

3D Wall Pro

User Guide revision 2.3
www.flashloaded.com

Table of Contents

Installation	5
Getting started	6
XML	7
Setting component parameters in the XML file	10
Defining elements through ActionScript	11
Method 1: Defining XML elements	11
This is an example as to how your code would look to define the items using XML elements:	11
Method 2: Defining element objects	12
This is an example as to how your code would look to define the items using objects:	12
Method 3: Defining an array of element objects	13
This is an example as to how your code would look to define an array of item objects:	13
Component Inspector Parameters	14
General settings	14
Camera settings	15
Depth of field settings	16
Scrollbar settings	16
Stage scrolling	17
Opening animation	17
Closing animation	18
Thumbnail appearance	19
Large element appearance	20
Glow settings	21

Borders	22
Preloader settings	22
Reflection settings	23
Video settings	23
SWF settings	24
Tooltip settings	24
Enabling mouse wheel scrolling for Mac browsers	25
How to improve performance	26
Resizing large images	26
Skinning	27
Skinning the preloader	28
Skinning the tooltip	28
Skinning the close button	28
Skinning the scrollbar	28
Skinning the video controls	28
Skinning the watermark	28
Skinning the video icon	28
YouTube videos	29
Adding youtube.js to a Mac mousewheel compatible HTML file	29
Adding youtube.js to a standard Flash HTML file	30
ActionScript events	31
Adding titles to images	34
Opening a URL on click	35
Toggling scrollbar visibility when zooming	36
Disabling mouse wheel zooming when viewing large images	37

ActionScript properties	38
ActionScript methods	42
Help	47

Installation

You will need Adobe Extension Manager in order to install this component. Extension Manager should have been installed by default when you installed Flash. You may download the latest version of Extension Manager for free from the [Adobe website](#).

1. Ensure that Flash is closed before installing the 3D Wall Pro component.
2. Unzip/extract the 3DWallPro.zip file that you downloaded. You will find a file called 3DWallPro.mxp. Double click on this file in order to install the component using Extension Manager.

3D Wall Pro should now be installed in your Flash Components Panel.

Getting started

1. Having installed 3D Wall Pro using the Adobe Extension Manager, start a new Flash ActionScript 3 file and save it.
2. Prepare your images, SWF's and FLV files:
3D Wall Pro can use the same or different images for the large and thumbnail images. In order to set this up, create two folders called, for example, *thumbs* and *images* in the same location as your Flash file. Place the thumbnail images in the thumbs folder and the large images in the images folder. It's much simpler and quicker to set up if the large and thumbnail images have the same filenames.

Thumbnails should also be created for video and SWF's.

3. Image, SWF and video data can be defined either through XML or ActionScript. At this stage, you should create the XML file which contains the element data. Please refer to the [XML](#) section of this userguide for instructions on creating the XML file or the section on [Defining elements using ActionScript](#).
4. Locate the 3D Wall Pro component in the components panel and drag it onto the stage.
5. Use the Free Transform tool or the properties panel to resize the component to the desired display area.
6. Click on the component and open the Component Inspector panel (shift +F7).
7. Enter the name of the XML file that you created in step 3 in the XMLSource parameter.
8. At this stage, you can already test 3D Wall Pro with the default parameters, to ensure that you have set it up correctly. Press Ctrl+Enter (win) or Cmnd+Enter (mac) to test your movie.
9. You can change the various parameter settings in the Component Inspector to obtain the desired look. Please see the [Component Inspector parameters](#) section for a description on each setting.

Note: In order for the animations to be smooth it's recommended to set your movie speed to 31 fps.

XML

Important note: The XML file format for 3D Wall Pro differs entirely from the XML file format for the standard 3D Wall.

All of the images for 3D Wall Pro can be specified using an XML file. You can also set all of the parameters in the XML file. By defining the images and parameters in an external XML file, you can publish the SWF file once and change the images or parameters whenever you wish.

Note: The width and height values should be declared for all images, unless the images are exactly the same size, in which case they will all use the `largeImageDefaultWidth` and `largeImageDefaultHeight` settings.

1. Open your favorite plain text editor (for example Notepad on Windows or TextEdit on Mac) and start a new file. *Note: If you are using TextEdit on Mac, choose Format > Make Plain Text*
2. Begin your file with the following line:

```
<?xml version="1.0" encoding="utf-8"?>
```

This is the standard xml declaration.

3. Add the following lines to your xml file (the bold lines are the new additions)

```
<?xml version="1.0" encoding="utf-8"?>  
<gallery xmlns:media="http://search.yahoo.com/mrss/">  
</gallery>
```

4. Add the following lines between the gallery tags (the bold lines are the new additions)

```
<?xml version="1.0" encoding="utf-8"?>  
<gallery xmlns:media="http://search.yahoo.com/mrss/">  
<settings>  
  <mediaFolder type="large" media="video">video/</mediaFolder>  
  <mediaFolder type="thumbnail" media="video">video/thumbs/</mediaFolder>  
  <mediaFolder type="large" media="swf">swf/</mediaFolder>  
  <mediaFolder type="thumbnail" media="swf">swf/thumbs/</mediaFolder>  
  <mediaFolder type="large" media="image">wallimages/</mediaFolder>  
  <mediaFolder type="thumbnail" media="image">wallimages/thumbs/</  
mediaFolder>  
</settings>  
</gallery>
```

The `mediaFolder` tags define the folder in which each type of media is located:
video flv files, video thumbnails, swf files, swf thumbnails, images, image thumbnails

5. Add the item tags to your XML file (the bold lines are the new additions)

```
<?xml version="1.0" encoding="utf-8"?>
<gallery xmlns:media="http://search.yahoo.com/mrss/">
<settings>
  <mediaFolder type="large" media="video">video/</mediaFolder>
  <mediaFolder type="thumbnail" media="video">video/thumbs/</mediaFolder>
  <mediaFolder type="large" media="swf">swf/</mediaFolder>
  <mediaFolder type="thumbnail" media="swf">swf/thumbs/</mediaFolder>
  <mediaFolder type="large" media="image">wallimages/</mediaFolder>
  <mediaFolder type="thumbnail" media="image">wallimages/thumbs/</
mediaFolder>
</settings>
<item>
  <media:text>Tooltip for image 8</media:text>
  <media:content url="8.jpg" type="image/jpeg" width="500" height="334" />
</item>
</gallery>
```

Variations of the item tag:

The media type must be set to reflect the type of media displayed. The different possibilities are as follows:

