

# Grid 3D



# Table of Contents

Installing the component .....	8
Quick example .....	9
Using the Component Inspector .....	14
Using Actionscript to set the properties .....	15
Instantiate the component using code .....	17
Loading content using Actionscript .....	19
Types of content .....	21
External content .....	21
Library assets .....	21
Display Objects (Movie Clip, Bitmap, Sprite) .....	22
Properties .....	25
xmlPath .....	25
items .....	25
itemsArray .....	25
itemWidth .....	26
itemHeight .....	26
currentPosition .....	26
currentPositionEasingStrength .....	26
scaleMode .....	27
direction .....	27
layers .....	27
cornerRoundness .....	28
overSkinReference .....	28
verticalAlign .....	28
horizontalAlign .....	29
horizontalDistance .....	29
verticalDistance .....	29
handCursor .....	30
smoothing .....	30
interactivelItems .....	30
border .....	30
borderColor .....	31
borderAlpha .....	31

borderThickness.....	31
preloaderType .....	32
preloaderSize.....	32
preloaderFillAlpha.....	32
preloaderLineAlpha.....	33
preloaderFillColor .....	33
preloaderLineColor .....	33
zoomItemPreloaderType.....	33
zoomItemPreloaderSize .....	34
zoomItemPreloaderFillColor.....	34
zoomItemFadeDuration .....	34
reflection .....	35
reflectionDistance.....	35
reflectionAlpha .....	35
reflectionRatio .....	35
quality.....	36
cameraZoom.....	36
cameraFocus.....	36
cameraDepth .....	37
cameraAcceleration .....	37
temporaryClipReference .....	37
temporaryClipAlpha.....	38
temporaryClipColor.....	38
skipBrokenPath.....	38
selectedItemDepth .....	39
selectedItemMoveDuration .....	39
selectedItemMoveEasing .....	39
selectEasingStrength.....	39
dragScroll.....	40
dragScrollSpeed .....	40
dragScrollEasingStrength .....	40
dragScrollReversed .....	41
scrollAmount .....	41
scrollDuration .....	41

scrollEasing.....	42
zoomDuration.....	42
zoomEasing.....	42
autoScroll.....	43
autoScrollDelay.....	43
autoScrollDirection.....	43
zoomAutoScroll.....	44
zoomAutoScrollDelay.....	44
zoomAutoScrollDirection.....	44
zoomItemScale.....	45
zoomItemScaleMode.....	45
useSmoothing.....	45
usePrecision.....	46
selectOver.....	46
zoomOver.....	46
effect.....	47
videoPlayer.....	47
livePreviewItems.....	47
numItems.....	49
selectedIndex.....	49
zoomIndex.....	49
zoomState.....	49
isScrolling.....	50
camera.....	50
Methods.....	51
addItem().....	51
addItemAt().....	51
addItems().....	51
addItemsAt().....	52
removeItem().....	52
removeItemAt().....	52
removeItemsAt().....	53
removeAllItems().....	53
replaceItem().....	53

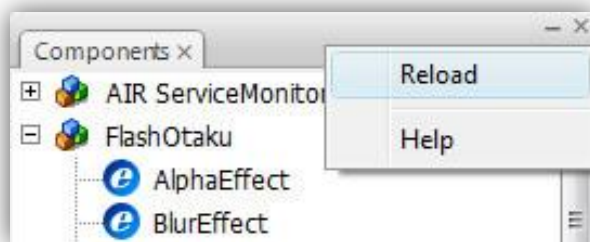
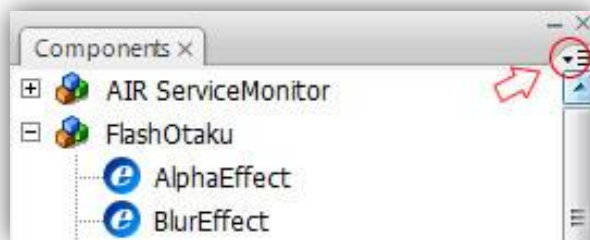
replaceItemAt()	54
spliceItems()	54
getItemAt()	55
getItemIndex()	55
select()	55
deselect()	55
selectNext()	56
selectPrevious()	56
scrollRight()	56
scrollLeft()	56
scrollUp()	57
scrollDown()	57
zoom()	57
zoomOut()	57
zoomNext()	58
zoomPrevious()	58
setSize()	58
move()	58
Events	59
ITEMS_LOAD_START	59
ITEMS_LOAD_PROGRESS	59
ITEMS_LOAD_COMPLETE	60
XML_LOAD_START	60
XML_LOAD_PROGRESS	61
XML_LOAD_COMPLETE	61
ITEM_ADD	61
ITEM_REMOVE	62
ITEM_CLICK	62
ITEM_DOUBLE_CLICK	63
ITEM_MOUSE_UP	63
ITEM_MOUSE_DOWN	64
ITEM_MOUSE_OVER	64
ITEM_MOUSE_OUT	65
ITEM_START	65

ITEM_PROGRESS.....	66
ITEM_COMPLETE .....	66
ITEM_ERROR.....	67
ZOOM_ITEM_CLICK.....	67
ZOOM_ITEM_DOUBLE_CLICK.....	68
ZOOM_ITEM_MOUSE_UP .....	68
ZOOM_ITEM_MOUSE_DOWN.....	69
ZOOM_ITEM_MOUSE_OVER .....	69
ZOOM_ITEM_MOUSE_OUT.....	70
ZOOM_ITEM_START .....	70
ZOOM_ITEM_PROGRESS.....	71
ZOOM_ITEM_COMPLETE .....	71
ZOOM_ITEM_ERROR.....	72
ITEM_SELECTED .....	72
ITEM_DESELECTED .....	73
UPDATE .....	73
SCROLL_START.....	73
SCROLL_PROGRESS .....	74
SCROLL_COMPLETE.....	74
ZOOM_START .....	75
ZOOM_PROGRESS.....	75
ZOOM_COMPLETE .....	75
ZOOM_OUT_START.....	76
ZOOM_OUT_PROGRESS .....	76
ZOOM_OUT_COMPLETE.....	76
DISPLAY_ZOOM_ITEM.....	77
OUTSIDE_CLICK.....	77
OUTSIDE_DOUBLE_CLICK .....	77
OUTSIDE_MOUSE_UP .....	78
OUTSIDE_MOUSE_DOWN .....	78
REACHED_TOP .....	79
REACHED_BOTTOM .....	79
REACHED_LEFT .....	79
REACHED_RIGHT.....	80



## Installing the component

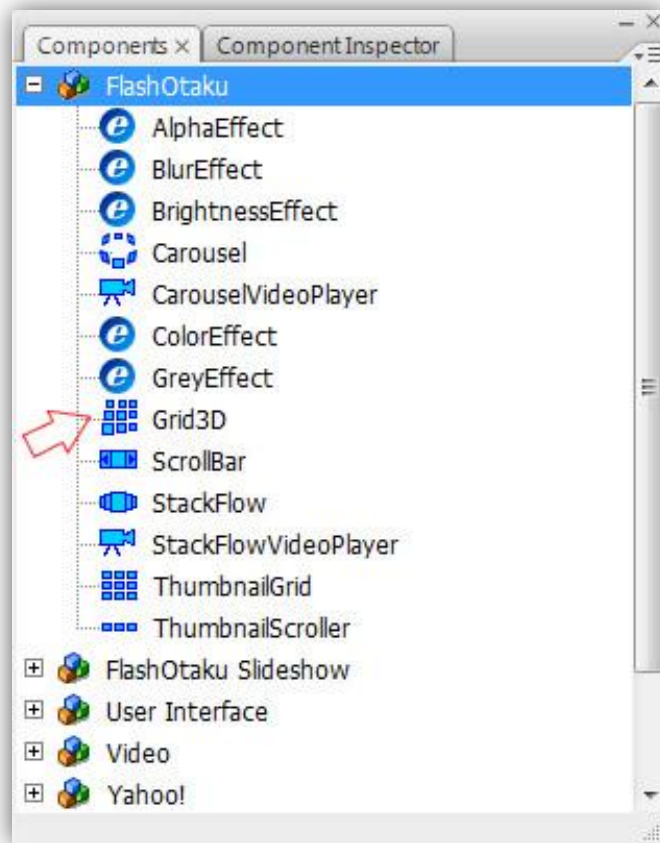
1. Double-Click on the Grid3D.mxp file.
2. Accept the license agreement.
3. If the Flash IDE was opened during installation process, reload the Components panel or restart the Flash IDE.



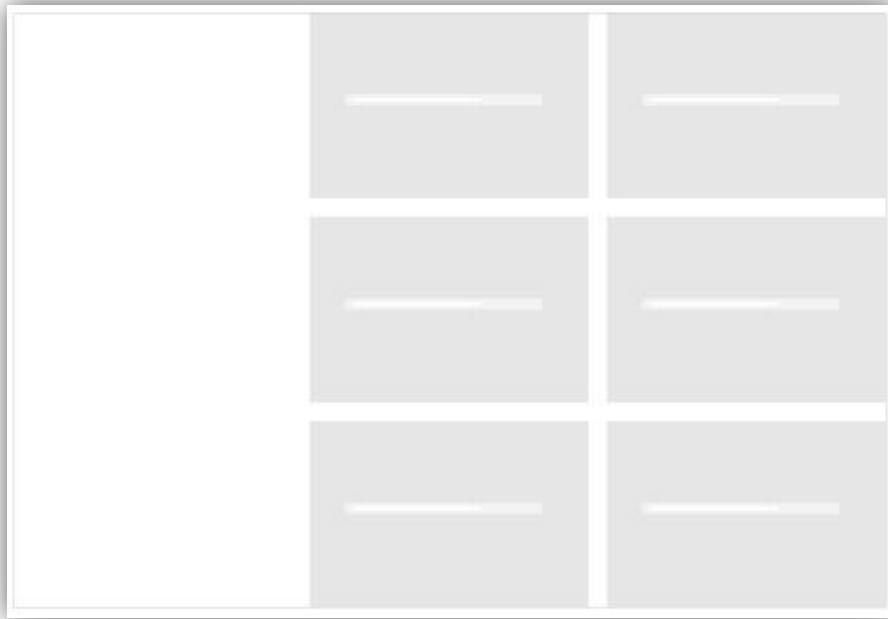
That's all. You are ready to use the component.

