



3DImageChanger version1.0

ADVANCED COMPONENT
FOR FLASH PROFESSIONAL 8

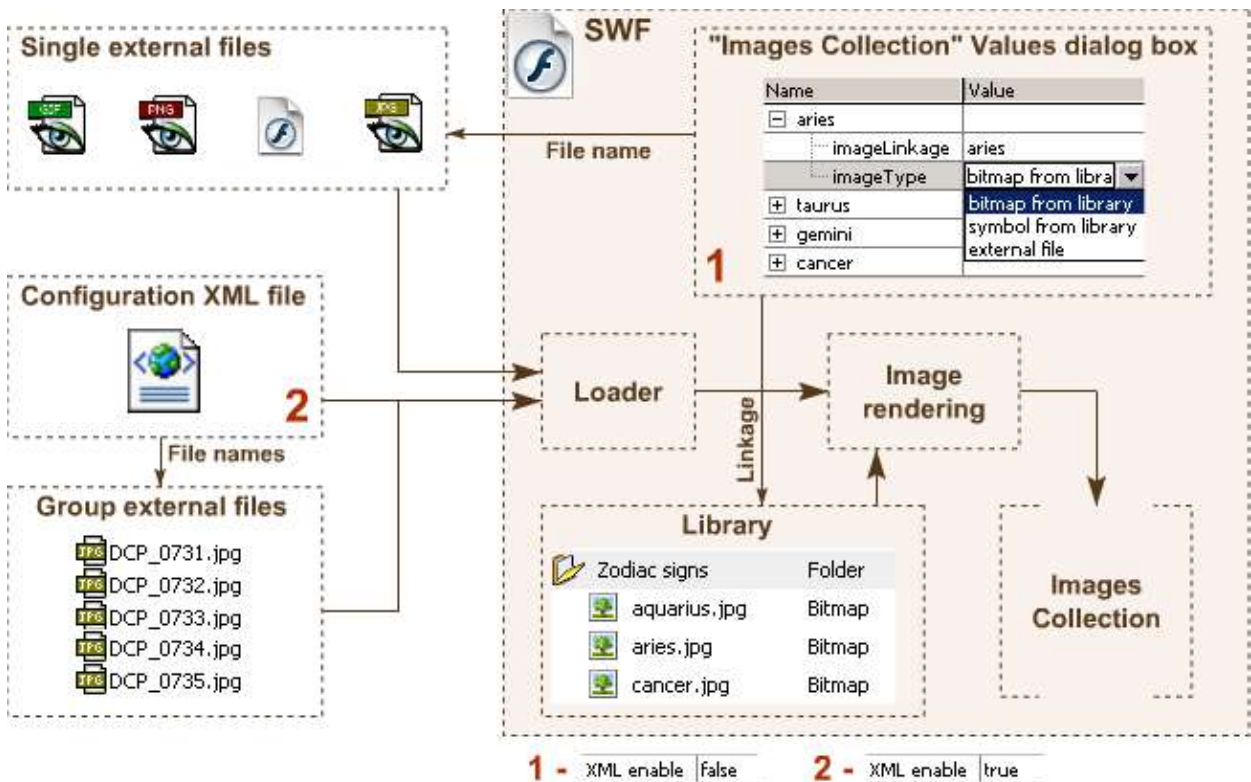
Overview

3DImageChanger is a revolutionary new Flash component based on advanced technology that allows changing images on screen by means of their 3-dimensional rotation. The component's exclusive system provides an easy and understandable way for adding graphic objects of different types to internal "Images Collection". You can attach not only the external graphic files but also symbols and bitmaps from library to your collection. Full-featured set of parameters and ActionScript methods, properties and events, in addition, turn the component into a powerful and universal tool for creating Rich Internet Applications.

3DImageChanger is a complex component that consists of three functional parts:

1. Image composition system.
2. Navigation system.
3. 3D effects creation system.

Image composition system is developed to carry out all actions on loading (if necessary) and rendering attached graphic objects with further adding to the "Images Collection". The "Images Collection" is the 2-dimensional array where images are stored as BitmapData objects.