Displaying an image:

```
<item>
  <media:text>Tooltip for image 8</media:text>
  <media:content url="8.jpg" type="image/jpeg" width="500" height="334" />
  <media:thumbnail url="8_thumb.jpg" />
</item>
```

Displaying an flv video:

```
<item>
  <media:text>A local flv</media:text>
  <media:content url="3.flv" type="video/x-flv" width="320" height="240" />
  <media:thumbnail url="3.png" />
</item>
```

Displaying an SWF:

```
<item>
  <media:text>SWF test</media:text>
  <media:content url="flashfile.swf" type="application/x-shockwave-flash"
width="400" height="300" />
  <media:thumbnail url="4.png" />
</item>
```

Displaying an streaming video using the RTMP protocol:

To use rtmp video, a server address must be given as an attribute of the content tag. Also, the .flv extension must be omitted from the file name.

```
<item>
  <media:text>A streamed flv</media:text>
  <media:content url="mymovie" rtmpServer="rtmp://streaming/server/
address/" type="video/x-flv" width="320" height="240" />
  <media:thumbnail url="3.png" />
</item>
```

Displaying an YouTube video:

```
<item>
  <media:text>YouTube video</media:text>
  <media:content url="http://www.youtube.com/watch?v=ABC" type="video/x-
flv" width="320" height="240" />
  <media:thumbnail url="video1.png" />
</item>
```

Here's an explanation of the tags used in the above examples:

text (optional): defines a tooltip for the element

type: the type of media:
image = image/jpeg
video = video/x-flv
swf = application/x-shockwave-flash

thumbnail url: the name of the thumbnail image.

Any additional optional parameters can be included within the item tags which can read through ActionScript events. For example:

```
<item name="Vacation" description="Vacation photo">
  <media:text>Tooltip for image 8</media:text>
  <media:content url="8.jpg" type="image/jpeg" width="500" height="334" />
  <media:thumbnail url="8_thumb.jpg" />
</item>
```

6. Save the XML file to the same folder as your Flash file. In this example, we have given the XML file the name: *images.xml*

7. Return to your Flash file. Enter the name and path to the XML file that you just created in the XMLSource parameter of the 3D Wall Pro that's on the stage.

Note: If your .swf file will be in a different folder to the HTML file in which it is embedded, you should enter the path to the XML file, relative to the location of the .html file.

8. Press Ctrl+Enter (Win) or Cmnd+Enter (Mac) to test your movie.

Setting component parameters in the XML file

All of the parameters that appear in the Component Inspector can be set in the XML file. Any parameter that is set in the XML will override the value for that parameter that has been set in the Component Inspector.

In order to set a parameter in the XML file, simply define the parameter and the value in the *gallery* tag like this:

```
<gallery xmlns:media="http://search.yahoo.com/mrss/" numRows="2"  
wallCurvature="-360" >
```

Defining elements through ActionScript

1. Give the 3D Wall Pro that's your the stage and instance name, for example: *wall*
2. Ensure that the *XMLSource* parameter in the Component Inspector is blank.
3. There are three ways to define the images:

Method 1: Defining XML elements

This is an example as to how your code would look to define the items using XML elements:

```
wall.setMediaPaths("wallimages/", "wallimages/thumbs/");

wall.addElements('<item><media:text xmlns:media="http://
search.yahoo.com/mrss/">Tooltip 1</media:text><media:content
xmlns:media="http://search.yahoo.com/mrss/" url="1.jpg" type="image/
jpeg" width="400" height="300" /><media:thumbnail xmlns:media="http://
search.yahoo.com/mrss/" url="thumb1.jpg" /></item>', 'XML');

wall.addElements('<item><media:text xmlns:media="http://
search.yahoo.com/mrss/">Tooltip 2</media:text><media:content
xmlns:media="http://search.yahoo.com/mrss/" url="2.jpg" type="image/
jpeg" width="400" height="300" /><media:thumbnail xmlns:media="http://
search.yahoo.com/mrss/" url="thumb2.jpg" /></item>', 'XML');

wall.addElements('<item><media:text xmlns:media="http://
search.yahoo.com/mrss/">Tooltip 3</media:text><media:content
xmlns:media="http://search.yahoo.com/mrss/" url="3.jpg" type="image/
jpeg" width="400" height="300" /><media:thumbnail xmlns:media="http://
search.yahoo.com/mrss/" url="thumb3.jpg" /></item>', 'XML');

wall.init();
```

In the above code:

setMediaPaths defines the folders containing the large and thumbnail images/elements (ending with a *"/*) in this order:

setMediaPaths(imageFolder, imageThumbnailsFolder, videoFolder, videoThumbnailsFolder, swfFolder, swfThumbnailsFolder)

addElements defines the XML element, following the same format as the item tags in the XML file.
init - after defining the items, the *init* method must be called to initialize the wall.

Method 2: Defining element objects

This is an example as to how your code would look to define the items using objects:

```
wall.setMediaPaths("wallimages/", "wallimages/thumbs/");
var obj:Object = new Object();
obj.type="image/jpeg";
obj.thumb="1.jpg";
obj.src="1.jpg";
obj.width=520;
obj.height=345;
wall.addElements(obj, "object");

var obj2:Object = new Object();
obj2.type="image/jpeg";
obj2.thumb="2.jpg";
obj2.src="2.jpg";
obj2.width=500;
obj2.height=332;
wall.addElements(obj2, "object");
wall.init();
```

In the above code:

setMediaPaths defines the folders containing the large and thumbnail images/elements (ending with a "/") in this format: *setMediaPaths("LARGE IMAGE FOLDER/", "THUMB IMAGE FOLDER");*

type defines the type of media: *image/jpeg, video/x-flv, application/x-shockwave-flash*

thumb defines name of the thumbnail image.

src defines the name of the large element (image, swf or flv)

addElements adds the item object to the 3D Wall Pro instance.

init - after defining the items, the init method must be called to initialize the wall.

Method 3: Defining an array of element objects

This is an example as to how your code would look to define an array of item objects:

```
wall.setMediaPaths("wallimages/", "wallimages/thumbs/");
var obj:Object = new Object();
obj.type="image/jpeg";
obj.thumb="1.jpg";
obj.src="1.jpg";
obj.width=520;
obj.height=345;

var obj2:Object = new Object();
obj2.type="image/jpeg";
obj2.thumb="2.jpg";
obj2.src="2.jpg";
obj2.width=500;
obj2.height=332;

var obj3:Object = new Object();
obj3.type="image/jpeg";
obj3.thumb="3.jpg";
obj3.src="3.jpg";
obj3.width=500;
obj3.height=332;

var objArray:Array = new Array();
objArray.push(obj);
objArray.push(obj2);
objArray.push(obj3);
wall.addElements(objArray, "array");

wall.init();
```

In the above code:

setMediaPaths defines the folders containing the large and thumbnail images/elements (ending with a "/") in this format: *setMediaPaths("LARGE IMAGE FOLDER/", "THUMB IMAGE FOLDER");*

type defines the type of media: *image/jpeg, video/x-flv, application/x-shockwave-flash*

thumb defines name of the thumbnail image.

src defines the name of the large element (image, swf or flv)

addElements adds the array of item objects to the 3D Wall Pro instance.

init - after defining the items, the init method must be called to initialize the wall.

