to 12). Further down are sections for 'Horizontal Zigzags' and 'Vertical Zigzags', each containing a 'Points' input field (both set to 2) and a 'Size' slider (both set to 0.5).

*Settings for the **Grid** shatter type*

- **Horizontal Cuts** – the number of cuts to make horizontally across the sprite when creating the shatter pieces. Can be set between 1 and 32.
- **Vertical Cuts** – the number of cuts to make vertically down the sprite when creating the shatter pieces. Can be set between 1 and 32. More cuts mean more, smaller pieces, but too many pieces can adversely affect performance, particularly if colliders are included.
- **Horizontal/Vertical Zigzags** – these settings allow you to give the shattered pieces zigzag edges. These can be set independently in the horizontal and vertical directions, and you can set the number of points on the zigzag and the size of the zigzags.

Hexagons Shatter Type

The hexagons shatter type can be used to shatter a sprite into tessellating hexagonal shapes, the number of which can be defined in both the horizontal and vertical directions:



The screenshot shows the 'Shatter Settings' panel for the 'Hexagons' shatter type. The dropdown menu is set to 'Hexagons'. Below it are sliders for 'Horizontal' (set to 4) and 'Vertical' (set to 4).

*Settings for the **Hexagons** shatter type*

- **Horizontal** – the number of hexagons to shatter the sprite into in the horizontal direction. Can be set between 1 and 32.
- **Vertical** – the number of hexagons to shatter the sprite into in the vertical direction. Can be set between 1 and 32.

Radial Shatter Type

The radial shatter type can be used to shatter a sprite from the centre (or another point on the sprite) outwards in a circular pattern, a bit like glass smashing from a single point. The settings for this type are as follows:

Shatter Settings

Type: Radial

Sectors: 16

Radials: 2

Centre: X 0.5 Y 0.5

*Settings for the **Radial** shatter type*

- **Sectors** – specifies the number of sectors to create when shattering the piece. A sector goes from the centre point to the edge of the sprite.
- **Radials** – specifies how many times to split up each sector.
- **Centre** – the centre point at which to shatter radially from.

Triangles Shatter Type

The triangles shatter type can be used to shatter a sprite into triangular shapes, going either horizontally or vertically along the sprite. The settings are as follows:

Shatter Settings

Type: Triangles

Orientation: Horizontal

Horizontal: 10

Vertical: 8

*Settings for the **Triangle** shatter type*

- **Orientation** – triangular shattered pieces are arranged in either rows (horizontal) or columns (vertical) of triangles. Which of these you want to use can be set here.
- **Horizontal** – the number of triangles to shatter the sprite into in the horizontal direction. Can be set between 1 and 32.
- **Vertical** – the number of triangles to shatter the sprite into in the vertical direction. Can be set between 1 and 32.

Randomness

You can apply randomness to any shatter type to randomize the size and shapes of the shattered pieces so they aren't all the same shape. The settings are:

Randomness

Randomize at Run-Time: ☐

Random Seed: 0

Randomness: 0.5

Randomness settings

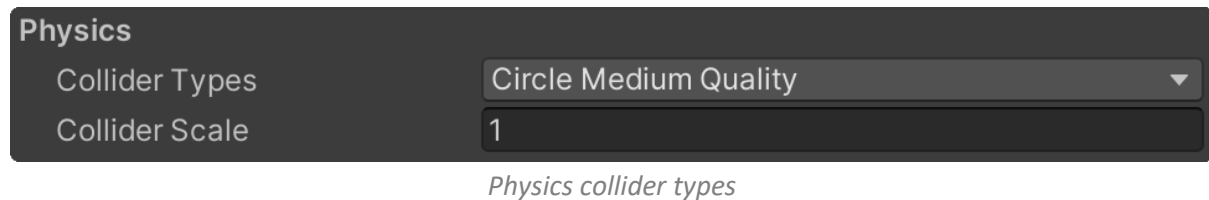
- **Randomize at Run-Time/Random Seed** – a particular random seed will generate the same size and shape of shattered pieces every time. You can set a random seed if there is a particular arrangement of shattered pieces that you like and want to be the same every

time. Alternatively you can tick **Randomize at Run-Time** which will hide the random seed field and ensure a different set of random shattered pieces are generated each time.

- **Randomness** – this is the amount the sizes and shapes of the shattered pieces will change. 0 will not apply any randomness, 1 will apply the maximum amount.

Physics

This is where you can decide on which colliders, if any, are attached to the shattered pieces that are generated. You can create colliders to allow the shattered pieces to interact with other physics objects in your scene:



- **Collider Types** – specifies the type of collider to attach to the shattered pieces, which are split into the following groups:
 - **None** – no colliders will be added to the shattered pieces.
 - **Circle** – circle colliders will be added. The position and radius of each circle collider will be automatically adjusted to best fit the shattered pieces.
 - **Box** – box colliders will be added. This position and size of each box collider will be automatically adjusted to best fit the shattered pieces.
 - **Polygon** – 2D polygon colliders will be added, which will approximate the shape of each sprite piece.