You can attach images to collection in three different ways of your choice:

- Using the **Images Collection** authoring parameter in the Property inspector or in the Component inspector.
- Creating **configuration XML file** for loading external graphic files (JPEG, GIF, PNG, SWF).
- Using the ActionScript method `addImage()` to add single images of supported types to collection.

Also, you can choose from a number of ActionScript methods and properties to manage images in collection. For more information, see [ThreeDimensionalChanger class](#).

Navigation system provides built-in navigation and sequencing capabilities, and lets you easily step through the objects that an “Images collection” contains. Images in collection maintain “state” that allows the user to advance to the next or previous image:

- When the next or previous image is shown, it becomes the current image. The current image, at the same time, loses its “current” state and is hidden.

Use ActionScript methods and properties to build advanced navigation through images in collection.

When a 3DImageChanger instance has focus and the **Keyboard Control** parameter is set to true, you can use the following keys to control it:

Key	Description
Up Arrow / Right Arrow	Navigates to the next image in the “Images Collection”.
Down Arrow / Left Arrow	Navigates to the previous image in the “Images Collection”.
Home	Navigates to the first image in the “Images Collection”.
End	Navigates to the last image in the “Images Collection”.
Spacebar	Toggles the displayMode property from "manual" to "auto-forward" / "auto-backward" and vice versa

For more information about controlling focus, see “FocusManager class” in *Macromedia Flash 8 Components Language Reference Help*.

3D effects creation system is the core of the component and assigned for 3D modeling depending on selected type of change effect. 3D rotation effects reproduce rotation of specified geometrical figures. 3DImageChanger includes the following five effects:

1. *Prismatic quadrangular* – represents 3D rotation of a quadrangular right prism around the axis that passes through the centers of its bases. The visible adjacent faces of prism are represented by neighboring images from collection.
2. *Prismatic triangular* – represents 3D rotation of a triangular right prism around the axis that passes through the centers of its bases. The visible adjacent faces of prism are represented by neighboring images from collection.
3. *Cylindrical convex* – represents 3D rotation of an external convex cylindrical surface around the axis of cylinder. The visible part of the surface is represented by neighboring images from collection.
4. *Cylindrical concave* – represents 3D rotation of an internal concave cylindrical surface around the axis of cylinder. The visible part of the surface is represented by neighboring images from collection.
5. *Plane rectangular* – represents 3D rotation of a rectangle around the axis that passes through the middles of its opposite sides. The area of rectangle is represented by corresponding image from collection.

The following image depicts the effects described above:



prismatic quadrangular



prismatic triangular



plane rectangular



cylindrical convex



cylindrical concave



original image

The table below contains the real dimensions of static image in 3DImageChanger (relative to the original image) for a specified effect in the published SWF file. The dimensions in the table are listed for two possible values of the Rotation Plane parameter: "horizontal" (as shown in the illustration above) and "vertical".

Dimensions	Change Effect				
	prismatic quadrangular	prismatic triangular	cylindrical convex	cylindrical concave	plane rectangular
<i>Horizontal Rotation Plane</i>					
width	100%	100%	63.66%	63.66%	100%
height	100%	100%	100%	100%	100%
<i>Vertical Rotation Plane</i>					
width	100%	100%	100%	100%	100%
height	100%	100%	63.66%	63.66%	100%

Please, keep in mind the following limitations when using 3DImageChanger:

- All features of 3DImageChanger component are available both in Flash Basic 8 and Flash Professional 8, except one – the “[Images Collection](#)” authoring parameter. In Flash Basic 8, you can not open the Values dialog box from “Images Collection” text box within the Parameters tab to manipulate the items in the collection while authoring. This is because of the Collection class (mx.utils.Collection) that is available in Flash Professional only. However, you can work around this limitation by using the [addImage\(\)](#) method instead of the “Images Collection” parameter in your Flash Basic 8 applications.
- The images you add to the collection can be both static and animated, but in the published SWF file all of them become static. That means, for example, if you attach a movie clip to the collection and this movie clip contains more than one frame – only the first frame will be shown.
- Although you can add images of different size to the collection, in the published SWF file all of them are scaled larger or smaller, if needed, to fit the “base dimensions”. For more information, see [Sizing 3DImageChanger component](#).

