3D Wall

User Guide revision 1.7

www.flashloaded.com

Table of Contents

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 component.

2. Unzip/extract the 3D Wall.zip file that you downloaded. You will find a file called 3D Wall.mxp. Double click on this file in order to install the component using Extension Manager.

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

Getting started

1. Having installed the 3D Wall using the Adobe Extension Manager, start a new Flash ActionScript 3 file and save it.

2. Prepare your images:
   The 3D Wall 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.

3. For the 3D Wall version 1.1.0 or higher, the image data can be defined either through XML or ActionScript. For versions prior to 1.1.0, only the XML option is available. At this stage, you should create the XML file which contains the image data. Please refer to the XML section of this userguide for instructions on creating the XML file or the section on Defining images using ActionScript.

4. Locate the 3D Wall 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 the 3D Wall 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

All of the images for the 3D Wall 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 imagesFolder="images/" thumbnailsFolder="thumbs/">
</gallery>
```

Edit the gallery entry as follows:

The ***imagesFolder*** element defines the folder containing the large images (ending with a "/")
The ***thumnailsFolder*** element defines the folder containing the thumbnails (ending with a "/")

4. Add the image tags to your XML file (the bold lines are the new additions)

Example where the large image and thumbnail images have the same filenames:

```
<?xml version="1.0" encoding="utf-8"?>
<gallery imagesFolder="images/" thumbnailsFolder="thumbs/">
    <img src="tree.jpg" width="650" height="400" />
    <img src="house.jpg" width="500" height="600" />
    <img src="car.jpg" />
</gallery>
```

Example where the large image and thumbnail images have the different filenames:

```
<?xml version="1.0" encoding="utf-8"?>
<gallery imagesFolder="images/" thumbnailsFolder="thumbs/">
    <img src="tree.jpg" thumbnailSrc="pic.jpg" width="650" 
height="400" />
    <img src="car.jpg" thumbnailSrc="pic2.jpg" width="500" 
height="600" />
    <img src="house.jpg" thumbnailSrc="house_th.jpg" />
</gallery>
```

Example where image sends a parameter which is read through ActionScript:

```
<?xml version="1.0" encoding="utf-8"?>
<gallery imagesFolder="images/" thumbnailsFolder="thumbs/">
    <img src="tree.jpg" name="tree" desc="Big tree" />
    <img src="house.jpg" name="house" desc="A house" />
    <img src="car.jpg" name="car" desc="Red car" />
</gallery>
```

src defines an image name for the large image.
width (optional) overrides the largeImageDefaultWidth setting for the individual image.
height (optional) overrides the largeImageDefaultHeight setting for the individual image.
thumbnailSrc (optional) defines the name of the thumbnail image only if is it different to the large image name.
name/desc (optional) any additional parameters can be added which can be read through an ActionScript event. In this example, we've added parameters called *name* and *car*.

5. 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*

6. 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 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.

7. 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 imagesFolder="images/" thumbnailsFolder="thumbs/" numRows="2"
wallCurvature="-360" >
```

Defining images through ActionScript

Since the 3D Wall version 1.1.0 it's possible to define the images through ActionScript.

1. Give the 3D Wall 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 images using XML elements:

```
wall.setMediaPaths("wallimages/", "wallimages/thumbs/");
wall.addElements("<img src='1.jpg' width='520' height='345'/>","XML");
wall.addElements("<img src='2.jpg' width='500' height='332'/>","XML");
wall.addElements("<img src='3.jpg' width='500' height='332'/>","XML");
wall.init();
```

In the above code:

setMediaPaths defines the folders containing the large and thumbnail images (ending with a "/") in this format: *setMediaPaths("LARGE IMAGE FOLDER/", "THUMB IMAGE FOLDER");*
addElements defines the XML element, following the same format as the image tags in the XML file.
init - after defining the images, the init method must be called to initialize the wall.

Method 2: Defining image objects elements

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

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

var obj2:Object = new Object();
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 (ending with a "/") in this format: *setMediaPaths("LARGE IMAGE FOLDER/", "THUMB IMAGE FOLDER");*
addElements adds the image object to the 3D Wall instance.
init - after defining the images, the init method must be called to initialize the wall.