## Quick example

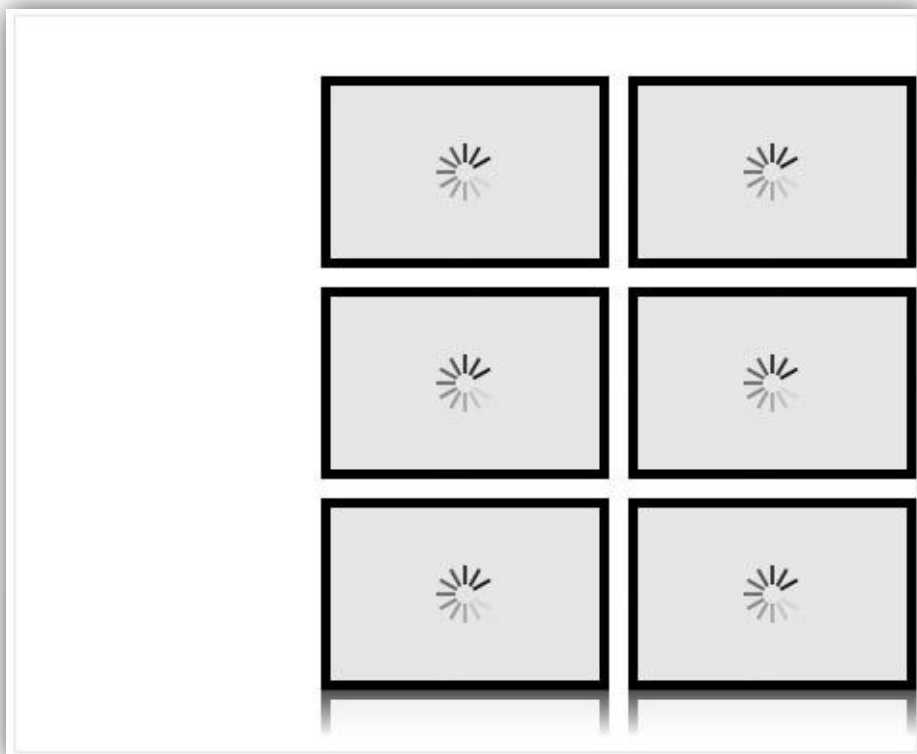
1. Create a new Actionscript 3 File and save the file as grid.fla.  
Test the movie in order to create the grid.swf file.
2. Open the Components panel (Window >Components) and drag an instance of the Grid3D component onto the stage. The Grid3D component can be found in the FlashOtaku folder.



This is how the component looks on stage:



As you can see, the component has a live preview which makes it easy for you to preview how the component will be rendered, without having to test the movie. If you chose a different preloader or change any other visual element, you will immediately be able to see those changes.



To change the color and transparency of the preview items, use the “Temporary Clip Alpha” and “Temporary Clip Color” properties. You can also use the “Live Preview Items” property to set how many items you want to use in the live preview.

Now we are ready to load some content.

3. In the same folder as grid.swf create a new folder and name it “images”. Then copy the images you want to load into this folder.
4. For this example we’ll use an XML file to specify the location of the images. Create an XML file in the same directory as grid.swf and the images folder, and name it images.xml.
5. The XML file needs to have a structure similar to this:

```
<?xml version="1.0" encoding="utf-8"?>
<images>
  <image source="images/image0.jpg" />
  <image source="images/image1.jpg"/>
  <image source="images/image2.jpg"/>
  <image source="images/image3.jpg"/>
  <image source="images/image4.jpg"/>
</images>
```

The only required keyword is “source”. You need to use this attribute to specify the location of the image you want to load.

The XML file could as well look like this:

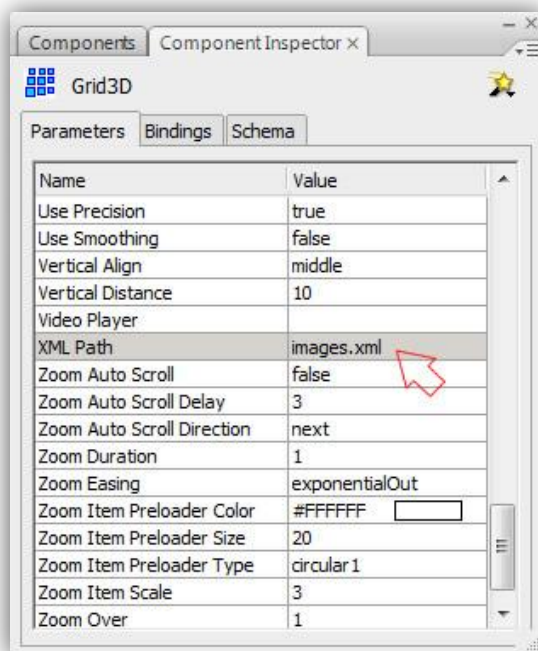
```
<?xml version="1.0" encoding="utf-8"?>
<thumbnails>
  <thumbnail source="Animals/Image_of_a_bird.jpg"/>
  <thumbnail source="Flowers/Sun_flower.jpg"/>
  <thumbnail source="WildLife /Lion.jpg"/>
  <thumbnail source="Space/Earth_from_Space.jpg"/>
</thumbnails>
```

You can also use the “zoom” attribute to specify the location of an image that will be displayed when you zoom over an item. For example you can use a lower resolution of an image to be displayed in the grid and when the user zooms in on that item, display the higher resolution of the same image. The XML file for this example would look like this:

```
<images>
  <image source="thumbnails/image0.jpg" zoom="hq_images/image0.jpg"/>
  <image source="thumbnails/image1.jpg" zoom="hq_images/image1.jpg"/>
  <image source="thumbnails/image2.jpg" zoom="hq_images/image2.jpg"/>
  <image source="thumbnails/image3.jpg" zoom="hq_images/image3.jpg"/>
  <image source="thumbnails/image4.jpg" zoom="hq_images/image4.jpg"/>
</images>
```

Another attribute that you can use is “video”. If you have the PRO version of this component, the “video” attribute allows you to specify the path of a video file or the id of a YouTube video. More information about this feature you can find in the Grid3D Video Player’s user guide.

6. Open the Component Inspector panel (Window > Component Inspector) and click on the instance you’ve dragged onto the stage. You should now see all the customizable properties in the Component Inspector. Scroll down to XML Path and enter the path of the XML file we are using, “images.xml”.

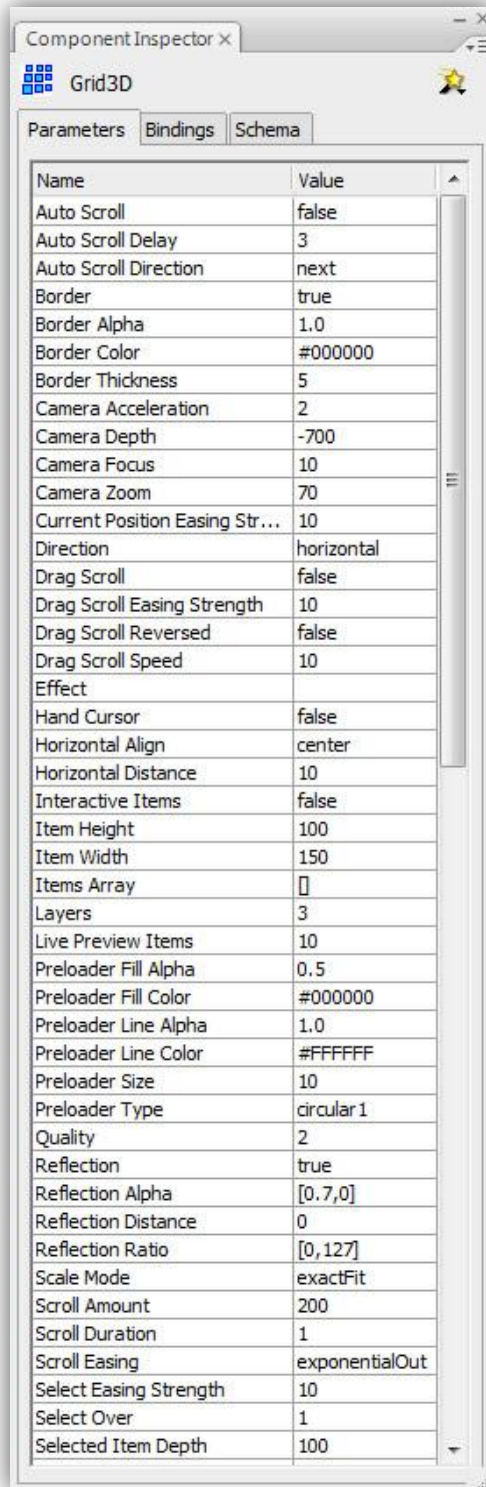


7. You are now ready to test the movie.

At this point, you have managed to load the images. In the following chapters you'll learn how to use all the available properties and methods to customize the component to your needs.

## Using the Component Inspector

The Component Inspector allows you to easily change the component's properties without using Actionscript code.



The Component Inspector has 2 columns: Name and Value.

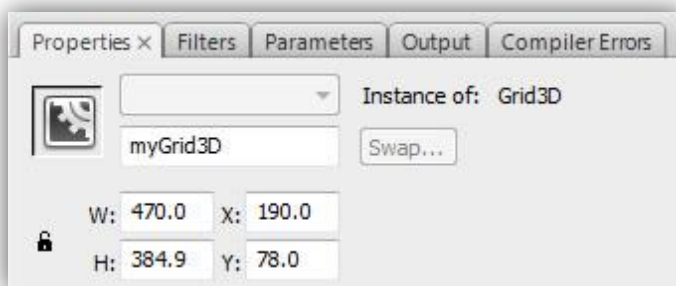
The Name column contains the name of the properties while the Value column contains the values of the properties.