Using 3DImageChanger

3DImageChanger can be used in various applications where the images should be changed in the certain order, at a specified interval or depending on user interactions, etc. Using parameter settings in the authoring environment and ActionScript methods, properties and events available for the component, you can build advanced and full-featured image management systems. Built-in 3D rotation modeling system will create unique and attractive appearance for different types of Flash applications.

Use 3DImageChanger to create any of the following:

- Photo gallery
- Animated menu bar
- Slide show with Venetian blind effect
- Image rollovers

This is not the complete list of possible usages, just ones that are likely to be used most often. Your creative power and imagination can greatly extend the scope of the 3DImageChanger component.

3DImageChanger parameters

You can set the following authoring parameters for each 3DImageChanger component instance in the Component inspector or the Property inspector:

Change Effect: Sets the type of 3D rotation effect that is used for changing an image. This parameter can be one of five predefined values: “prismatic quadrangular”, “prismatic triangular”, “cylindrical convex”, “cylindrical concave”, or “plane rectangular”; the default value is “prismatic quadrangular”. For more information, see [3D effects creation system](#).

Change Effect	prismatic quadrangular
Delay (seconds)	prismatic quadrangular
Display Mode	prismatic triangular
Granularity (pix...	cylindrical convex
Images Collection	cylindrical concave
Keyboard Control	plane rectangular
Keyboard Control	false

Delay (seconds): A number that indicates (in seconds) how long each image “holds” before automatically displaying the next image in que. This parameter accepts any positive number or 0 (including fractional values) and is enabled at Display Mode parameter set to “auto-forward” or “auto-backward”. If Delay is equal to 0, the images are displayed continuously. The default value is 3.

Delay (seconds)	3
Display Mode	manual
Granularity (pix...	10
Images Collection	[]
Keyboard Control	false
Lock Mode	false

Note: If you apply a [visual effect](#) to a 3DImageChanger instance, the delay starts at the end of a “Fade-Out” phase and lasts to the beginning of the next “Fade-In” phase.

Display Mode: This controls the automatic playback of your “Images Collection” when first loaded by a user. By default, it is set to “manual,” which means 3DImageChanger will display the first image, but will not set the Delay timer (see above). It will simply stop and await user input. Changing this to “auto-

forward” or “auto-backward” will instruct 3DImageChanger to automatically display the next image in que after the Delay timer has expired.

Display Mode	manual
Granularity (pix...	manual
Images Collection	auto-forward
Keyboard Control	auto-backward
Lock Mode	false
Preloader Alpha	100

Note: “Auto-forward” means that the first loaded image will be the first one from the collection; the next loaded will be the second one from the collection, and so on. “Auto-backward” means that the first loaded image will be the last one from the collection; the next loaded will be the next-to-last one from the collection, and so on.

Granularity (pixels): Defines the degree of detail and the rendering quality while rotating an image. In numerical expression, it is the height or the width (depending on whether the Rotation Plane parameter is set to “vertical” or “horizontal”) of segments into which the image is split when rotating.

Granularity (pixels)	10
Images Collection	7
Keyboard Control	8
Lock Mode	9
Preloader Alpha	10
Preloader Color	11
	12

The smaller the segments, the more the image is smoothed and anti-aliased, and vice versa. At the same time, reducing the value of Granularity, you slow down the speed of rotation by increasing the time of rendering. This parameter can be an integer between 1 and 100 pixels. The default value is 10.

Images Collection: An object type parameter that contains two values to be set: imageLinkage and imageType. It allows adding, modifying, moving, and deleting the “Images Collection” items in the Values dialog box while authoring. The “Images Collection” is the array of images you can manipulate within 3DImageChanger.