Furthermore, these types are split down into these quality levels, with lower quality collider types providing a less accurate approximation of the shatter piece but being quicker to generate:

- **Low Quality**
 - **Medium Quality**
 - **High Quality**
- **Collider Scale** – the amount to scale the colliders. You can set this to less than one if for example you want the shattered pieces to overlap slightly.

If you have a lot of shattered pieces with colliders, the amount of time required to generate the shattered pieces will increase. Circle and box colliders are the quickest to generate and polygons are the slowest. Lower quality versions of any collider will also be quicker. If you have a lot of pieces, circle colliders are usually enough to create realistic shattered pieces.

You should also bear in mind that there is a run-time overhead to using colliders in the physics engine. Circles are most efficient, followed by boxes and then polygons.

Sprite Shattering Properties

When your scene is run, you can control how the shattered pieces react when the sprite is shattered. You can control the following settings:

Sprite Shattering Properties

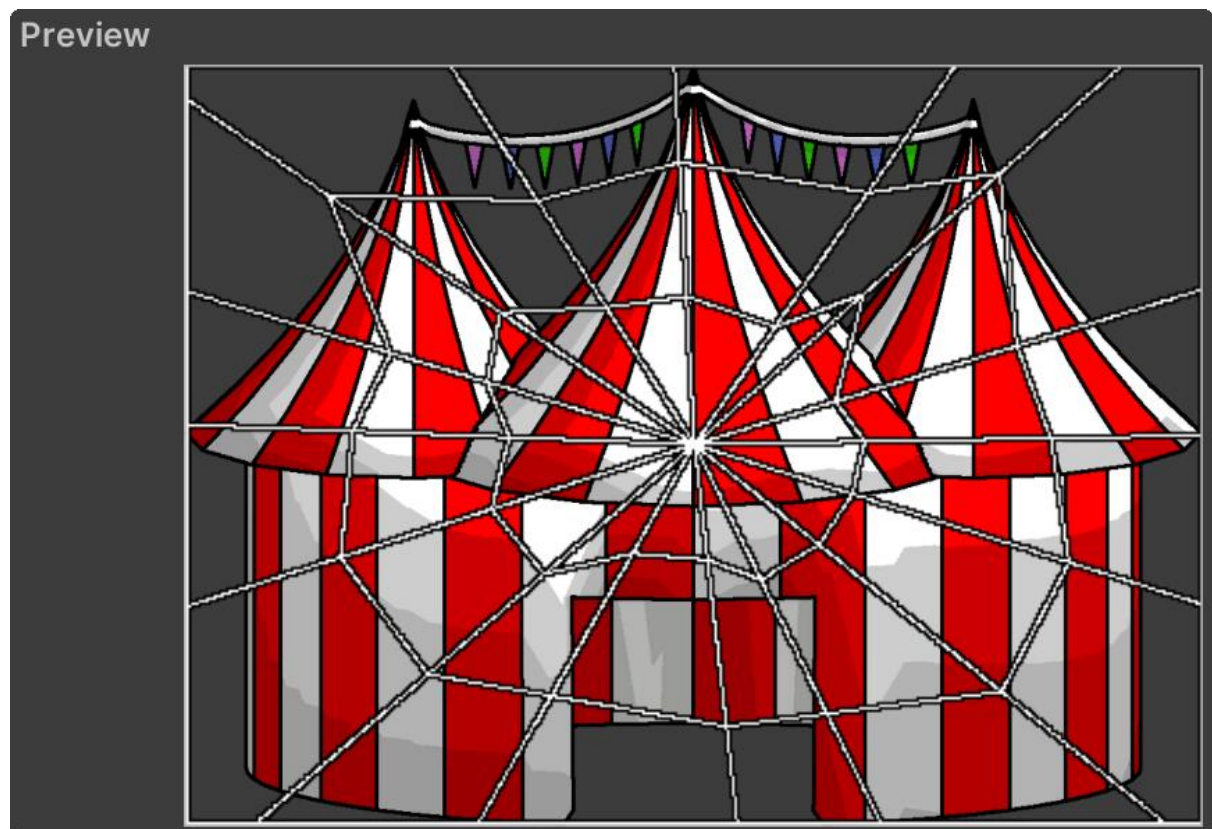
Explode From	X	0.5	Y	0.5
Explosion Force	X	0	Y	5

Sprite shattering properties

- **Explode From** – the position to explode the shattered pieces from (where in the X direction, 0 is the left of the sprite and 1 is the right, and in the Y direction 0 is the bottom of the sprite and 1 is the top). For example to explode the pieces out from the centre, set this to 0.5, 0.5.
- **Explode Force** – the force with which to explode the shattered pieces. A greater force will make them fly further away, and a force of zero will make them just fall where they are.

Preview

To see how the parameters you have set will affect how your sprite shatters, a preview is shown in the inspector showing the shape of the shattered pieces:



A preview of a radial sprite shatter

Version

The current version of Sprite Shatter is shown at the bottom of the inspector component. Click on it to view version changes.

Version 2.0.5

The Sprite Shatter version number

Shattering Sprites at Run-Time

When you are happy with the Sprite Shatter settings, it is as simple as calling **shatter()** or **reset()** at run-time on the **Shatter** component you have created. To illustrate this, the circus example calls **shatter()** if you left-click anywhere in the scene and **reset()** if you right-click.