All these properties can also be changed using Actionscript code, as you will see in the next chapter.

## Using Actionscript to set the properties

Setting the properties using Actionscript is very easy and can give you a greater flexibility than using the Component Inspector.

Before writing any code, you need to give the component an instance name, “myGrid3D”, for example.



It's also a good practice to create a separate layer for the Actionscript code.



Now, if you want to specify the path to the XML file using Actionscript, all the code you need to write is this:

```
myGrid3D.xmlPath = "images.xml"
```

Let's say you don't want to load the images from the beginning but only after you press a button. This is very easy to do using code.

First create a MovieClip symbol and give it an instance name of “myButton”.

Add this code in the actionscript layer:

```
import flash.events.MouseEvent;

myButton.addEventListener(MouseEvent.CLICK, clickHandler);

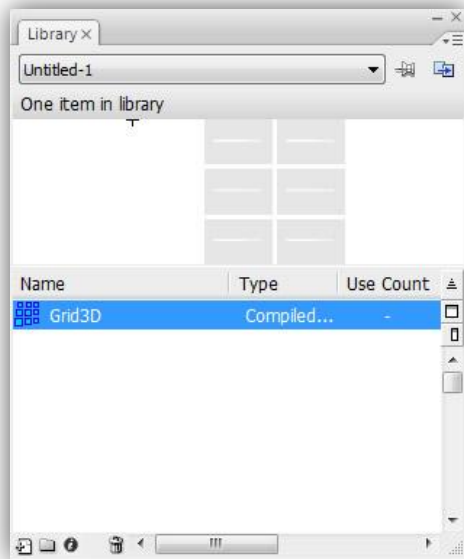
function clickHandler(event:MouseEvent):void
{
    myGrid3D.xmlPath = "images.xml";
}
```

As you can see it's very easy and gives you the flexibility you couldn't have using just the Component Inspector.

In the next chapter you will learn how to instantiate the component using code as opposed to dragging it onto the stage, which will give you an even greater flexibility.

## Instantiate the component using code

Before instantiating the component, you need to make sure the component is in the library (Window > Library). If it's not there, first you need to drag an instance of the component into the library.



Now, we are ready to start coding.

1. Import the Grid3D class.

```
import com.flashotaku.grid.Grid3D;
```

2. Instantiate the Grid3D class.

```
var myGrid3D:Grid3D = new Grid3D();
```

3. Add the instance to the display list.

```
addChild(myGrid3D);
```

These 3 lines of code are the equivalent of dragging the component onto the stage and giving it the instance name of “myGrid3D”. The advantage of using code over dragging the component onto the stage is that it allows you to choose when the component will appear on stage. Maybe you want to instantiate the component only after you press a button or some other piece of code executed.

The next example shows how to instantiate the component only after a button was pressed.

Create a MovieClip symbol and give it the instance name of “myButton”. Then, in the actionscript layer, add the following code:

```
import com.flashotaku.grid.Grid3D;
import flash.events.MouseEvent;

var myGrid3D:Grid3D;

myButton.addEventListener(MouseEvent.CLICK, clickHandler);

function clickHandler(event:MouseEvent):void
{
    myGrid3D = new Grid3D();
    addChild(myGrid3D);
    myGrid3D.xmlPath = "images.xml";
}
```

## Loading content using Actionscript

You are not required to use an XML file to load the content. You can also do this with Actionscript, using the `items` property. All you need to do is create an array of Objects, each object containing the source property and then pass this array to the component's `items` property.

```
var myItems:Array = [{source:"images/image0.jpg"},
                    {source:"images/image1.jpg"},
                    {source:"images/image2.jpg"},
                    {source:"images/image3.jpg"},
                    {source:"images/image4.jpg"}];

myGrid3D.items = myItems;
```

Each item needs to have the "source" property to specify the location of the image. You can also set additional properties for each item to store data that you might want to use in your project. For example, you could specify a title, a description or an URL for each item.

### *Example:*

```
{source:"images/image0.jpg", title:"Some title",
description:"Some description", link:"http://flashedaku.com"}
```

Note that you can add as many additional properties as you want and also the name of these properties is not important.

### *Example:*

```
{source:"images/image0.jpg", abc:"Some data", xyz:"Some more
data"}
```

You can also add additional data to each item if you use an XML file by creating additional attributes. The name and number of these additional attributes is not important.

*Example:*

```
<?xml version="1.0" encoding="utf-8"?>
<images>
  <image source="images/image0.jpg" title="Title"/>
  <image source="images/image1.jpg" abc="some data"/>
  <image source="images/image2.jpg" xyz="more data"/>
  <image source="images/image3.jpg"/>
  <image source="images/image4.jpg"/>
</images>
```

If you don't want to set any additional data you could use another property to load the content, `itemsArray`. You can pass this property an array of strings that contains the location of the content you want to load.

*Example:*

```
var myItems:Array = ["images/image0.jpg", "images/image1.jpg",
"images/image2.jpg", "images/image3.jpg", "images/image4.jpg"].
myGrid3D.itemsArray = myItems;
```

As you can see, using this property requires less code. Also, this property is available in the Component Inspector while the `items` property isn't.

# Types of content

The Grid3D component allows you to load external files like JPGs, GIFs, PNGs and SWFs but also library assets and Display Objects.

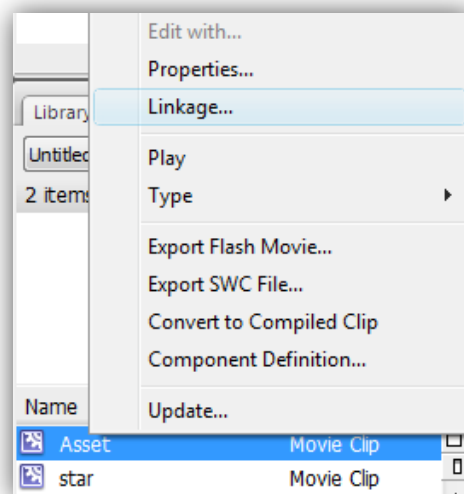
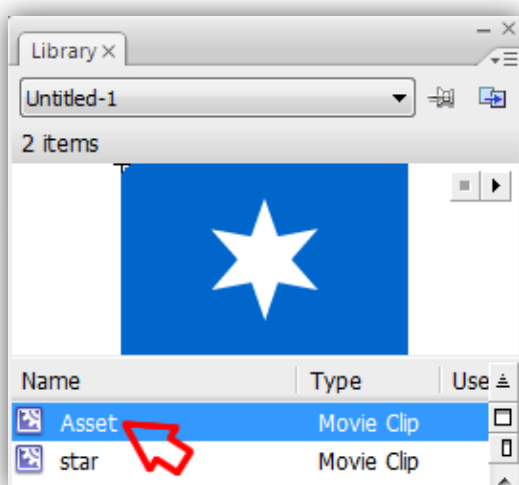
## External content

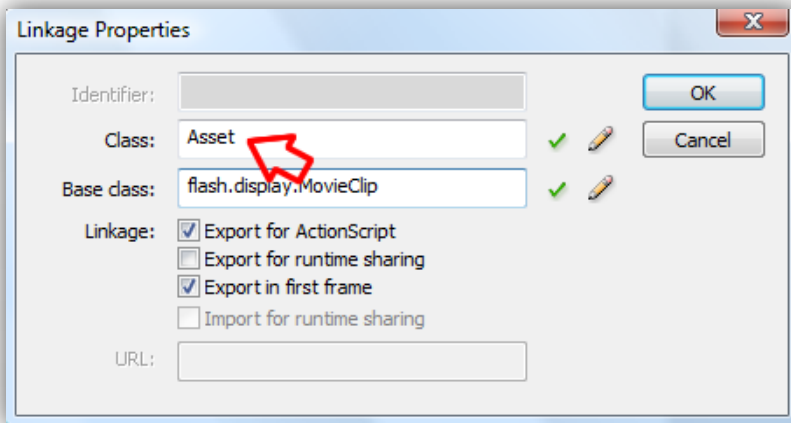
You've seen in the previous examples how to load external content. You simply need to specify the path to the content as the source.

```
myGrid3D.addItem({source: "images/MyImage.jpg"});
```

## Library assets

Before loading a symbol from the Library, you need to associate a Class to the symbol. For this, go to the Library, right-click on the symbol, chose Linkage, select Export for Actionscript, and enter a name in the Class field. Then, all you need to do is specify that Class name as the source.





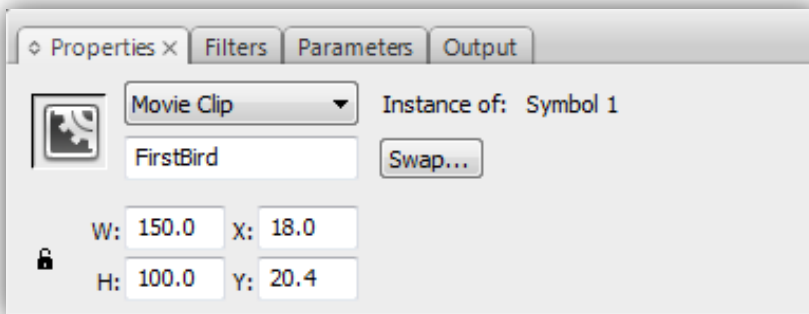
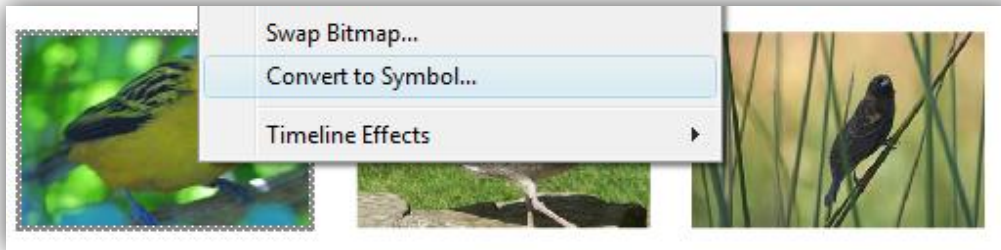
```
var myItems:Array = [{source:"Asset"},  
{source:" images/image0.jpg"}, {source:" images/image1.jpg"}]  
  
myGrid3D.items = myItems;
```

The name of the Class doesn't necessary have to be the same as the name of the symbol but when you specify the source you always use the name of the Class. I could have named the class MyAssetSample, for example, and then the source would be `{source:"MyAssetSample"}`.