The screenshot shows the 'Values' dialog box with a tree view on the left and a detailed view on the right. The tree view shows a folder '- linkage -' containing 'imageLinkage' and 'imageType'. The 'imageType' dropdown is open, showing options: 'bitmap from library', 'symbol from library', and 'external file'. The detailed view on the right shows the following parameters:

Images Collection	[]
Keyboard Control	false
Lock Mode	false
Preloader Alpha	100
Preloader Color	#FFFFFF
Preloader Location	center

To define a collection item click the Add (+) button. To add, modify, move and delete collection items click the Add (+), Delete (-), and arrow buttons.


imageLinkage Can be either (a) identifier for an asset in the library (bitmap or symbol); or (b) the absolute or relative URL of the JPEG, GIF, PNG, or SWF file to be loaded. A relative path must be relative to the SWF file at level 0. Absolute URLs must include the protocol reference, such as http://.

imageType Specifies the type and location of the image you want to add to your collection. This parameter can be one of three predefined values: “bitmap from library”, “symbol from library”, “external file”. The default value is “bitmap from library”.

The image type must correspond to the image linkage value. Otherwise, the image will not be loaded to 3DImageChanger.

Note: The “Images Collection” parameter is enabled in Flash Professional 8 only (because of the Collection interface). In Flash Basic 8, use the `addImage()` method instead of this parameter.


Keyboard Control: Determines whether 3DImageChanger uses default keyboard handling to navigate through the images in collection when the component instance has focus (see [Navigation system](#) for more information). This parameter can be either “true” or “false”. The default value is “false”.

Keyboard Control	false
Lock Mode	false
Preloader Alpha	true
Preloader Color	#99CC33 
Preloader Location	center
Preloader Mode	always visible

Lock Mode: Turns on/off the component’s ability to “store” user interactions (“true” or “false”). That means 3DImageChanger can react differently to navigation commands (such as “gotoNextImage”, etc.) while changing (rotating) images:


true 3DImageChanger ignores any navigation command.

false 3DImageChanger stores up to 3 navigation commands running every next of them after the previous one has been completed. If the commands are opposite to each other in direction (for example, “gotoNextImage” – “gotoPreviousImage”), 3DImageChanger reverses rotation immediately not waiting for the end of the movement.

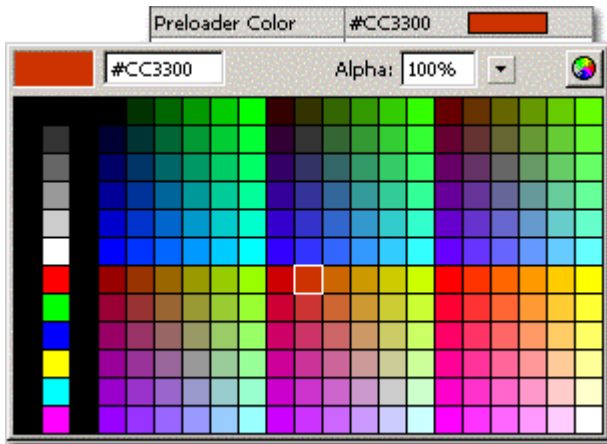
Lock Mode	false
Preloader Alpha	false
Preloader Color	true 
Preloader Location	center
Preloader Mode	always visible
Preloader Type	progress circle

The default value is “false”.

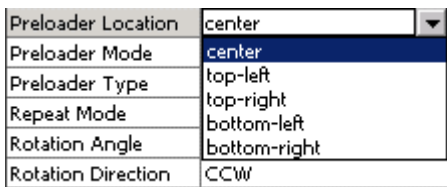
Preloader Alpha: The transparency percentage for the preloader animation. The parameter accepts any number at or between 0 and 100. The default value is 100.

Preloader Alpha	100
Preloader Color	#FFFFFF 
Preloader Location	center
Preloader Mode	always visible
Preloader Type	progress circle
Repeat Mode	true

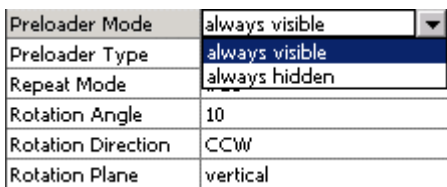
Preloader Color: The color of the preloader animation. This parameter accepts hex values. The default value is “#FFFFFF”.



Preloader Location: Specifies the location of the preloader in relation to the static image borders. This parameter can be one of five predefined values: “center”, “top-left”, “top-right”, “bottom-left”, or “bottom-right”; the default value is “center”.