Component Inspector Parameters

General settings

Parameter	Description	Example
XMLSource	The path and filename of the XML file containing the image/item information. If you are loading a Flickr feed, you can specify the URL of the Flickr RSS feed here. For example: <code>http://api.flickr.com/services/feeds/photos_public.gne?id=123@N05&lang=en-us&format=rss_200</code>	images.xml
XMLNoCache	Defines whether to force 3D Wall Pro to use a non-cached version of the XML file. This is useful for situations where the XML file may be updated. <i>Note: This parameter must always be set to false when testing locally.</i>	true
numRows	The number of rows to display. The number of columns is calculated automatically.	5
wallCurvature	The angle at which the wall curves. A positive number produces a concave effect (as if the viewer is inside the wall) and a negative number produces a convex effect (outside the wall). Set this to -360 to carousel type of effect. <i>Note: In some cases, it might be best to set the cameraRotationFactor to 1 (no rotation).</i>	-180
elementLimit	Limits the maximum number of elements to load. This is useful when loading images from Flickr where you do not know the amount of images in the feed.	60
smoothElements	Sets whether the smoothing of images is controlled automatically by 3D Wall Pro or if smoothing is always on. When set in <i>auto</i> mode, smoothing is automatically turned off while animations are occurring. Setting this parameter to <i>on</i> may have an impact on performance on curved walls. Options available are: <i>auto</i> , <i>on</i> or <i>off</i>	auto
allowKeyboardControl	Sets whether to enable the arrow and space keys for navigation and zooming.	true

Parameter	Description	Example
cameraActionOnScrollWheel	Sets whether mouse wheel scrolling should be enabled for zooming, cycling through the items or disabled.	zoom
scrollWheelSensitivity	The higher the number, the more sensitive the scrollwheel zooming is.	3
useHandCursor	Sets whether to show the hand cursor when the mouse is over the thumbnails and large images/items or not.	false
showWarningMessages	Sets whether to display performance related settings warning messages when testing in the Flash IDE or not.	true
onBadImageContinueLoading	Sets whether to ignore images that could not be found while loading and to continue loading.	false

Camera settings

Parameter	Description	Example
cameraRotationFactor	Sets the amount to rotate/swivel the wall when the wall is scrolling. Set this to 1 for no rotation. This should be used in conjunction with the cameraReactionTime parameter to achieve the desired effect.	3
cameraReactionTime	The reaction time of the camera.	7
cameraStartPosition	The position of the wall when the wall initially loads in percentage: <i>0 = Starts from the left</i> <i>0.5 = Starts from the center</i> <i>1 = Starts from the right</i>	50
cameraTiltStart	The starting tilt setting for the wall.	200
cameraDistance	The distance of the camera from the wall. Larger distances result in less rotation.	
cameraZoom	This works the same same way that zoom works on a camera. This should be set in combination with the cameraDistance setting to achieve the desired look.	6

Depth of field settings

Parameter	Description	Example
depthOfFieldEnabled	Sets whether to display the background thumbnails with as clear or blurry (indicating depth of field). This is used in cases where you can view the backs of the thumbnails (e.g. wallCurvature = -360).	true
depthOfFieldFineness	The higher the number, the better the quality however this will have an impact on performance.	30
depthOfFieldMaxBlur	The maximum amount that the image furthest back should blur.	10
depthOfFieldBrightnessVariation	The percentage amount of variation in the brightness for images that are further back.	70

Scrollbar settings

Parameter	Description	Example
scrollbar	Determines whether the built-in scrollbar is visible or not.	true
scrollbarWidth	The width of the scroller's dragbar. If you change the width through skinning, the new width should be changed here as well.	50
scrolltrackWidth	The width of the invisible scrolltrack. The scrollbar will only be able to scroll along the width of the track. In most cases you would set this to be the same as the component width.	800
scrolltrackX	The X position of the scrolltrack (relative to the component).	0
scrolltrackY	The Y position of the scrolltrack (relative to the component).	3

Stage scrolling

Parameter	Description	Example
stageScrollVerticalEnabled	Allows tilting by clicking and dragging the mouse. Enabling this has a major impact on performance.	false
stageScrollVerticalUpperLimit	The y value of the upper limit of the stage scrolling (0 marks the top of the wall).	500
stageScrollVerticalLowerLimit	The y value of the lower limit of the stage scrolling. 0 marks the lowest image in the wall. <i>Note: This must be a negative value. For walls of -360 curvature, a setting of -100 is recommended.</i>	-200
stageScrollSensitivity	The higher the number, the more sensitive the stage scrolling is.	3
autoScroll	Determines whether to enabled free mouse movement scrolling or not.	true
autoScrollXSensitivity	Sets the reaction sensitivity of the horizontal mouse movement. Recommended range: 1 to 2	1.5
autoScrollYSensitivity	Sets the reaction sensitivity of the vertical mouse movement. Recommended range: 1 to 2 <i>Note: Setting this to a low value will affect the tilt range.</i>	1.5
autoScrollDeadZone	Defines the number of pixels in the center area in which auto scrolling is not active. This is used to ensure that the wall remains still while moving the mouse in this zone.	50

Opening animation

Parameter	Description	Example
thumbnailFlyInEasing	The easing style for the opening thumbnail animation. <i>* See easing styles</i>	easeOutQuad
thumbnailFlyInStartScale	The percentage at which the thumbnails start their opening animation.	0
thumbnailFlyInDistance	The distance from the camera at which the thumbnails start the opening animation.	800

Parameter	Description	Example
thumbNailerFlyInTimingDistribution	The number of seconds it should take for all of the thumbnails to have their animation sequence started. 0 means they all fly in together, higher numbers result in them coming in one by one. This only applies if preloadAllElementsBeforeShowing is set to true.	2
thumbnailFlyInTime	The speed of the thumbnail opening animation.	2
thumbnailFlyInFrom	Set the thumbnail opening animation to be from behind the camera or behind the wall.	behind camera
thumbnailFlyInRotationX	The amount to rotate the thumbnails around the X axis when flying in.	0
thumbnailFlyInRotationY	The amount to rotate the thumbnails around the Y axis when flying in.	100
thumbnailFlyInRotationZ	The amount to rotate the thumbnails around the Z axis when flying in.	0