You can also pass the source without quotation marks:  
`{source:MyAssetSample}`.

## Display Objects (Movie Clip, Bitmap, Sprite)

For this example I've imported 3 images (Bitmaps) to the Stage. Now, all I need to do is convert the Bitmaps to Movie Clips and give them instance names. To convert a Bitmap to Movie Clip right-click on the image and chose "Convert to Symbol". You can specify a name for the Symbol if you want (it's not necessary for this example) then click OK. Then click on the image, which is now a Movie Clip, and give it an instance name in the Properties panel.



You need to follow these steps for each image you want to load. After this you simply need to specify the instance names of the movie clips as the source.

```
var myItems:Array = [{source:"FirstBird"},  
{source: "SecondBird"}, {source: "ThirdBird"}];  
  
myGrid3D.items = myItems;
```

When you test the movie, the component will take those movie clips from the stage and load them as items.

Note that any movie clip can be loaded in a single item. If you want to load the same movie clip into multiple items, you should load it as a library asset.

It's very similar when you work with Sprite objects. You actually have two options here. One would be to specify the instance of the Sprite object as the source, without quotations.

```
var rectangle:Sprite = new Sprite();  
rectangle.graphics.beginFill(0xFF0000);  
rectangle.graphics.drawRect(0, 0, 100, 100);
```

```
rectangle.graphics.endFill();
```

```
myGrid3D.addItem({source:rectangle});
```

The other option is used when you use an XML file to load the items and you need to somehow specify the Sprite object in the XML.

```
<image source="rectangle"/>
```

In this case you need to also give a name to the Sprite object and add the object to the display list.

```
var rectangle:Sprite = new Sprite();  
rectangle.graphics.beginFill(0xFF0000);  
rectangle.graphics.drawRect(0, 0, 100, 100);  
rectangle.graphics.endFill();  
rectangle.name = "rectangle";  
addChild(rectangle);
```

The name of the instance needs to be the same as the name used in the XML file. If you name the Sprite object "myRectangle" (`rectangle.name = "myRectangle"`), in the XML file you need to have:

```
<image source="myRectangle"/>
```

# Properties

## xmlPath

*Component Inspector Name:* XML Path.

*Type:* String.

*Default Value:* "" (empty string).

*Description:* The location of the XML file.

*Example:* myGrid3D.xmlPath = "files/xml/images.xml";

## items

*Component Inspector Name:* Not Available.

*Type:* Array of objects.

*Default Value:* [] (empty array).

*Description:* An array of objects, each object containing the source of the content and other additional data.

*Example:* myGrid3D.items = [{source:"files/images/image0.jpg"},  
{source:"files/images/image1.jpg"}, {source:"files/images/image2.jpg"}];

## itemsArray

*Component Inspector Name:* Items Array.

*Type:* Array of strings.

*Default Value:* [] (empty array).

*Description:* An array of strings, each string containing the source of the content.

*Example:* myGrid3D.itemsArray = ["files/images/image0.jpg",  
"files/images/image1.jpg", "files/images/image2.jpg"];

### **itemWidth**

*Component Inspector Name:* Item Width.

*Type:* Number.

*Default Value:* 150.

*Description:* The maximum width of each item.

*Example:* `myGrid3D.itemWidth = 200;`

### **itemHeight**

*Component Inspector Name:* Item Height.

*Type:* Number.

*Default Value:* 100.

*Description:* The maximum height of each item.

*Example:* `myGrid3D.itemWidth = 150;`

### **currentPosition**

*Component Inspector Name:* Not Available.

*Type:* Number.

*Default Value:* 0.

*Description:* A number between 0 and 1 representing the current position of the grid.

*Example:* `myGrid3D.currentPosition = 0.5;`

### **currentPositionEasingStrength**

*Component Inspector Name:* Current Position Easing Strength.

*Type:* Number.

*Default Value:* 30.

*Description:* Indicates the strength of the easing when the position of the grid is set using the `currentPosition` property.

*Example:* `myGrid3D.currentPositionEasingStrength = 50;`

## scaleMode

*Component Inspector Name:* Scale Mode.

*Type:* String.

*Default Value:* "exactFit".

*Available Values:* "exactFit" and "maintainAspectRatio";

*Description:* The scaling of each item. "exactFit" will resize all items to the maximum width and height. "maintainAspectRatio" will resize the items to the maximum width or height, maintaining the aspect ratio.

*Example:* `myGrid3D.scaleMode = "maintainAspectRatio";`

## direction

*Component Inspector Name:* Direction.

*Type:* String.

*Default Value:* "horizontal".

*Available Values:* "horizontal" and "vertical".

*Description:* The direction of the grid.

*Example:* `myGrid3D.direction = "vertical";`

## layers

*Component Inspector Name:* Layers.

*Type:* uint.

*Default Value:* 3.

*Description:* The number of layers represents the number of rows when direction is set to “horizontal” or the number of columns when direction is set to “vertical”.

*Example:* `myGrid3D.layers = 4;`

### **cornerRoundness**

*Component Inspector Name:* Corner Roundness.

*Type:* Number.

*Default Value:* 0.

*Description:* The amount by which the corners of the items will be rounded.

*Example:* `myGrid3D.cornerRoundness = 50;`

### **overSkinReference**

*Component Inspector Name:* Over Skin Reference.

*Type:* String.

*Default Value:* “”.

*Description:* The name of a Class associated to a library symbol that will be displayed on top of each item. This property can be used to create a gloss effect or to add a watermark.

*Example:* `myGrid3D.overSkinReference = “Gloss”;`

### **verticalAlign**

*Component Inspector Name:* Vertical Align.

*Type:* String.

*Default Value:* “middle”.

*Available Values:* “middle”, “top” and “bottom”.

*Description:* The vertical alignment of each item when the direction is set to “horizontal”.

*Example:* `myGrid3D.verticalAlign = “bottom”;`

### **horizontalAlign**

*Component Inspector Name:* Horizontal Align.

*Type:* String.

*Default Value:* “center”.

*Available Values:* “center, “left” and “right”.

*Description:* The horizontal alignment of each item when the direction is set to “vertical”.

*Example:* `myGrid3D.horizontalAlign = “left”;`

### **horizontalDistance**

*Component Inspector Name:* Horizontal Distance.

*Type:* Number.

*Default Value:* 10.

*Description:* The horizontal distance between items.

*Example:* `myGrid3D.horizontalDistance = 25;`

### **verticalDistance**

*Component Inspector Name:* Vertical Distance.

*Type:* Number.

*Default Value:* 10.

*Description:* The vertical distance between items.

*Example:* `myGrid3D.verticalDistance = 25;`

## handCursor

*Component Inspector Name:* Hand Cursor.

*Type:* Boolean.

*Default Value:* false.

*Description:* Indicates whether a hand cursor will be displayed when the mouse rolls over an item.

*Example:* `myGrid3D.handCursor = true;`

## smoothing

*Component Inspector Name:* Smoothing.

*Type:* Boolean.

*Default Value:* true.

*Description:* Indicates whether the images will be smoothed.

*Example:* `myGrid3D.smoothing = false;`

## interactiveItems

*Component Inspector Name:* Interactive Items.

*Type:* Boolean.

*Default Value:* false.

*Description:* Indicates whether the items will be interactive. Set this property to true if the items contain objects that need to respond to mouse interaction.

*Example:* `myGrid3D.interactiveItems = true;`

## border

*Component Inspector Name:* Border.

*Type:* Boolean.

*Default Value:* false.

*Description:* Indicates whether a border will be added for each item.

*Example:* `myGrid3D.border = true;`

### **borderColor**

*Component Inspector Name:* Border Color.

*Type:* uint.

*Default Value:* 0xFFFFFFFF.

*Description:* The color of the border.

*Example:* `myGrid3D.borderColor = 0xFF00FF;`

### **borderAlpha**

*Component Inspector Name:* Border Alpha.

*Type:* Number.

*Default Value:* 1.0.

*Description:* The alpha of the border.

*Example:* `myGrid3D.borderAlpha = 0.5;`

### **borderThickness**

*Component Inspector Name:* Border Thickness.

*Type:* Number.

*Default Value:* 5.

*Description:* The thickness of the border.

*Example:* `myGrid3D.borderThickness = 0xFF00FF;`

## preloaderType

*Component Inspector Name:* Preloader Type.

*Type:* String.

*Default Value:* "bar".

*Available Values:* "bar", "pie", "circular1", "circular2", "circular3" and "none"

*Description:* The type of preloader. "none" indicates that no preloader will be used.

*Example:* `myGrid3D.preloaderType = "circular2";`

## preloaderSize

*Component Inspector Name:* Preloader Size.

*Type:* Number.

*Default Value:* 100.

*Description:* The width of the preloader if preloaderType is set to "bar" and the radius of the preloader if preloaderType is set to "pie", "circular1", "circular2" or "circular3".

*Example:* `myGrid3D.preloaderSize = 15;`

## preloaderFillAlpha

*Component Inspector Name:* Preloader Fill Alpha.

*Type:* Number.

*Default Value:* 0.5.

*Description:* The alpha of the preloader's fill portion.

*Example:* `myGrid3D.preloaderFillAlpha = 0.7;`

### **preloaderLineAlpha**

*Component Inspector Name:* Preloader Line Alpha.

*Type:* Number.

*Default Value:* 1.0.

*Description:* The alpha of the preloader's line portion.

*Example:* `myGrid3D.preloaderLineAlpha = 0.2;`

### **preloaderFillColor**

*Component Inspector Name:* Preloader Fill Color.

*Type:* uint.

*Default Value:* 0xFFFFFFFF.