Method 3: Defining an array of image objects

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

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

var obj2:Object = new Object();
obj2.src="2.jpg";
obj2.width=500;
obj2.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 (ending with a "/") in this format: *setMediaPaths("LARGE IMAGE FOLDER/", "THUMB IMAGE FOLDER");*
addElements adds the array of image objects to the 3D Wall instance.
init - after defining the images, 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 information.	images.xml
XMLNoCache	Defines whether to force the 3D Wall to use a non-cached version of the XML file. This is useful for situations where the XML file may be updated. *Note: This parameter must always be set to false when testing locally.*	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. *Note: In some cases, it might be best to set the camerRotationFactor to 1 (no rotation).*	-180
smoothImages	Sets whether the smoothing of images is controlled automatically by the 3D Wall or if smoothing is always on. When set in *auto* mode, smoothing is automatically turned off while animations are occurring. Setting this parameter to *on* may have an impact on performance on curved walls. Options available are: *auto*, *on* or *off*	auto
allowKeyboardControl	Sets whether to enable the arrow and space keys for navigation and zooming.	true
cameraActionOnScrollWheel	Sets whether mouse wheel scrolling should be enabled for zooming, cycling through the images 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 or not.	false

Parameter	Description	Example
showWarningMessages	Sets whether to display performance related settings warning messages when testing in the Flash IDE or not.	true

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: *0 = Starts from the left* *50 = Starts from the center* *100 = Starts from the right*	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

Parameter	Description	Example
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

Parameter	Description	Example
stageScrollVerticalLowerLimit	The y value of the lower limit of the stage scrolling. 0 marks the lowest image in the wall. *Note: This must be a negative value. For walls of -360 curvature, a setting of -100 is recommended.*	-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 *Note: Setting this to a low value will affect the tilt range.*	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. ** See easing styles*	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
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 preloadAllImagesBeforeShowing is set to true.	2

Parameter	Description	Example
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. ** See easing styles*	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
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

Parameter	Description	Example
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. *Note: The glow and depth of field settings can affect the spacing*	15
thumbnailSpacingVertical	The number of pixels vertically between each thumbnail. *Note: The glow and depth of field settings can affect the spacing*	15
thumbnailMaxWidth	The maximum width that the thumbnails will display at.	120
thumbnailMaxHeight	The maximum height that the thumbnails will display at.	80
thumbnailDoubleSided	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. *Note: Setting this to true impacts the performance.*	true
thumbnailSegmentsHorizontal	The number of horizontal segments that the thumbnails are divided into. Increase this number only if the images are too distorted. *Note: It is recommended to leave this at a setting of 1 wherever possible as higher numbers can impact performance.*	1

Parameter	Description	Example
thumbnailSegmentsVertical	The number of horizontal segments that the thumbnails are divided into. Increase this number only if the images are too distorted. *Note: It is recommended to leave this at a setting of 1 wherever possible as higher numbers can impact performance.*	1

Large image appearance

Parameter	Description	Example
scaleImageDown	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
largeImageDefaultWidth	The default width of the large images. This value can be overridden for individual images in the XML.	600
largeImageDefaultHeight	The default height of the large images. This value can be overridden for individual images in the XML.	450
largeImageZoomOutTime	The speed at which the large images return to the thumbnails.	1
largeImageZoomOutEasing	The easing style for the motion for which the large images return to the thumbnails. ** See easing styles*	easeOutQuad
largeImageZoomInTime	The speed at which the large images enlarge.	1
largeImageZoomInEasing	The easing style for the motion for which the large images enlarge. ** See easing styles*	easeOutQuad
largeImageTogglesOnDoubleClick	Sets whether the large image should open and close on single or double click.	false

Glow settings

Parameter	Description	Example
glowEnabled	Sets whether to display a glow around the large image and the last selected thumbnail or not.	true
glowColour	The color of the glow.	#FF0000
glowAlpha	The alpha percentage of the glow.	100
glowBlurX	The amount of blur of the glow in the X direction. *Note: This setting can also affect the space between the thumbnails*	8
glowBlurY	The amount of blur of the glow in the Y direction. *Note: This setting can also affect the space between the thumbnails*	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
preloadAllImagesBeforeShowing	Sets whether to preload all of the thumbnails before displaying anything. It is recommended to always set this to *true* 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.	100
reflectionGradientStartAlpha	The alpha setting for the side of the reflection closest to the thumbnail image.	100
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. *Note: The glow and depth of field settings can affect the reflection distance.*	0

* 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 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 the 3D Wall. 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 the 3D Wall, 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*
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.