Closing animation

Parameter	Description	Example
thumbnailFlyOutEasing	The easing style for the closing thumbnail animation, used when closing the wall through ActionScript. <i>* See easing styles</i>	easeOutQuad
thumbnailFlyOutStartScale	The percentage at which the thumbnails start their closing animation.	0
thumbnailFlyOutDistance	The distance from the camera at which the thumbnails start the closing animation.	800

Parameter	Description	Example
thumbNailerFlyOutTimingDistribution	The number of seconds it should take for all of the thumbnails to have their animation sequence started. 0 means they all fly out together, higher numbers result in them flying out one by one.	2
thumbnailFlyOutTime	The speed of the thumbnail closing animation.	2
thumbnailFlyOutFrom	Set the thumbnail closing animation to be from behind the camera or behind the wall.	behind camera
thumbnailFlyOutRotationX	The amount to rotate the thumbnails around the X axis when flying out.	0
thumbnailFlyOutRotationY	The amount to rotate the thumbnails around the Y axis when flying out.	100
thumbnailFlyOutRotationZ	The amount to rotate the thumbnails around the Z axis when flying out.	0

Thumbnail appearance

Parameter	Description	Example
thumbnailSpacingHorizontal	The number of pixels horizontally between each thumbnail. <i>Note: The glow and depth of field settings can affect the spacing</i>	15
thumbnailSpacingVertical	The number of pixels vertically between each thumbnail. <i>Note: The glow and depth of field settings can affect the spacing</i>	15
thumbnailMaxWidth	The maximum width that the thumbnails will display at.	120
thumbnailMaxHeight	The maximum height that the thumbnails will display at.	80

Parameter	Description	Example
thumbnailsDoubleSided	Sets whether the thumbnails should be double sided or not. This would be used in situations where the wall curvature is set at an angle where you see the back of the thumbnails. <i>Note: Setting this to true impacts the performance.</i>	true
thumbnailSegmentsHorizontal	The number of horizontal segments that the thumbnails are divided into. Increase this number only if the images are too distorted. <i>Note: It is recommended to leave this at a setting of 1 wherever possible as higher numbers can impact performance.</i>	1
thumbnailSegmentsVertical	The number of horizontal segments that the thumbnails are divided into. Increase this number only if the images are too distorted. <i>Note: It is recommended to leave this at a setting of 1 wherever possible as higher numbers can impact performance.</i>	1
stretchFlickrThumbnails	This parameter is used when loading images from Flickr. It defines whether to stretch the Flickr generated thumbnail images to the defined thumbnail size or to leave them as their standard Flickr square shape.	true

Large element appearance

Parameter	Description	Example
scaleElementDown	Sets whether images that are too large to fit into the viewport of the component are scaled down proportionally to fit (taking glow into account).	true
largeElementDefaultWidth	The default width of the large images. This value can be overridden for individual images in the XML.	600
largeElementDefaultHeight	The default height of the large images. This value can be overridden for individual images in the XML.	450

Parameter	Description	Example
largeElementZoomOutTime	The speed at which the large images return to the thumbnails.	1
largeElementZoomOutEasing	The easing style for the motion for which the large images return to the thumbnails. <i>* See easing styles</i>	easeOutQuad
largeElementZoomInTime	The speed at which the large images enlarge.	1
largeElementZoomInEasing	The easing style for the motion for which the large images enlarge. <i>* See easing styles</i>	easeOutQuad
largeImageTogglesOnDoubleClick	Sets whether the large image should open and close on single or double click.	false
useWatermark	Sets whether to display a watermark over the large images or not. The watermark can be customized. See the skinning section for more details.	true
watermarkAlpha	The alpha/opacity of the watermark. Set this at a decimal between 0 and 1, with 1 being maximum alpha.	1

Glow settings

Parameter	Description	Example
glowEnabled	Sets whether to display a glow around the large image and the last selected thumbnail or not. Note: Glow will be automatically disabled if the video or SWF thumbnails have been set to animate on mouse over.	true
glowColour	The color of the glow.	#FF0000
glowAlpha	The alpha of the glow (between 0-1).	1
glowBlurX	The amount of blur of the glow in the X direction. <i>Note: This setting can also affect the space between the thumbnails</i>	8

Parameter	Description	Example
glowBlurY	The amount of blur of the glow in the Y direction. <i>Note: This setting can also affect the space between the thumbnails</i>	8
glowStrength	The strength of the glow.	2
glowQuality	The quality of the glow.	1
glowInner	Sets whether the glow is positioned around the inside of outside of the image.	false
glowKnockout	Sets whether the glow should have a knockout look or not.	false

Borders

Parameter	Description	Example
thumbBorder	The width of the border color around the images. Setting this to any value above 0 will disable glow.	2
thumbBorderColourUp	The color of the image borders.	#FF0000
thumbBorderColourOver	The mouse over color of the image borders.	#00FF00

Preloader settings

Parameter	Description	Example
preloadAllElementsBeforeShowing	Sets whether to preload all of the thumbnails before displaying anything. It is recommended to always set this to <i>true</i> to ensure a smooth opening animation.	true
preloadText	The text that should appear to indicate the number of thumbnails loading and total thumbnails to load. Use %NUM% to represent the number of thumbnails loaded and %TOTAL% to represent the total loaded.	Loaded %NUM% of %TOTAL% thumbnails
preloaderAlpha	The percentage of opacity of the preloader.	75

Reflection settings

Parameter	Description	Example
showReflections	Sets whether to show the reflections or not.	true
reflectionScale	The vertical scale of the reflections as a percentage of the thumbnail size (0-1).	1
reflectionGradientStartAlpha	The alpha setting for the side of the reflection closest to the thumbnail image (0-1)	1
reflectionGradientEndAlpha	The alpha setting for the side of the reflection furthest from the thumbnail image.	0
reflectionDistance	The distance (in pixels) of the reflections from the bottom row of thumbnails. <i>Note: The glow and depth of field settings can affect the reflection distance.</i>	0