*Description:* The color of the preloader's fill portion.

*Example:* `myGrid3D.preloaderFillColor = 0x0000FF;`

### **preloaderLineColor**

*Component Inspector Name:* Preloader Line Color.

*Type:* uint.

*Default Value:* 0xFFFFFFFF.

*Description:* The color of the preloader's line portion.

*Example:* `myGrid3D.preloaderLineColor = 0xFF0000;`

### **zoomItemPreloaderType**

*Component Inspector Name:* Zoom Item Preloader Type.

*Type:* String.

*Default Value:* "circular1".

*Available Values:* “circular1”, “circular2”, “circular3” and “none”

*Description:* The type of preloader. “none” indicates that no preloader will be used.

*Example:* `myGrid3D.zoomItemPreloaderType = “circular2”;`

### **zoomItemPreloaderSize**

*Component Inspector Name:* Zoom Item Preloader Size.

*Type:* Number.

*Default Value:* 20.

*Description:* The radius of the preloader.

*Example:* `myGrid3D.zoomItemPreloaderSize =15;`

### **zoomItemPreloaderFillColor**

*Component Inspector Name:* Zoom Item Preloader Fill Color.

*Type:* uint.

*Default Value:* 0xFFFFFFFF.

*Description:* The color of the preloader.

*Example:* `myGrid3D.zoomItemPreloaderFillColor = 0x0000FF;`

### **zoomItemFadeDuration**

*Component Inspector Name:* Zoom Item Fade Duration.

*Type:* Number.

*Default Value:* 0.5.

*Description:* The duration, in seconds, for the zoomed item to fade in.

*Example:* `myGrid3D.zoomItemFadeDuration = 1;`

## reflection

*Component Inspector Name:* Reflection.

*Type:* Boolean.

*Default Value:* false.

*Description:* Indicates whether a reflection will be added for the items.

*Example:* myGrid3D.reflection = true;

## reflectionDistance

*Component Inspector Name:* Reflection Distance.

*Type:* Number.

*Default Value:* 0.

*Description:* The distance of the reflection from the bottom edge of the item.

*Example:* myGrid3D.reflectionDistance = 5;

## reflectionAlpha

*Component Inspector Name:* Reflection Alpha.

*Type:* Array of numbers.

*Default Value:* [0.7, 0].

*Description:* An array of two numbers representing the alpha values at the beginning and at the end of the reflection.

*Example:* myGrid3D.reflectionAlpha = [1.0, 0.0];

## reflectionRatio

*Component Inspector Name:* Reflection Ratio.

*Type:* Array of numbers.

*Default Value:* [0, 127].

*Description:* An array of two numbers containing the ratio values of the reflection.

*Example:* myGrid3D.reflectionRatio = [0, 255];

### **quality**

*Component Inspector Name:* Quality.

*Type:* uint.

*Default Value:* 2.

*Description:* A number representing the quality of the perspective view for the items. Setting this property to higher values will decrease the performance.

*Example:* myGrid3D.quality = 1;

### **cameraZoom**

*Component Inspector Name:* Camera Zoom.

*Type:* Number.

*Default Value:* 70.

*Description:* A number representing the zoom value of the 3D Camera.

*Example:* myGrid3D.cameraZoom = 40;

### **cameraFocus**

*Component Inspector Name:* Camera Focus.

*Type:* Number.

*Default Value:* 10.

*Description:* A number representing the focus value of the 3D Camera.

*Example:* myGrid3D.cameraFocus = 15;

### **cameraDepth**

*Component Inspector Name:* Camera Depth.

*Type:* Number.

*Default Value:* -700.

*Description:* A number representing the position, on the Z axis, of the 3D Camera.

*Example:* myGrid3D.cameraDepth = -800;

### **cameraAcceleration**

*Component Inspector Name:* Camera Acceleration.

*Type:* Number.

*Default Value:* 2.

*Description:* A number representing the acceleration of the 3D Camera when the grid is moving.

*Example:* myGrid3D.cameraAcceleration = 3;

### **temporaryClipReference**

*Component Inspector Name:* Temporary Clip Reference.

*Type:* Object.

*Default Value:* null.

*Description:* The name of a Class associated to a library symbol that will be displayed for each item until the content of the item is loaded. If no value is specified, automatically a rectangle will be used with a specified alpha and color. If you don't want a temporary clip to be displayed at all, simply set the temporaryClipAlpha property to 0.

*Example:* myGrid3D.temporarilyClipReference = "TempClip";

### **temporarilyClipAlpha**

*Component Inspector Name:* Temporary Clip Alpha.

*Type:* Number.

*Default Value:* 0.5.

*Description:* The alpha of the temporary clip. This property will be ignored if a reference to a temporary clip is set.

*Example:* myGrid3D.temporarilyClipAlpha = 1.0;

### **temporarilyClipColor**

*Component Inspector Name:* Temporary Clip Color.

*Type:* uint.

*Default Value:* 0xCCCCCC.

*Description:* The color of the temporary clip. This property will be ignored if a reference to a temporary clip is set.

*Example:* myGrid3D.temporarilyClipColor = 0x000000;

### **skipBrokenPath**

*Component Inspector Name:* Skip Broken Path.

*Type:* Boolean.

*Default Value:* false.

*Description:* Indicates whether the component will ignore invalid sources when external content is loaded.

*Example:* myGrid3D.skipBrokenPath = true;

### **selectedItemDepth**

*Component Inspector Name:* Selected Item Depth.

*Type:* Number.

*Default Value:* 100.

*Description:* Represents the position, on the Z axis, to which the item will move when it is selected.

*Example:* `myGrid3D.selectedItemDepth = 200;`

### **selectedItemMoveDuration**

*Component Inspector Name:* Selected Item Move Duration.

*Type:* Number.

*Default Value:* 1.

*Description:* Represents the duration of the movement when an item is selected.

*Example:* `myGrid3D.selectedItemMoveDuration = 0.4;`

### **selectedItemMoveEasing**

*Component Inspector Name:* Selected Item Move Easing.

*Type:* String.

*Default Value:* "exponentialOut".

*Description:* Represents the easing of the movement when an item is selected.

*Example:* `myGrid3D.selectedItemMoveEasing = "linear";`

### **selectEasingStrength**

*Component Inspector Name:* Select Easing Strength.

*Type:* Number.

*Default Value:* 10.

*Description:* Represents the strength of the easing when an item is selected.

*Example:* `myGrid3D.selectEasingStrength = 20;`

### **dragScroll**

*Component Inspector Name:* Drag Scroll.

*Type:* Boolean.

*Default Value:* false.

*Description:* Indicates whether scrolling the component's items by mouse dragging is activated.

*Example:* `myGrid3D.dragScroll = true;`

### **dragScrollSpeed**

*Component Inspector Name:* Drag Scroll Speed.

*Type:* Number.

*Default Value:* 30.

*Description:* The speed of the scrolling when `dragScroll` is set to true. A higher value indicates a higher speed.

*Example:* `myGrid3D.dragScrollSpeed = 30;`

### **dragScrollEasingStrength**

*Component Inspector Name:* Drag Scroll Easing Strength.

*Type:* Number.

*Default Value:* 10.

*Description:* The strength of the easing when dragScroll is set to true. A lower value indicates a stronger easing.

*Example:* myGrid3D.dragScrollEasingStrength = 30;

### **dragScrollReversed**

*Component Inspector Name:* Drag Scroll Reversed.

*Type:* Boolean.

*Default Value:* false.

*Description:* Indicates whether the items are scrolling in reversed direction relative to the mouse movement.

*Example:* myGrid3D.dragScrollReversed = true;

### **scrollAmount**

*Component Inspector Name:* Scroll Amount.

*Type:* Number.

*Default Value:* 200.

*Description:* The amount to scroll when using scrollLeft(), scrollRight(), scrollUp() and scrollDown().

*Example:* myGrid3D.scrollAmount = 100;

### **scrollDuration**

*Component Inspector Name:* Scroll Duration.

*Type:* Number.

*Default Value:* 1.

*Description:* The scroll duration in seconds.

*Example:* myGrid3D.scrollDuration = 3;

## scrollEasing

*Component Inspector Name:* Scroll Easing.

*Type:* String.

*Default Value:* "exponentialOut".

*Available values:* "linear", "backIn", "backOut", "backInOut", "bounceIn", "bounceOut", "bounceInOut", "circularIn", "circularOut", "circularInOut", "cubicIn", "cubicOut", "cubicOut", "elasticIn", "elasticOut", "elasticInOut", "exponentialIn", "exponentialOut", "exponentialInOut", "quadraticIn", "quadraticOut", "quadraticInOut", "quarticIn", "quarticOut", "quarticInOut", "quinticIn", "quinticOut", "quinticInOut", "sineIn", "sineOut", "sineInOut"

*Description:* The scroll easing.

*Example:* myGrid3D.scrollEasing = "backOut";

## zoomDuration

*Component Inspector Name:* Zoom Duration.

*Type:* Number.

*Default Value:* 1.

*Description:* The zoom duration in seconds.

*Example:* myGrid3D.zoomDuration = 3;

## zoomEasing

*Component Inspector Name:* Zoom Easing.

*Type:* String.

*Default Value:* "exponentialOut".

*Available values:* "linear", "backIn", "backOut", "backInOut", "bounceIn", "bounceOut", "bounceInOut", "circularIn", "circularOut", "circularInOut", "cubicIn", "cubicOut", "cubicOut", "elasticIn", "elasticOut", "elasticInOut", "exponentialIn", "exponentialOut", "exponentialInOut", "quadraticIn",

“quadraticOut”, “quadraticInOut”, “quarticIn”, “quarticOut”, “quarticInOut”, “quinticIn”, “quinticOut”, “quinticInOut”, “sineIn”, “sineOut”, “sineInOut”

*Description:* The zoom easing.

*Example:* `myGrid3D.zoomEasing = “backOut”;`

### **autoScroll**

*Component Inspector Name:* Auto Scroll.

*Type:* Boolean.

*Default Value:* false.