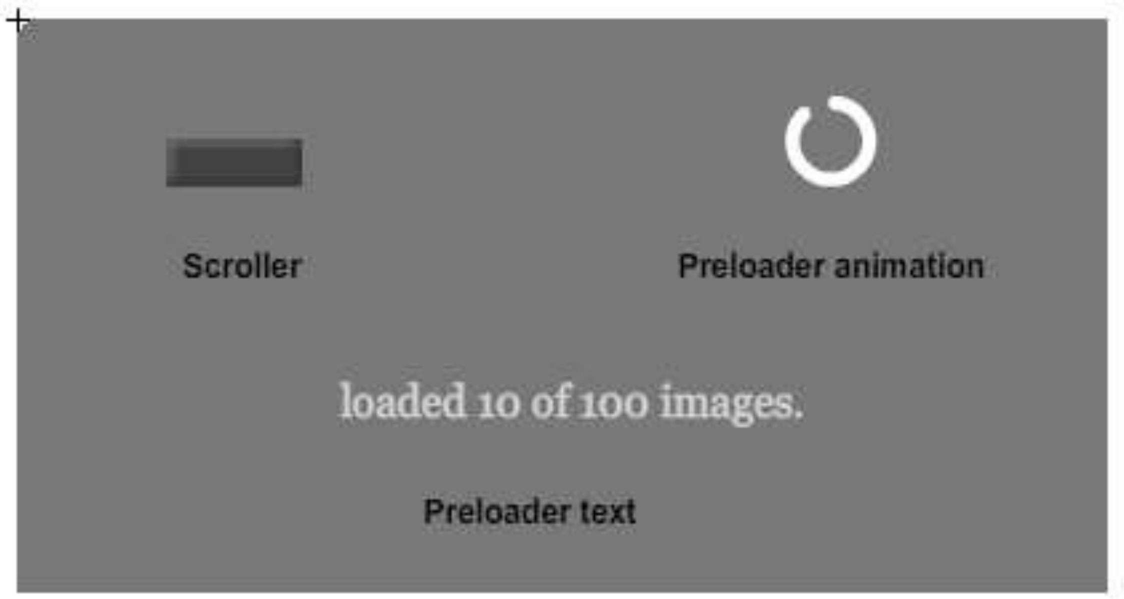
Resizing large images

The 3D Wall 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 the 3D Wall 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*.

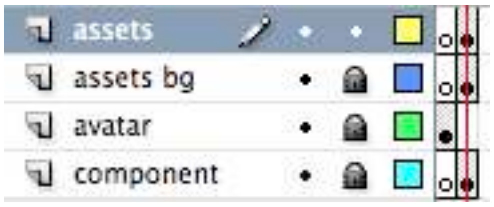
Skinning

The scrollbar, preloader animation and preloader text can be skinned to match your desired look and feel. Double click anywhere on the 3D Wall component that's on the stage in order to skin the 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 scrollbar

Double click on the *scrollbar* movie clip in order to edit the look or size 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.

ActionScript events

Events are called whenever the 3D Wall 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 image properties:

type: the type of event that was initiated (e.g: *CLICK*, *MOUSE_OVER*, *MOUSE_OUT*)
src: the source file of the image on which the event was initiated
param: the optional parameter for the image as defined in the *param* value in the XML
state: whether the image is in the default state or enlarged state (*Wall3DElement*.*STATE_DEFAULT* or .*STATE_ENLARGED*)

The following events are included:

Wall3DEvent*.*WALL_LOADED
Broadcasted when all the thumbnails have loaded and the animation sequence starts.

Wall3DEvent*.*INTRO_ANIM_STARTED
Broadcasted when the opening animation sequence starts.