Video settings

Parameter	Description	Example
videoBufferTime	The amount of time to buffer when playing videos. In most cases you should not need to change this setting.	0.1
videoControls	Sets whether to display controls for the large videos or not.	true
videoControlsAutoHide	Sets whether the video controls should hide automatically on mouse out.	true
videoControlsVerticalOffset	The vertical offset distance (in pixels) of the video controls from the large video.	10
videoAlwaysRestarts	Sets whether the videos always start from the beginning each time the thumbnails are clicked. <i>Note: YouTube videos will always restart from the beginning, regardless of this setting.</i>	true
videoLoops	Sets whether the videos loops or not.	true
showVideoIcon	Set to true to display an a video symbol on thumbnails for videos.	true

Parameter	Description	Example
youTubePlayerBridgePath	This is required for YouTube videos only: The path and filename to the youTubePlayerBridge.swf file. See the YouTube videos section for more information. <i>Note: This path should be relative to the html file in which the 3D Wall Pro SWF is embedded.</i>	youTubePlayerBridge.swf
youTubeVideoScrubbing	This is required for YouTube videos only: Sets whether to allow the progress bar to be dragged (scrubbing) while playing YouTube videos. It is recommended to set this to <i>false</i> for most scenarios.	false

SWF settings

Parameter	Description	Example
showCloseButton	Sets whether to display a close button in the top right corner of SWF thumbnails or not. SWF's will not close when clicking on them. The look of the close button can be customized. See the skinning section for more details.	true

Tooltip settings

Parameter	Description	Example
showToolTips	Sets whether to display tooltips or not.	true
tooltipAlignment	Sets the position of the tooltips in relation to the mouse. Options available are: <i>top right, top left, bottom right, bottom left</i>	top right
tooltipOffset	The number of pixels to offset the tooltip from the mouse cursor.	10

* The following easing styles are available:

linear, easeInQuad, easeOutQuad, easeInOutQuad, easeInExpo, easeOutExpo, easeInOutExpo, easeOutInExpo, easeInElastic, easeOutElastic, easeInOutElastic, easeOutInElastic, easeInBack, easeOutBack, easeInOutBack, easeOutInBack, easeOutBounce, easeInBounce, easeInOutBounce, easeOutInBounce, easeInCubic, easeOutCubic, easeInOutCubic, easeOutInCubic, easeInQuart, easeOutQuart, easeInOutQuart, easeOutInQuart, easeInQuint, easeOutQuint, easeInOutQuint, easeOutInQuint, easeInSine, easeOutSine, easeInOutSine, easeOutInSine, easeInCirc, easeOutCirc, easeInOutCirc, easeOutInCirc

Enabling mouse wheel scrolling for Mac browsers

The current version of the Flash Player does not natively support mouse wheel scrolling in Mac browsers. We have built a solution for this into 3D Wall Pro. In order to use this solution, you must construct your HTML file like this:

1. Copy the **js** folder, that was included with your download, to the same folder in which your HTML file will reside.
2. Write the following code in the <head></head> section of your HTML file:

```
<script type="text/javascript" src="js/swfobject.js"></script>
<script type="text/javascript" src="js/swfmacmousewheel2.js"></script>
<script type="text/javascript">
  var vars = {};
  var params = { scale:'noScale', salign:'lt', menu:'true' };
  var attributes = { id:'wallObject', name:'wallObject' };
  swfmacmousewheel.registerObject(attributes.id);
</script>
```

3. Write the following code in the body of your HTML file, where you would like the Flash SWF to be located:

```
<script>swfobject.embedSWF("wallexample.swf", "flashContent", "1000",
"600", "9.0.0", "js/expressInstall.swf", vars, params, attributes );</
script>
```

Note: Change the items marked in bold to match your SWF filename, height and width.

This will work when testing online only. You must ensure that you upload the **js** folder with your HTML file.

How to improve performance

There are several factors which can affect the performance of 3D Wall Pro, especially when viewed on older computers. The following settings will help improve the performance:

1. Set ***depthOfFieldEnabled*** to *false*
2. If ***depthOfFieldEnabled*** is set to *true*, try entering a lower number for ***depthOfFieldFineness***
3. Set ***glowEnabled*** to *false*
4. Set ***showReflections*** to *false*
5. Set ***stageScrollVerticalEnabled*** (tilting) to *false*
6. If ***stageScrollVerticalEnabled*** is set to *true*, use a ***stageScrollVerticalLowerLimit*** of *-100*
7. Set ***smoothImages*** to *auto* or *off*
8. Large component sizes can impact performance, especially if there are a large number of images. Try make the component size smaller.

Note: You may need to tweak the above settings based on the number of images displayed.

Resizing large images

3D Wall Pro does not resize the large images by specifying a smaller image size, although the size of the large images must be specified in the component or XML file. This size specification is necessary in order for 3D Wall Pro to zoom to the correct size and to resize the thumbnails proportionally. The large images can be automatically scaled down to fit the maximum size of the component by setting the *scaleImageDown* parameter to *true*.

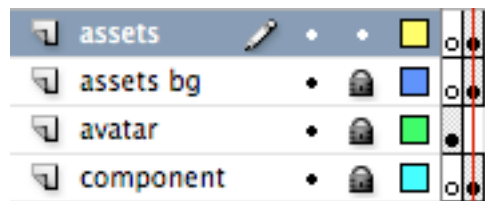
Skinning

The scrollbar, preloader animation, preloader text, close button, video controls, watermark and video icon can all be skinned to match your desired look and feel. Double click anywhere on the 3D Wall Pro component that's on the stage in order to skin these elements.

You should now see the skinnable movie clips:



Note: In order to skin any of these elements, you must unlock the assets layer:



Skinning the preloader

Double click on the *imagesPreloader* movie clip in order to skin the preloader text and animation. You should see a dynamic textfield which displays the loading text and the preloader animation movie clip (which is called *preloader*).

You can change the color of the built-in preloader animation by changing the tint of the *preloader* movie clip or you can double click on this movie clip in order to change the symbols and animation entirely.

Note: You can replace this movie clip with another animation if you wish. The preloader animation movie clip must have a center registration point.

Edit the textfield in order to change the font and style of the preloading text.

Skinning the tooltip

Double click on the *tooltip* movie clip in order to edit the look of the tooltip. Resize the width of the tooltip background and the width of the textfield to the maximum desired tooltip width. Longer text will wrap onto multiple lines.

Skinning the close button

Double click on the *close button* movie clip in order to edit the look or size of the close button which appears on SWF's.

Skinning the scrollbar

Double click on the *scrollbar* movie clip in order to edit the look of the scrollbar. If you change the size of the scrollbar, you must enter the new width in the *scrollbarWidth* property in the Component Inspector.

Skinning the video controls

The video controls can be skinned by double clicking on the *video controls* movie clip.