*Description:* Indicates whether the component’s items will be scrolled automatically.

*Example:* `myGrid3D.autoScroll = true;`

### **autoScrollDelay**

*Component Inspector Name:* Auto Scroll Delay.

*Type:* Number.

*Default Value:* 3.

*Description:* Indicates the delay of the auto scroll, in seconds.

*Example:* `myGrid3D.autoScrollDelay = 4;`

### **autoScrollDirection**

*Component Inspector Name:* Auto Scroll Direction.

*Type:* String.

*Default Value:* “next”.

*Available Values:* “next” and “previous”.

*Description:* Indicates whether the next item or the previous item will be selected.

*Example:* `myGrid3D.autoScrollDirection = "previous";`

### **zoomAutoScroll**

*Component Inspector Name:* Zoom Auto Scroll.

*Type:* Boolean.

*Default Value:* false.

*Description:* Indicates whether the component's items will be zoomed automatically.

*Example:* `myGrid3D.zoomAutoScroll = true;`

### **zoomAutoScrollDelay**

*Component Inspector Name:* Zoom Auto Scroll Delay.

*Type:* Number.

*Default Value:* 3.

*Description:* Indicates the delay of the zoom auto scroll, in seconds.

*Example:* `myGrid3D.zoomAutoScrollDelay = 4;`

### **zoomAutoScrollDirection**

*Component Inspector Name:* Zoom Auto Scroll Direction.

*Type:* String.

*Default Value:* "next".

*Available Values:* "next" and "previous".

*Description:* Indicates whether the next item or the previous item will be zoomed.

*Example:* myGrid3D.zoomAutoScrollDirection = "previous";

### **zoomItemScale**

*Component Inspector Name:* Zoom Item Scale.

*Type:* Number.

*Default Value:* 3.

*Description:* Indicates how much the items will be scaled when zooming is used.

*Example:* myGrid3D.zoomItemScale = 2;

### **zoomItemScaleMode**

*Component Inspector Name:* Zoom Item Scale Mode.

*Type:* String.

*Default Value:* "exactFit".

*Available Values:* "exactFit" and "maintainAspectRatio";

*Description:* The scaling of the zoomed item. "exactFit" will resize the item to the maximum width and height. "maintainAspectRatio" will resize the item to the maximum width or height, maintaining the aspect ratio.

*Example:* myGrid3D.zoomItemScaleMode = "maintainAspectRatio";

### **useSmoothing**

*Component Inspector Name:* Use Smoothing.

*Type:* Boolean.

*Default Value:* false.

*Description:* Indicates whether the images will be smoothed during the scrolling process. Leave this property to false for a smoother scrolling. When scrolling is complete, the images will automatically be smoothed.

*Example:* `myGrid3D.useSmoothing = true;`

### **usePrecision**

*Component Inspector Name:* Use Precision.

*Type:* Boolean.

*Default Value:* true.

*Description:* Indicates whether the 3D perspective of the images will be precise during the scrolling process. Set this property to false for a smoother scrolling. If set to false, the images will look a little distorted. When the scrolling is complete, the 3D perspective will automatically become precise.

*Example:* `myGrid3D.usePrecision = false;`

### **selectOver**

*Component Inspector Name:* Select Over.

*Type:* uint.

*Default Value:* 1.

*Description:* Indicates over how many items the scrolling will be executed when `selectNext()` and `selectPrevious()` methods are used.

*Example:* `myGrid3D.selectOver = 3;`

### **zoomOver**

*Component Inspector Name:* Zoom Over.

*Type:* uint.

*Default Value:* 1.

*Description:* Indicates over how many items the zooming will be executed when `zoomNext()` and `zoomPrevious()` methods are used.

*Example:* `myGrid3D.zoomOver = 3;`

## effect

*Component Inspector Name:* Effect.

*Type:* Object.

*Default Value:* null.

*Description:* An instance of an Effect component.

*Example:*

```
var greyEffect:GreyEffect = new GreyEffect();  
myGrid3D.effect = greyEffect; or myGrid3D.effect = "greyEffect";
```

## videoPlayer

*Component Inspector Name:* Video Player.

*Type:* Object.

*Default Value:* null.

*Description:* An instance of a Grid3DVideoPlayer component.

*Example:*

```
var vp:Grid3DVideoPlayer = new Grid3DVideoPlayer();  
myGrid3D.videoPlayer = vp;
```

## livePreviewItems

*Component Inspector Name:* Live Preview Items.

*Type:* uint.

*Default Value:* 10.

*Description:* The number of items used for the live preview.



## *Read-only properties*

### **numItems**

*Type:* uint.

*Description:* Returns the total number of items.

*Example:*

```
trace(myGrid3D.numItems);
```

### **selectedIndex**

*Type:* int.

*Description:* Returns the index of the selected item.

*Example:*

```
trace(myGrid3D.selectedIndex);
```

### **zoomIndex**

*Type:* int.

*Description:* Returns the index of the zoomed item.

*Example:*

```
trace(myGrid3D.zoomIndex);
```

### **zoomState**

*Type:* String.

*Description:* Returns “in” if the items are zoomed in. Else it returns “out”.

*Example:*

```
trace(myGrid3D.zoomState);
```

## isScrolling

*Type:* Boolean.

*Description:* Indicates whether the component is currently scrolling.

*Example:*

```
trace(myGrid3D.isScrolling);
```

## camera

*Type:* Camera3D.

*Description:* This is a reference to the instance of the Camera3D class.

Using this reference to the Camera3D instance you can manipulate the position of the camera, zoom, focus and other properties.

More information about this class at

<http://docs.pv3d.org/org/papervision3d/cameras/Camera3D.html>

*Example:*

```
myGrid3D.camera.zoom = 100;
```

```
myGrid3D.camera.focus = 30;
```

```
myGrid3D.camera.y = 200;
```

# Methods

## **addItem()**

*Implementation:* addItem(data:Object):void

*Description:* Adds a new item at the end of the component.

*Parameters:* data - An object containing the item's data.

*Example:*

```
myGrid3D.addItem({source:"images/image1.jpg", title:"Flower",  
description:"Image description"});
```

## **addItemAt()**

*Implementation:* addItemAt(data:Object, index:uint):void

*Description:* Adds a new item at the specified index.

*Parameters:* data - An object containing the item's data.

index - The index at which the item is to be added.

*Example:*

```
myGrid3D.addItemAt({source:"images/image1.jpg",  
title:"Flower", description:"Image description"}, 3);
```

## **addItems()**

*Implementation:* addItems(items:Array):void

*Description:* Adds an array of items to the end of the component.

*Parameters:* items - An array of items to be added.

*Example:*

```
var myItems:Array = [{source:"images/image0.jpg"},
```

```
        {source:"images/image1.jpg",title:"Bird"},
        {source:"images/image2.jpg",title:"Sky"},
        {source:"images/image3.jpg"}];

myGrid3D.addItem(myItems);
```

## **addItemAt()**

**Implementation:** addItem(items:Array, index:uint):void

**Description:** Adds an array of items at the specified index.

**Parameters:** items - An array of items to be added.

index - The index at which the items will be added.

**Example:**

```
var myItems:Array = [{source:"images/image0.jpg"},
                    {source:"images/image1.jpg",title:"Bird"},
                    {source:"images/image2.jpg",title:"Sky"},
                    {source:"images/image3.jpg"}];

myGrid3D.addItem(myItems, 5);
```

## **removeItem()**

**Implementation:** removeItem(item:IDisplayItem):void

**Description:** Removes the specified item.

**Parameters:** item - The item to be removed.

**Example:**

```
myGrid3D.removeItem(myGrid3D.getItemAt(4));
```

## **removeItemAt()**

**Implementation:** removeItemAt(index:uint):void

*Description:* Removes the item at the specified index.

*Parameters:* index - The index of the item to be removed.

*Example:*

```
myGrid3D.removeItemAt(1);
```

### **removeItemsAt()**

*Implementation:* `removeItemsAt(startIndex:uint, removeCount:uint)`

*Description:* Removes a specified number of items starting at a specified index.

*Parameters:* startIndex - The index of the first item which is to be removed.

removeCount - The number of items to be removed.

*Example:*

```
myGrid3D.removeItemsAt(0, 3);
```

### **removeAllItems()**

*Implementation:* `removeAllItems():void`

*Description:* Removes all items.

*Example:*

```
myGrid3D.removeAllItems();
```

### **replaceItem()**

*Implementation:* `replaceItem(data:Object, item:IDisplayItem)`

*Description:* Replace an existing item with a new item.

*Parameters:* data - The data of the new item.

item - The item to be replaced.

*Example:*

```
var data:Object = {source:"images/image1.jpg", desc:"Bird"};
```

```
myGrid3D.replaceItem(data, myGrid3D.getItemAt(5));
```

## **replaceItemAt()**

**Implementation:** `replaceItemAt(data:Object, index:uint)`

**Description:** Replace an existing item at a specified index with a new item.

**Parameters:** data - The data of the new item.

index - The index of the item to be replaced.

**Example:**

```
var data:Object = {source:"images/image1.jpg", desc:"Bird"};
myGrid3D.replaceItem(data, 2);
```

## **spliceItems()**

**Implementation:**

```
spliceItems(startIndex:uint, removeCount:uint, items:Array = null):void
```

**Description:** Removes a specified number of items, starting at a specified index and adds an array of items at the specified index.

**Parameters:** startIndex - The index of the first element which is to be removed.

removeCount - The number of elements to be removed.

items - The items to be added.

**Example:**

```
var newItems:Array = [{source:"images/image0.jpg"},
                      {source:"images/image1.jpg",title:"Bird"},
                      {source:"images/image2.jpg",title:"Sky"},
                      {source:"images/image3.jpg"}];
myGrid3D.spliceItems(0, 3, newItems);
```

## getItemAt()

*Implementation:* getItemAt(index:uint) : IDisplayItem

*Description:* Returns the item at the specified index.

*Parameters:* index - The index of the item to be returned.

*Example:*

```
trace(myGrid3D.getItemAt(2).index); // output 2
```

## getItemIndex()

*Implementation:* getItemIndex(item:IDisplayItem) : int

*Description:* Returns the index of the specified item.