Wall3DEvent*.*INTRO_ANIM_COMPLETE
Broadcasted when the opening animation sequence had completed.

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 image.

Wall3DEvent*.*DOUBLE_CLICK
Broadcasted when double clicking on a large or thumbnail image, provided that *largeImageTogglesOnDoubleClick* is enabled.

Wall3DEvent*.*MOUSE_OVER
Broadcasted when the viewer moves their mouse over a large or thumbnail image.

Wall3DEvent*.*MOUSE_OUT
Broadcasted when the viewer moves their mouse off a large or thumbnail image.

Wall3DEvent*.*IMAGE_SELECTED
Broadcasted when an image is selected. This event is recommended over the CLICK event as this will be called when the image is selected using the arrow keys.

Wall3DEvent*.*IMAGE_DESELECTED
Broadcasted when the selected image returns to being a thumbnail.

Wall3DScrollEvent.SCROLLBAR_OVER
Broadcasted when the mouse is placed over the scrollbar.

Wall3DScrollEvent.SCROLLBAR_OUT
Broadcasted when the mouse leaves the scrollbar area
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_STAGE
Broadcasted when scrolling is performed using the scrollbar.

Wall3DScrollEvent*.*SCROLL_TOOL_SCROLLBAR
Broadcasted when scrolling is performed by clicking and dragging the stage.

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

```
import com.flashloaded.Wall3D.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 + ", image state: " + e.image.state + ",
    src: " + e.image.src + ", optional parameter: " + e.image.param);
}
```

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

```
import com.flashloaded.Wall3D.Wall3DEvent;
3DWallinstance.addEventListener(Wall3DEvent.CLICK,clickHandler);

function clickHandler(evt:Wall3DEvent):void{
    trace(evt.image.src);
    gotoAndStop(evt.image.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 ***IMAGE_SELECTED*** event. This is how you would do this:

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

```
<img src="tree.jpg" width="650" height="400" title="Tree title" />
```

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 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 the 3D Wall appears:

```
import com.flashloaded.Wall3D.Wall3DEvent;
wall.addEventListener(Wall3DEvent.IMAGE_SELECTED,selectedHandler);
wall.addEventListener(Wall3DEvent.IMAGE_DESELECTED,deselectedHandler);

function selectedHandler(evt:Wall3DEvent):void{
    title_txt.text = evt.image.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 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:

```
<img src="tree.jpg" width="650" height="400"
url="http://www.flashloaded.com" />
```

2. Give the 3D Wall 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 the 3D Wall appears:

```
import com.flashloaded.Wall3D.Wall3DEvent;
import flash.net.navigateToURL;
import flash.net.URLRequest;

wall.addEventListener(Wall3DEvent.CLICK,clickHandler);

function clickHandler(evt:Wall3DEvent):void{
    if(evt.image.state == "default") {
        var url:String = evt.image.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.*

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

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;
```

selectedImage: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.selectedImage.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;
```

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 the 3D Wall. 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
```

deselectImage

Availability
Flash Player 9

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

Example
```
3DWallinstance.deselectImage();
```

getAllImages

Availability
Flash Player 9

Description
Method; returns an array of all of the Wall3DElements

Example
```
3DWallinstance.getAllImages();
```

getImagesByParam

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 ***img*** tag of the XML.

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

Example
```
3DWallinstance.getImagesByParam("favorite","yes");
```

getImagesByURL

Availability
Flash Player 9

Description
Method; returns an array of images with the specified URL.

Syntax
```
3DWallinstance.getImagesByURL(url:String);
```
Example
```
3DWallinstance.getImagesByURL(images/photo1.jpg);
```

init

Availability
Flash Player 9

Description
Method; initializes the wall. This method can be called with an XML file name specified in order to initialize the wall with that XML. Note: This method can only be called once, for first time initialization.

Syntax
```
3DWallinstance.init();
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 getAllImages, getImagesByParam or getImageByURL functions.

Syntax
```
3DWallinstance.showThumbnail(targetedImage:Wall3DElement);
```

Example
```
3DWallinstance.showThumbnail(imageArray[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 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.