Skinning the watermark

The watermark controls can be skinned by double clicking on the *watermark* movie clip.

Skinning the video icon

The video icon can be skinned by double clicking on the *video icon* movie clip. Within this movie clip, you will see a backing layer defining the thumbnail area. The video icon can be positioned in the top left, top right or center of this thumbnail, by dragging it relative to the backing layer.

YouTube videos

3D Wall Pro can display videos that are fed directly from YouTube. In order to display YouTube videos, you will need the following two files which are included with the 3D Wall Pro download: `youtubePlayerBridge.swf` and `youtube.js`

The javascript file (`youtube.js`) is not necessary when testing locally in the Flash IDE or if the SWF is not running in a browser.

1. Copy the `youtubePlayerBridge.swf` file to the same folder as your SWF. If your published SWF and HTML file are not in the same folder, ensure that you set the path relative to the HTML file in the `youtubePlayerBridgePath` parameter in the Component Inspector.
2. You can now test your wall locally and the videos should play from YouTube however, they will not work when viewed through a web browser without including the `youtube.js` JavaScript file.

Adding `youtube.js` to a Mac mousewheel compatible HTML file

These instructions should be followed if you are using the recommended HTML format and JavaScript files to enable mousewheel zooming for Mac browser.

1. Open the `youtube.js` file in any text editor or an application, such as Dreamweaver.
2. The first line contains: `var swfName = "swf name or id";`
3. Replace "`swf name or id`" with the ***id*** that you gave to your 3D Wall Pro swf in the attributes parameter of the HTML file. In our example, we called it *wallObject*, therefore we would change this line to:

```
var swfName = "wallObject";
```

4. Save the `youtube.js` file and copy it to the same location as your swf file or to a subdirectory.
5. Insert the following line in the `<head></head>` section of your HTML file to reference `youtube.js`:

```
<script type="text/javascript" src="youtube.js"></script>
```

Note: If you placed `youtube.js` in a subdirectory then you should change the path to the file accordingly.

The YouTube videos should now play correctly when viewed through a browser.

Adding youTube.js to a standard Flash HTML file

These instructions should be followed if you are using a standard HTML file or the HTML file that is product when publishing the Flash files.

1. Open the *youTube.js* file in any text editor or an application, such as Dreamweaver.

2. The first line contains: `var swfName = "swf name or id";`

3. Replace "swf name or id" with the **name** of your SWF file, **without .swf**. For example:

```
var swfName = "mywall";
```

4. Save the *youTube.js* file and copy it to the same location as your swf file or to a subdirectory.

5. Insert the following line in the `<head></head>` section of your HTML file to reference *youTube.js*:

```
<script type="text/javascript" src="youTube.js"></script>
```

Note: If you placed youTube.js in a subdirectory then you should change the path to the file accordingly.

The YouTube videos should now play correctly when viewed through a browser.

ActionScript events

Events are called whenever 3D Wall Pro performs the specified action. The component includes an event class called Wall3DEvent in the com.flashloaded.Wall3D package.

The event has an item property which holds the following element properties:

type: the type of event that was initiated (e.g: *CLICK*, *MOUSE_OVER*, *MOUSE_OUT*)

src: the source file of the element on which the event was initiated

param: the optional parameter for the element as defined in the *param* value in the XML

state: whether the element is in the default state or enlarged state (*Wall3DElement.STATE_DEFAULT* or *Wall3DElement.STATE_ENLARGED*)

The following events are included:

Wall3DEvent.CAMERA_MOVEMENT_STARTED

Broadcasted whenever the camera starts to move

Wall3DEvent.CAMERA_MOVEMENT_COMPLETE

Broadcasted whenever the camera movements has completed.

Wall3DEvent.CLICK

Broadcasted when clicking on a large or thumbnail element.

Wall3DEvent.CLOSE_WALL_COMPLETE

Broadcasted when the wall has finished closing (when animated closing has been selected) .

Wall3DEvent.CLOSE_WALL_STARTED

Broadcasted when the wall starts to close (when animated closing has been selected).

Wall3DEvent.DOUBLE_CLICK

Broadcasted when double clicking on a large or thumbnail element, provided that *largeElementTogglesOnDoubleClick* is enabled.

Wall3DEvent.ELEMENT_DESELECTED

Broadcasted when the selected element returns to being a thumbnail.

Wall3DEvent.ELEMENT_SELECTED

Broadcasted when an element is selected. This event is recommended over the CLICK event as this will be called when the element is selected using the arrow keys.

Wall3DEvent.INTRO_ANIM_COMPLETE

Broadcasted when the opening animation sequence had completed.

Wall3DEvent.INTRO_ANIM_STARTED

Broadcasted when the opening animation sequence starts.

Wall3DEvent.LARGE_ELEMENT_ADDED

Broadcasted when a large image is displayed.

Wall3DEvent.LARGE_ELEMENT_REMOVED

Broadcasted when a large image is removed.

Wall3DEvent.MOUSE_OUT

Broadcasted when the viewer moves their mouse off a large or thumbnail element.

Wall3DEvent.MOUSE_OVER

Broadcasted when the viewer moves their mouse over a large or thumbnail element.

Wall3DEvent.NETWORK_ERROR

Broadcasted when there is an error loading an element.

Wall3DEvent.THUMBNAIL_LOADED

Broadcasted as each thumbnail has loaded.

Wall3DEvent.VIDEO_COMPLETE

Broadcasted when a video completes playing.

Wall3DEvent.VIDEO_STARTED

Broadcasted when a video starts playing.

Wall3DEvent.WALL_LOADED

Broadcasted when all the thumbnails have loaded and the animation sequence starts.

Wall3DScrollEvent.SCROLLBAR_OUT

Broadcasted when the mouse leaves the scrollbar area

Wall3DScrollEvent.SCROLLBAR_OVER

Broadcasted when the mouse is placed over the scrollbar.

Wall3DScrollEvent.SCROLLING_PERCENT

Broadcasted when the scrolling percentage changes.

Wall3DScrollEvent.SCROLLING_STARTED

Broadcasted when scrolling is happening.

Wall3DScrollEvent.SCROLLING_STOPPED

Broadcasted when scrolling has stopped.

Wall3DScrollEvent.SCROLL_TOOL_SCROLLBAR

Broadcasted when scrolling is performed by clicking and dragging the stage.

Wall3DScrollEvent.SCROLL_TOOL_STAGE

Broadcasted when scrolling is performed using the scrollbar.