*Parameters:* item - The item to be located.

*Example:*

```
var nr:int = myGrid3D.getItemIndex(myGrid3D.getItemAt(7));  
trace(nr); // output 7
```

## select()

*Implementation:* select(index:uint) : void

*Description:* Selects the item at the specified index.

*Parameter:* index – The index of the item to be selected.

*Example:*

```
myGrid3D.select(5);
```

## deselect()

*Implementation:* deselect(index:uint) : void

*Description:* Deselects the item at the specified index.

*Parameter:* index – The index of the item to be deselected.

*Example:*

```
myGrid3D.deselect (myGrid3D.selectedIndex) ;
```

### **selectNext()**

*Implementation:* `selectNext () :void`

*Description:* Selects the next item.

*Example:*

```
myGrid3D.selectNext () ;
```

### **selectPrevious()**

*Implementation:* `selectPrevious () :void`

*Description:* Selects the previous item.

*Example:*

```
myGrid3D.selectPrevious () ;
```

### **scrollRight()**

*Implementation:* `scrollRight () :void`

*Description:* Scrolls the component to the right by a certain amount, given by the `scrollAmount` property.

*Example:*

```
myGrid3D.scrollRight () ;
```

### **scrollLeft()**

*Implementation:* `scrollLeft () :void`

*Description:* Scrolls the component to the left by a certain amount, given by the `scrollAmount` property.

*Example:*

```
myGrid3D.scrollLeft();
```

### **scrollUp()**

*Implementation:* scrollUp():void

*Description:* Scrolls the component to the top by a certain amount, given by the scrollAmount property.

*Example:*

```
myGrid3D.scrollUp();
```

### **scrollDown()**

*Implementation:* scrollDown():void

*Description:* Scrolls the component to the bottom by a certain amount, given by the scrollAmount property.

*Example:*

```
myGrid3D.scrollDown();
```

### **zoom()**

*Implementation:* zoom(index:uint):void

*Description:* Zooms the item at the specified index.

*Parameter:* index – The index of the item to be zoomed.

*Example:*

```
myGrid3D.zoom(5);
```

### **zoomOut()**

*Implementation:* zoomOut():void

*Description:* Zooms out the grid.

*Example:*

```
myGrid3D.zoomOut();
```

### **zoomNext()**

*Implementation:* zoomNext():void

*Description:* Zooms the next item.

*Example:*

```
myGrid3D.zoomNext();
```

### **zoomPrevious()**

*Implementation:* zoomPrevious():void

*Description:* Zooms the previous item.

*Example:*

```
myGrid3D.zoomPrevious();
```

### **setSize()**

*Implementation:* setSize(width:Number, height:Number):void

*Description:* Sets the component's size at the specified width and height.

*Example:*

```
myGrid3D.setSize(700, 500);
```

### **move()**

*Implementation:* move(x:Number, y:Number):void

*Description:* Moves the component at the specified position.

*Example:*

```
myGrid3D.move(100, 100);
```

## Events

### ITEMS\_LOAD\_START

*Description:* Dispatched when the loading of a group of items begins.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;

myGrid3D.addEventListener(Grid3DEvent.ITEMS_LOAD_START, eventHandler)

function eventHandler(event:Grid3DEvent):void
{
    trace("loading begins");
}
```

### ITEMS\_LOAD\_PROGRESS

*Description:* Dispatched every time an items was loaded, during the loading of a group of items.

*Special Properties:* itemsLoaded – The number of loaded items.

itemsTotal – The total number of items to be loaded.

**Example:**

```
import com.flashotaku.events.Grid3DEvent;

myGrid3D.addEventListener(Grid3DEvent.ITEMS_LOAD_PROGRESS, eventHandler)

function eventHandler(event:Grid3DEvent):void
{
    trace("Loaded " + event.itemsLoaded + " out of " + event.itemsTotal);
}
```

### ITEMS\_LOAD\_COMPLETE

**Description:** Dispatched when the loading of a group of items is complete.

**Example:**

```
import com.flashotaku.events.Grid3DEvent;

myGrid3D.addEventListener(Grid3DEvent.ITEMS_LOAD_COMPLETE, eventHandler)

function eventHandler(event:Grid3DEvent):void
{
    trace("loading complete");
}
```

### XML\_LOAD\_START

**Description:** Dispatched when the loading of the XML file begins.

**Example:**

```
import com.flashotaku.events.Grid3DEvent;

myGrid3D.addEventListener(Grid3DEvent.XML_LOAD_START, eventHandler);

function eventHandler(event:Grid3DEvent):void
{
    trace("XML loading begins");
}
```

## XML\_LOAD\_PROGRESS

*Description:* Dispatched during the loading of the XML file.

*Special Properties:* bytesLoaded – The number of bytes loaded thus far.

bytesTotal – The total number of bytes to be loaded.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.XML_LOAD_PROGRESS, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace("Loaded " + event.bytesLoaded + " out of " + event.bytesTotal);
}
```

## XML\_LOAD\_COMPLETE

*Description:* Dispatched when the loading of an XML file is complete.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.XML_LOAD_COMPLETE, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace("XML file loaded");
}
```

## ITEM\_ADD

*Description:* Dispatched when a new item was added.

*Special Properties:* item – The item that was added.

index – The index of the item.

data – The data of the item.

*Example:*

```

import com.flashotaku.events.Grid3DEvent;

myGrid3D.addEventListener(Grid3DEvent.ITEM_ADD, eventHandler);

function eventHandler(event:Grid3DEvent):void
{
    trace(event.index); // output 3

    trace(event.data); // output {source:"images/image1.jpg", title:"Flower",
                                desc:"Sun flower"}

    trace(event.data.title); // output "Flower"
}

myGrid3D.addItemAt({source:"images/image1.jpg", title:"Flower", desc:"Sun
flower"}, 3);

```

## ITEM\_REMOVE

*Description:* Dispatched when an item was removed.

*Special Properties:* item – The item that was removed.

index – The index of the item.

data – The data of the item.

*Example:*

```

import com.flashotaku.events.Grid3DEvent;

myGrid3D.addEventListener(Grid3DEvent.ITEM_REMOVE, eventHandler);

function eventHandler(event:Grid3DEvent):void
{
    trace(event.index); // output 1
}

myGrid3D.removeItemAt(1);

```

## ITEM\_CLICK

*Description:* Dispatched when an item was clicked.

*Special Properties:* item – The item that was clicked.

index – The index of the item.

data – The data of the item.

**Example:**

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.ITEM_CLICK, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace(event.index);
}
```

### **ITEM\_DOUBLE\_CLICK**

*Description:* Dispatched when an item was double-clicked.

*Special Properties:* item – The item that was double-clicked.

index – The index of the item.

data – The data of the item.

**Example:**

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.ITEM_DOUBLE_CLICK, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace(event.data);
}
```

### **ITEM\_MOUSE\_UP**

*Description:* Dispatched when the mouse is released over an item.

*Special Properties:* item – The item under the mouse pointer.

index – The index of the item.

**data** – The data of the item.

**Example:**

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.ITEM_MOUSE_UP, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace(event.index);
}
```

### **ITEM\_MOUSE\_DOWN**

**Description:** Dispatched when the mouse is pressed over an item.

**Special Properties:** **item** – The item under the mouse pointer.

**index** – The index of the item.

**data** – The data of the item.

**Example:**

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.ITEM_MOUSE_DOWN, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace(event.index);
}
```

### **ITEM\_MOUSE\_OVER**

**Description:** Dispatched when the mouse pointer is moved over an item.

**Special Properties:** **item** – The item under the mouse pointer.

**index** – The index of the item.

**data** – The data of the item.

### *Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.ITEM_MOUSE_OVER, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace(event.index);
}
```

## **ITEM\_MOUSE\_OUT**

*Description:* Dispatched when the mouse pointer is moved away from an item.

*Special Properties:* item – The item under the mouse pointer.

index – The index of the item.

data – The data of the item.

### *Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.ITEM_MOUSE_OUT, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace(event.index);
}
```

## **ITEM\_START**

*Description:* Dispatched when the content of an item begins to load.

*Special Properties:* item – The item that begins to load its content.

index – The index of the item.

data – The data of the item.

### *Example:*

```
import com.flashotaku.events.Grid3DEvent;
```

```
myGrid3D.addEventListener(Grid3DEvent.ITEM_START, eventHandler);

function eventHandler(event:Grid3DEvent):void
{
    trace("Item at index: " + event.index + " has started loading");
}
```

## ITEM\_PROGRESS

*Description:* Dispatched during the loading of an item's content.

*Special Properties:* item – The item that is loading its content.

index – The index of the item.

data – The data of the item.

bytesLoaded – The number of bytes loaded thus far.

bytesTotal – The number of bytes to be loaded.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;

myGrid3D.addEventListener(Grid3DEvent.ITEM_PROGRESS, eventHandler);

function eventHandler(event:Grid3DEvent):void
{
    trace("Loaded " + event.bytesLoaded + " out of " +event.bytesTotal);
}
```

## ITEM\_COMPLETE

*Description:* Dispatched when the content of an item is completely loaded.

*Special Properties:* item – The item that has completely loaded its content.

index – The index of the item.

data – The data of the item.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;

myGrid3D.addEventListener(Grid3DEvent.ITEM_COMPLETE, eventHandler);

function eventHandler(event:Grid3DEvent):void
{
    trace("Item at index: " + event.index + " has finished loading");
}
```

## ITEM\_ERROR

*Description:* Dispatched when an error has occurred during the loading process.

*Special Properties:* item – The item on which the error occurred.

index – The index of the item.

data – The data of the item.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;

myGrid3D.addEventListener(Grid3DEvent.ITEM_ERROR, eventHandler);

function eventHandler(event:Grid3DEvent):void
{
    trace("Item at index: " + event.index + " could not be loaded");
}
```

## ZOOM\_ITEM\_CLICK

*Description:* Dispatched when a zoomed item was clicked.

