
Options for Explode Anything

There are two major ways to change things when you use the `BSPixelTransition` class. The first is to simply change methods (*explode()*, *explodeflat()*, *explodeSin()*, *explodeSinWave()*) to achieve different kinds of explosions. The second is to modify the parameters (*duration*, *size*, *jitter*, *jitterBase*) to further manipulate these explosions.

To use the different methods, simply call them from the class:

```
BSPixelTransition.explode(myMovieClip);
```

or

```
BSPixelTransition.explodeFlat(myMovieClip);
```

etc.

To use the different parameters, bundle them in an object passed as the second parameter of the chosen exploding method:

```
BSPixelTransition.explode(myMovieClip, {size:50});
```

or

```
BSPixelTransition.explode(myMovieClip, {duration:2, jitter:22, jitterBase:-1});
```

etc.

For more complex set-ups or questions, be sure to visit the [FAQ.pdf](#) document. If other troubles occur, please leave a comment on the item's page or contact me through my FlashDen page (<http://flashden.net/user/Potku>).

BSPixelTransition Methods

Method: `explode(myDisplayObject:DisplayObject, optionalParameters:Object (optional))`

Code Example: `BSPixelTransition.explode(myMovieClip);`

Optional Parameters: `duration`, `size`, `jitter`, `jitterBase`, `onComplete`

Description: This explosion method is the most common explosion method. This is the only method that uses *jitter* and *jitterBase*, which it uses to achieve its 3D effect.

Method: `explodeFlat(myDisplayObject:DisplayObject, optionalParameters:Object (optional))`

Code Example: `BSPixelTransition.explodeFlat(myMovieClip);`

Optional Parameters: `duration`, `size`, `onComplete`

Description: This explosion method is just like the *explode()* method except it does not use *jitter* or *jitterBase* and so does not achieve the 3D effect.

Method: `explodeSin(myDisplayObject:DisplayObject, optionalParameters:Object (optional))`

Code Example: `BSPixelTransition.explodeSin(myMovieClip);`

Optional Parameters: `duration`, `size`, `onComplete`

Description: This explosion method has the interesting effect of creating swirling clumps of pixels as they explode (using *Math.sin()* as a guide).

Method: `explodeSinWave(myDisplayObject:DisplayObject, optionalParameters:Object (optional))`

Code Example: `BSPixelTransition.explodeSinWav(myMovieClip);`

Optional Parameters: `duration`, `size`, `onComplete`

Description: This explosion method has the interesting effect of creating wavy line of pixels as they explode (using *Math.sin()* as a guide).



BSPixelTransition Parameters

Parameter: duration OR dur (Optional)

Code Example: `BSPixelTransition.explode(myMovieClip, {duration:1});`

Options: 0 to approximately 1.79e+308 (Number class's MAX_VALUE. Note: You cannot write this value in scientific notation.)

Default: 1

Description: Duration is (roughly) how many seconds the explosion lasts. It is not exactly in seconds, as there are many factors that dictate how long the transition lasts, but it is close.

Parameter: size (Optional)

Code Example: `BSPixelTransition.explode(myMovieClip, {size:10});`

Options: 0, -1.5, 123, etc.(-1.79e+308 to 1.79e+308 (Number class's MAX_VALUE. Note: You cannot write this value in scientific notation.))

Default: 10

Description: This tells the explosion method how large to make the explosion, relative to the scale of the original object (Note: this becomes more complex when you introduce *jitter* in the *explode()* method. See the *jitter* parameter for more.) For example, if your exploding Sprite is 100px X 100px, having a *size* value of 3 makes it explode to 300px X 300px. Note that if you have a negative size, the DisplayObject that you explode will actually *implode*. Very cool!

Parameter: jitter (Optional)

Code Example: `BSPixelTransition.explode(myMovieClip, {jitter:1});`

Options: 0, -1.5, 123, etc.(-1.79e+308 to 1.79e+308 (Number class's MAX_VALUE. Note: You cannot write this value in scientific notation.))

Default: 1

Description: *jitter* is used (only in the *explode()* method) for making a more 3 Dimensional shape. *jitter* is a random factor added to the *size* to determine randomness in the pixel's ultimate destination. In other words, if you have an exploding Sprite that was originally 100px X 100px and its exploding *size* was 3, it would normally have the potential to just blow up to 300px in either dimension. With an added jitter of 2 (2x100 = 200px) in each direction, the potential of the size goes up to between 300px and 500px (300px + 200px) in either dimension. In this way, *size* and *jitter* both work together to determine final size.

Parameter: jitterBase (Optional)

Code Example: `BSPixelTransition.explode(myMovieClip, {jitterBase:0});`

Options: 0, -1.5, 123, etc.(-1.79e+308 to 1.79e+308 (Number class's MAX_VALUE. Note: You cannot write this value in scientific notation.))

Default: 0

Description: *jitterBase* is used in conjunction with *jitter* (only in the *explode()* method) for making a more 3 Dimensional shape. *jitter* is a random factor added to the **original** *size* to determine randomness in the pixel's ultimate destination. In other words, if you have an exploding Sprite that was originally 100px X 100px and its exploding *size* was 3, it would normally have the potential to just blow up to 300px in either dimension. With an added jitter of 2 (2x100 = 200px) in each direction, the potential of the size goes up to between 300px and 500px (300px + 200px) in either dimension. When you add *jitterBase* of -1 (-1x100px = -100px) into the mix, the potential size range goes from 200px [300px - 100px] to 400px ([300px - 100px] + 200px). In this way, *size* and *jitter* **and** *jitterBase* all work together to determine final size. It is a good idea to think of *size* as your base for the explosion, *jitter* as the variableness, and *jitterBase* as the relative adjustment to the base point. I know this is starting to sound a little involved, so if you don't get it, that's ok. Just do what I do: Mess around with the numbers until you get what you want! :)

Parameter: onComplete (Optional)

Code Example: `BSPixelTransition.explode(myMovieClip, {onComplete:exploderFinished});`

Options: Function that accepts a DisplayObject as a parameter.

Default: null

Description: When the explosion is finished, you may want to do your own garbage collection or do something else with the exploded (but now hidden) DisplayObject (Sprite, MovieClip, etc.). If that's the case, just pass function to the *onComplete* parameter and when the explosion is finished, it will call your function with the exploded DisplayObject as its single parameter. For example, if you wanted to remove an exploded named "myMovieClip" from the stage, your code might look like the following:

```
BSPixelTransition.explode(myMovieClip, {onComplete:exploderFinished});  
  
function exploderFinished(myDO:DisplayObject):void{  
    removeChild(myDO); // This would remove myMovieClip, since that is what was exploded.  
}
```

For other ways you can manipulate this class, see the [FAQ.pdf](#) file.

Thanks again for purchasing Explode Anything! :)