Example 1- This outputs a trace for the CLICK, MOUSE_OVER and MOUSE_OUT events:

```
import com.flashloaded.Wall3DPro.Wall3DEvent;

3DWallinstance.addEventListener(Wall3DEvent.CLICK, traceMe);
3DWallinstance.addEventListener(Wall3DEvent.MOUSE_OVER, traceMe);
3DWallinstance.addEventListener(Wall3DEvent.MOUSE_OUT, traceMe);

function traceMe(e:Wall3DEvent):void {
    trace("action: " + e.type + ", element state: " + e.element.state
+ ", src: " + e.element.src + ", optional parameter: " +
    e.element.param);
}
```

Example 2- This is how you could use the click event:

```
import com.flashloaded.Wall3DPro.Wall3DEvent;
3DWallinstance.addEventListener(Wall3DEvent.CLICK,clickHandler);

function clickHandler(evt:Wall3DEvent):void{
    trace(evt.element.src);
    gotoAndStop(evt.element.param);
}
```

Adding titles to images

You can add a title or description for each image which appears in a textfield. This is done by adding a parameter containing the title in the XML file and by reading the parameter in the **ELEMENT_SELECTED** event. This is how you would do this:

1. Add a title value to item tag in the XML. For example:

```
<item title="Tree title" >
  <media:content url="t.jpg" type="image/jpeg" width="500" height="334" />
  <media:thumbnail url="tree_thumb.jpg" />
</item>
```

2. Place a textfield on your stage where you would like the titles to appear. Set the textfield to be dynamic and give it an instance name, e.g: *title_txt*
3. Give the 3D Wall Pro that's on the stage an instance name, e.g: *wall*
4. Type the following ActionScript code on the timeline, in the first frame in which 3D Wall Pro appears:

```
import com.flashloaded.Wall3DPro.Wall3DEvent;
wall.addEventListener(Wall3DEvent.ELEMENT_SELECTED,selectedHandler);
wall.addEventListener(Wall3DEvent.ELEMENT_DESELECTED,deselectedHandler
);

function selectedHandler(evt:Wall3DEvent):void{
  title_txt.text = evt.element.getUserProperty("title");
}

function deselectedHandler(evt:Wall3DEvent):void{
  title_txt.text = "";
}
```

*Note: In this example, you would replace **wall** with the instance name of your 3D Wall Pro and replace **title_txt** with the instance name of your title textfield.*

You should now see that the title appears when viewing and moving between the large images.

Opening a URL on click

You can have a URL open when clicking on an image. This is done by specifying the URL in a self defined parameter element of the XML file (which you can call *url*) and by reading the URL in the **CLICK** event. This is how you would do this:

1. Add a parameter called *url* to each image in the XML. The parameter will contain the URL. For example:

```
<item url="http://www.flashloaded.com" >
  <media:content url="fl.jpg" type="image/jpeg" width="500" height="334" />
  <media:thumbnail url="fl_thumb.jpg" />
</item>
```

2. Give the 3D Wall Pro that's on the stage an instance name, e.g: *wall*

4. Type the following ActionScript code on the timeline, in the first frame in which 3D Wall Pro appears:

```
import com.flashloaded.Wall3DPro.Wall3DEvent;
import flash.net.navigateToURL;
import flash.net.URLRequest;

wall.addEventListener(Wall3DEvent.CLICK,clickHandler);

function clickHandler(evt:Wall3DEvent):void{
  if(evt.element.state == "default") {
    var url:String = evt.element.getUserProperty("url");
    var request:URLRequest = new URLRequest(url);
    navigateToURL(request, "_blank");
  }
}
```

*Note: In this example, you would replace **wall** with the instance name of your 3D Wall Pro.*

You should now see that the URL opens when clicking on a thumbnail that has a URL assigned to it.

Toggling scrollbar visibility when zooming

This example is included with your download (scrollbar_tutorial.fla).

1. Set up an instance of the 3DWallPro as described in the [Getting Started](#) section of the userguide.
2. Select the 3DWallPro on the stage, press Ctrl+F3 (windows) or Cmnd+F3 (mac) to open the properties panel and give it the instance name **myWall**.
3. With the component still selected on the stage press Shift+F7 to open the component inspector panel. Set the showScrollbar parameter to false.
4. Press Ctrl+F7 (windows) or Cmnd+F7 (mac) to open the components panel, locate the **User Interface** folder and drag a copy of the **Slider** component onto the stage. Position the Slider component where you'd like it to appear and in the properties panel give it the instance name **myDragBar**.
5. With the Slider still selected open the component inspector panel and set the liveDragging parameter to true. Set the maximum value to 1 (leave the minimum value as 0). Set the value to 0.5, the snapInterval to 0.01 and the tickInterval to 0.01.
6. Create a new layer on your timeline, select the keyframe and press Alt+F9 to open the Actions panel. Enter the following script:

```
import fl.controls.Slider;
import fl.events.SliderEvent;
import com.flashloaded.Wall3DPro.*;

myDragBar.addEventListener(SliderEvent.CHANGE, dragHandler);

myWall.addEventListener(Wall3DEvent.CLICK, clickHandler);

function dragHandler(eo:SliderEvent):void {
    myWall.moveCameraTo(eo.value, myWall.cameraYPosition);
}

function clickHandler(eo:Wall3DEvent):void {
    myDragBar.visible = myDragBar.visible ? false : true;
}
```

The first three lines of this script import the classes that the script uses. The fourth and fifth lines create two event listeners that trigger functions when the Slider value is changed and when the 3DWallPro is clicked. The dragHandler function uses the moveCameraTo method to alter the position of the 3DwallPro when it is called. The clickHandler function toggles the visibility of the Slider instance as the wall zooms in and out.

7. Save and test your file.

Disabling mouse wheel zooming when viewing large images

You can disable mouse wheel zooming when viewing large images and re-enable it after returning to thumbnail view by placing the following code on the timeline:

```
import com.flashloaded.Wall3DPro.Wall3DEvent;

wall.addEventListener(Wall3DEvent.ELEMENT_SELECTED, disableScrolling);
wall.addEventListener(Wall3DEvent.ELEMENT_DESELECTED,
enableScrolling);

function disableScrolling(e:Wall3DEvent):void {
    wall.cameraActionOnScrollWheel = "no action";
}
function enableScrolling(e:Wall3DEvent):void {
    wall.cameraActionOnScrollWheel = "zoom";
}
```

*Note: In this example, you would replace **wall** with the instance name of your 3D Wall Pro.*

ActionScript properties

cameraXPosition:Number

Availability

Flash Player 9

Description

Property; a 0-1 value that relates to the percentage that the camera is along the wall. Values outside of this range will automatically be set to 0-1 (e.g. 1.5 is 0.5).