*Special Properties:* item – The item that was clicked.

index – The index of the item.

data – The data of the item.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;

myGrid3D.addEventListener(Grid3DEvent.ZOOM_ITEM_CLICK, eventHandler);
```

```
function eventHandler(event:Grid3DEvent):void
{
    trace(event.index);
}
```

## **ZOOM\_ITEM\_DOUBLE\_CLICK**

*Description:* Dispatched when a zoomed item was double-clicked.

*Special Properties:* item – The item that was double-clicked.

index – The index of the item.

data – The data of the item.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.ZOOM_ITEM_DOUBLE_CLICK,
eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace(event.data);
}
```

## **ZOOM\_ITEM\_MOUSE\_UP**

*Description:* Dispatched when the mouse is released over a zoomed item.

*Special Properties:* item – The item under the mouse pointer.

index – The index of the item.

data – The data of the item.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.ZOOM_ITEM_MOUSE_UP, eventHandler);
function eventHandler(event:Grid3DEvent):void
```

```
{  
    trace(event.index);  
}
```

## ZOOM\_ITEM\_MOUSE\_DOWN

*Description:* Dispatched when the mouse is pressed over a zoomed item.

*Special Properties:* item – The item under the mouse pointer.

index – The index of the item.

data – The data of the item.

### *Example:*

```
import com.flashotaku.events.Grid3DEvent;  
myGrid3D.addEventListener(Grid3DEvent.ZOOM_ITEM_MOUSE_DOWN, eventHandler);  
function eventHandler(event:Grid3DEvent):void  
{  
    trace(event.index);  
}
```

## ZOOM\_ITEM\_MOUSE\_OVER

*Description:* Dispatched when the mouse pointer is moved over a zoomed item.

*Special Properties:* item – The item under the mouse pointer.

index – The index of the item.

data – The data of the item.

### *Example:*

```
import com.flashotaku.events.Grid3DEvent;  
myGrid3D.addEventListener(Grid3DEvent.ZOOM_ITEM_MOUSE_OVER, eventHandler);  
function eventHandler(event:Grid3DEvent):void  
{  
    trace(event.index);  
}
```

```
}
```

## ZOOM\_ITEM\_MOUSE\_OUT

*Description:* Dispatched when the mouse pointer is moved away from a zoomed item.

*Special Properties:* item – The item under the mouse pointer.

index – The index of the item.

data – The data of the item.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.ZOOM_ITEM_MOUSE_OUT, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace(event.index);
}
```

## ZOOM\_ITEM\_START

*Description:* Dispatched when the content of a zoomed item begins to load.

*Special Properties:* item – The item that begins to load its content.

index – The index of the item.

data – The data of the item.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.ZOOM_ITEM_START, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace("Item at index: " + event.index + " has started loading");
}
```

## ZOOM\_ITEM\_PROGRESS

*Description:* Dispatched during the loading of a zoomed item's content.

*Special Properties:* item – The item that is loading its content.

index – The index of the item.

data – The data of the item.

bytesLoaded – The number of bytes loaded thus far.

bytesLoaded – The number of bytes to be loaded.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.ZOOM_ITEM_PROGRESS, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace("Loaded " + event.bytesLoaded + " out of " +event.bytesTotal);
}
```

## ZOOM\_ITEM\_COMPLETE

*Description:* Dispatched when the content of a zoomed item is completely loaded.

*Special Properties:* item – The item that has completely loaded its content.

index – The index of the item.

data – The data of the item.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.ZOOM_ITEM_COMPLETE, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
```

```
        trace("Item at index: " + event.index + " has finished loading");
    }
```

## ZOOM\_ITEM\_ERROR

*Description:* Dispatched when an error has occurred during the loading process.

*Special Properties:* item – The item on which the error occurred.

index – The index of the item.

data – The data of the item.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.ZOOM_ITEM_ERROR, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace("Item at index: " + event.index + " could not be loaded");
}
```

## ITEM\_SELECTED

*Description:* Dispatched when an item was selected.

*Special Properties:* item – The item which was selected.

index – The index of the item.

data – The data of the item.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.ITEM_SELECTED, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace("Index of the selected item: " + event.index);
}
```

## ITEM\_DESELECTED

*Description:* Dispatched when an item was deselected.

*Special Properties:* item – The item which was deselected.

index – The index of the item.

data – The data of the item.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.ITEM_DESELECTED, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace("Index of the deselected item: " + event.index);
}
```

## UPDATE

*Description:* Dispatched after the component was updated because a property changed or an item was added or removed.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.UPDATE, eventHandler);

function eventHandler(event:Grid3DEvent):void
{
    trace("Something changed!");
}
```

## SCROLL\_START

*Description:* Dispatched when the scrolling starts.

**Example:**

```
import com.flashotaku.events.Grid3DEvent;

myGrid3D.addEventListener(Grid3DEvent.SCROLL_START, eventHandler);

function eventHandler(event:Grid3DEvent):void
{
    trace("scroll started");
}
```

### **SCROLL\_PROGRESS**

**Description:** Dispatched during the scrolling process.

**Example:**

```
import com.flashotaku.events.Grid3DEvent;

myGrid3D.addEventListener(Grid3DEvent.SCROLL_PROGRESS, eventHandler);

function eventHandler(event:Grid3DEvent):void
{
    trace("is scrolling");
}
```

### **SCROLL\_COMPLETE**

**Description:** Dispatched when the scrolling is complete.

**Example:**

```
import com.flashotaku.events.Grid3DEvent;

myGrid3D.addEventListener(Grid3DEvent.SCROLL_COMPLETE, eventHandler);

function eventHandler(event:Grid3DEvent):void
{
    trace("scroll complete");
}
```

```
}
```

## ZOOM\_START

*Description:* Dispatched when the zooming starts.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.ZOOM_START, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace("zoom started");
}
```

## ZOOM\_PROGRESS

*Description:* Dispatched during the zooming process.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.ZOOM_PROGRESS, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace("is zooming");
}
```

## ZOOM\_COMPLETE

*Description:* Dispatched when the zooming is complete.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.ZOOM_COMPLETE, eventHandler);

function eventHandler(event:Grid3DEvent):void
```

```
{  
    trace("zoom complete");  
}
```

## ZOOM\_OUT\_START

*Description:* Dispatched when the zoom out starts.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;  
myGrid3D.addEventListener(Grid3DEvent.ZOOM_OUT_START, eventHandler);  
function eventHandler(event:Grid3DEvent):void  
{  
    trace("zoom out started");  
}
```

## ZOOM\_OUT\_PROGRESS

*Description:* Dispatched during the zoom out process.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;  
myGrid3D.addEventListener(Grid3DEvent.ZOOM_OUT_PROGRESS, eventHandler);  
function eventHandler(event:Grid3DEvent):void  
{  
    trace("is zooming out");  
}
```

## ZOOM\_OUT\_COMPLETE

*Description:* Dispatched when the zoom out is complete.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;  
myGrid3D.addEventListener(Grid3DEvent.ZOOM_OUT_COMPLETE, eventHandler);
```

```
function eventHandler(event:Grid3DEvent):void
{
    trace("zoom out complete");
}
```

## **DISPLAY\_ZOOM\_ITEM**

*Description:* Dispatched when the zoomed item is displayed.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.DISPLAY_ZOOM_ITEM, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace(event.index);
}
```

## **OUTSIDE\_CLICK**

*Description:* Dispatched when the user clicks outside the component's items.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;
myGrid3D.addEventListener(Grid3DEvent.OUTSIDE_CLICK, eventHandler);
function eventHandler(event:Grid3DEvent):void
{
    trace("outside click");
}
```

## **OUTSIDE\_DOUBLE\_CLICK**

*Description:* Dispatched when the user double-clicks outside the component's items.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;
```

```
myGrid3D.addEventListener(Grid3DEvent.OUTSIDE_DOUBLE_CLICK, eventHandler);  
  
function eventHandler(event:Grid3DEvent):void  
{  
    trace("outside double click");  
}
```

## **OUTSIDE\_MOUSE\_UP**

*Description:* Dispatched when the mouse is released outside the component's items.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;  
  
myGrid3D.addEventListener(Grid3DEvent.OUTSIDE_MOUSE_UP, eventHandler);  
  
function eventHandler(event:Grid3DEvent):void  
{  
    trace("outside mouse up");  
}
```

## **OUTSIDE\_MOUSE\_DOWN**

*Description:* Dispatched when the mouse is pressed outside the component's items.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;  
  
myGrid3D.addEventListener(Grid3DEvent.OUTSIDE_MOUSE_DOWN, eventHandler);  
  
function eventHandler(event:Grid3DEvent):void  
{  
    trace("outside mouse down");  
}
```

## REACHED\_TOP

*Description:* Dispatched when the component has reached the top edge after `scrollUp()` method was called.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;

myGrid.addEventListener(Grid3DEvent.REACHED_TOP, eventHandler);

function eventHandler(event:Grid3DEvent):void
{
    trace("reached top");
}
```

## REACHED\_BOTTOM

*Description:* Dispatched when the component has reached the top edge after `scrollDown()` method was called.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;

myGrid.addEventListener(Grid3DEvent.REACHED_BOTTOM, eventHandler);

function eventHandler(event:Grid3DEvent):void
{
    trace("reached bottom");
}
```

## REACHED\_LEFT

*Description:* Dispatched when the component has reached the left edge after `scrollLeft()` method was called.

*Example:*

```
import com.flashotaku.events.Grid3DEvent;

myGrid.addEventListener(Grid3DEvent.REACHED_LEFT, eventHandler);

function eventHandler(event:Grid3DEvent):void
{
```

```
        trace("reached left");
    }
```

## **REACHED\_RIGHT**

*Description:* Dispatched when the component has reached the right edge after `scrollRight()` method was called.

### *Example:*

```
import com.flashotaku.events.Grid3DEvent;

myGrid.addEventListener(Grid3DEvent.REACHED_RIGHT, eventHandler);

function eventHandler(event:Grid3DEvent):void
{
    trace("reached right");
}
```