Example

```
3DWallinstance.cameraXPosition = 0.75;
```

cameraYPosition:int

Availability

Flash Player 9

Description

Property; the y value of the camera

Example

```
3DWallinstance.cameraYPosition = 100;
```

centrePoint:Number

Availability

Flash Player 9

Description

Property; the vertical centre point of the wall, useful for vertically centering the camera.

interactive:Boolean

Availability

Flash Player 9

Description

Property; this triggers whether or not the user can click or scroll the images. Useful for when any predefined animations are happening.

Example

```
3DWallinstance.interactive = false;
```

selectedElement:Wall3DElement

Availability

Flash Player 9

Description

Property; the Wall3DElement that is currently selected. Through the Wall3DElement class, it's possible to get the src, param, state, x, y, z, position in percentage along the wall, scaling, and rotationX, Y and Z. Smoothing can also be turned on and off individually.

Example

```
trace(wallInstance.selectedElement.src);
```

Values available

src, rowPos, columnPos, imageNumber, fullWidth, fullHeight, x, y, z, rotationX, rotationY, rotationZ, smooth, thumbnailScaleX, thumbnailScaleY

totalColumns:Number

Availability

Flash Player 9

Description

Property; this forces the total number of columns.

Example

```
3DWallinstance.totalColumns = 10;
```

wallEnabled:Boolean

Availability

Flash Player 9

Description

Property; sets the wall to be enabled or disabled.

Example

```
3DWallinstance.wallEnabled = false;
```

zoom:Number

Availability

Flash Player 9

Description

Property; gets or sets the zoom level. When set, it tweens it. When got, it returns the exact position of the camera at that exact moment, not the target position. Immediately after setting it, a get will produce a different result as it will still be tweening to that zoom level.

Example

```
3DWallinstance.zoom = 5;
```

Unsupported properties

The following properties are not supported by us as they are for advanced developers who wish to interact with the Papervision3D objects and complex calculations. These properties should be used with caution as incorrect usage might compromise the speed and performance of the component.

scene

camera

cameraFocalPointIndependent

Availability

Flash Player 9

Description

Properties; These three properties allow access to the actual Papervision3D objects. With knowledge of Papervision3D, it's possible to add additional 3D elements which can be manipulated using these variables. For example, you could have objects flying around the wall or sitting in the middle etc.

Note: Support questions relating to the use of the above these three properties should be directed to the Papervision3D forums.

wallWidth: Number

wallRadius: Number

Availability

Flash Player 9

Description

Properties; These properties are used to animate things along the curve of the wall.

Note: These properties are for advanced developers only and require the usage of complex trigonometrical calculations.

ActionScript methods

closeWall

Availability

Flash Player 9

Description

Method; closes 3D Wall Pro. The wall can either be instantly closed or animated while closing. Note: This may not necessarily remove the wall from memory.

Example

```
3DWallinstance.closeWall(); // closes the wall instantly
3DWallinstance.closeWall(true); // animates the closing of the wall
```

deselectElement

Availability

Flash Player 9

Description

Method; if an element is currently selected, this will zoom it out

Example

```
3DWallinstance.deselectElement();
```

filterXML

Availability

Flash Player 9

Description

Method; modifies the XML to only include the items that match those attribute name and value pairs. The attributes must be placed on the item tag of the XML. Note: This will only work if set before the wall initializes. It cannot be used after initialization.

Syntax

```
3DWallinstance.filterXML(filterName:String, filterValue:String);
```

Example

```
3DWallinstance.filterXML("animals", "dogs");
```

getAllElements

Availability

Flash Player 9

Description

Method; returns an array of all of the Wall3DElements

Example

```
3DWallinstance.getAllElements();
```

getElementsByParam

Availability

Flash Player 9

Description

Method; returns an array of Wall3DElements with the specified user defined parameter name and matching value. The user defined parameter name and value would be specified in the *item* tag of the XML.

Syntax

```
3DWallinstance.getElementsByParam(paramName:String,  
paramValue:String);
```

Example

```
3DWallinstance.getElementsByParam("favorite","yes");
```

getElementsByURL

Availability

Flash Player 9

Description

Method; returns an array of elements with the specified URL.

Syntax

```
3DWallinstance.getElementsByURL(url:String);
```

Example

```
3DWallinstance.getElementsByURL(images/photo1.jpg);
```

init

Availability

Flash Player 9

Description

Method; initializes or reinitializes the wall with the data from the specified XML file.

Syntax

```
3DWallinstance.init(xmlFilename:String);
```

Example

```
3DWallinstance.init("images.xml");
```

moveCamera

Availability

Flash Player 9

Description

Method; moves the camera by the specified number of rows and columns

Syntax

```
3DWallinstance.moveCamera(rowsToMove:int, columnsToMove:int);
```

Example

```
3DWallinstance.moveCamera(1, 3);
```

moveCameraTo

Availability

Flash Player 9

Description

Method; moves the camera to the specified X percentage (0-1) and the specified Y value.

Syntax

```
3DWallinstance.moveCameraTo(percentX:Number, y:int);
```

Example

```
3DWallinstance.moveCameraTo(0.3, 30);
```

showThumbnail

Availability

Flash Player 9

Description

Method; zooms in to the specified Wall3DElement. Takes a Wall3DElement as a parameter that would typically be retrieved through the getAllElements, getElementsByParam or getElementsByURL functions.

Syntax

```
3DWallinstance.showThumbnail(targetedElement:Wall3DElement);
```

Example

```
3DWallinstance.showThumbnail(elementArray[5]);
```

tilt

Availability

Flash Player 9

Description

Method; tilts the wall up or down

Syntax

```
3DWallinstance.tilt(amount:Number);
```

Example

```
3DWallinstance.tilt(100);
```

zoom

Availability

Flash Player 9

Description

Method; sets the amount to zoom the camera. The range should be generally between 0-6.

Syntax

```
3DWallinstance.zoom(amount:Number);
```

Example

```
3DWallinstance.zoom(0.2);
```

zoomAdjust

Availability

Flash Player 9

Description

Method; adds the passed amount to the current zoom (as opposed to setting it outright).

Syntax

```
3DWallinstance.zoom(amount:Number);
```

Example

```
3DWallinstance.zoom(0.2);
```

Help

This component is fully supported by the Flashloaded support team through our support forum. You will also find tips and additional information in the forum as well as announcements for version updates: [3D Wall Pro Support Forum](#)

Note: In order to post a question in the forum, you will need to [register](#) by creating a username and password. This registration differs from your account login.