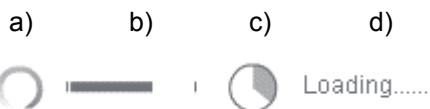


Preloader Mode: Indicates whether the preloader is visible or hidden when loading images. This parameter can be one of two predefined values: “always visible” or “always hidden”. The default value is “always visible”.



Note: Preloader animation is displayed certainly for external graphic files only. The loading progress of an XML file is also reflected if its size exceeds 20 KB.

Preloader Type: Specifies the preloader appearance. This parameter can be one of four predefined values: (a) “spiral”, (b) “progress bar”, (c) “progress circle”, (d) “text string”. The default value is “progress circle”.



Preloader Type	progress circle
Repeat Mode	spiral
Rotation Angle	progress bar
Rotation Direction	progress circle
Rotation Plane	text string
Scale Content	vertical
	fit image

Repeat Mode: This parameter instructs 3DImageChanger how to behave when reaching the final image in the “Images Collection”. There are two values available:

false Auto-playback stops and Display Mode changes to “manual” (if previously set to “auto-forward” or “auto-backward”).

true The collection restarts with either the first image (Display Mode set to “auto-forward” or “manual”), or the last image (Display Mode set to “auto-backward”).

Repeat Mode	true
Rotation Angle	false
Rotation Direction	true
Rotation Plane	vertical
Scale Content	fit changer
Visual Effect	none

The default value is “true”.

Rotation Angle: This parameter specifies the degree of rotation of images while their 3-dimensional rotation. The smaller the angle, the more the rotation is smooth, and vice versa. At the same time, reducing the value of Rotation Angle, you slow down the speed of rotation by increasing the time of rendering.

Rotation Angle	10
Rotation Direction	8
Rotation Plane	9
Scale Content	10
Visual Effect	12
Visual Effect Quality	15
	18

You can select any number from the list of predefined values for this parameter; but the real value of rotation angle will be the one from the table below that is the closest to the selected number for correspondent Change Effect.

Change Effect	Possible values for Rotation Angle, deg.																					
	1	2	3	4	5	6	8	9	10	12	15	18	20	30	36	45	60	90	120	180		
prismatic quadrangular	1	2	3	-	5	6	-	9	10	-	15	18	-	-	30	-	-	45	-	90	-	-
prismatic triangular	1	2	3	4	5	6	8	-	10	12	15	-	20	24	30	-	40	-	60	-	120	-
cylindrical convex	1	2	3	4	5	6	-	9	10	12	15	18	20	-	30	36	-	45	60	90	-	180
cylindrical concave	1	2	3	4	5	6	-	9	10	12	15	18	20	-	30	36	-	45	60	90	-	180
plane rectangular	1	2	3	4	5	6	-	9	10	12	15	18	20	-	30	36	-	45	60	90	-	180

The default value for Rotation Angle is 10.

Rotation Direction: The conditional parameter that specifies the direction of rotation while changing images. There are two values available:

CCW (Counterclockwise): top-down (for Rotation Plane set to “vertical”) or from left to right (for Rotation Plane set to “horizontal”).

CW (Clockwise): bottom-up (for Rotation Plane set to “vertical”) or from right to left (for Rotation Plane set to “horizontal”).

This is valid for the Display Mode parameter set to “auto-forward” or “manual” when moving forward through the “Images Collection” (to every next image). If the Display Mode parameter is set to “auto-backward”, the directions of rotation are opposite of listed above. Also, if moving backward through the “Images Collection” (to every previous image), the directions of rotation are opposite of listed above.

Rotation Direction	CCW
Rotation Plane	CW
Scale Content	CCW
Visual Effect	none
Visual Effect Quality	medium
Visual Effect Speed	medium

The default value is “CWW”.

Rotation Plane: Defines the plane of rotation while changing images. This parameter can be either “horizontal” or “vertical”. The default value is “vertical”.

Rotation Plane	vertical
Scale Content	horizontal
Visual Effect	vertical
Visual Effect Quality	medium
Visual Effect Speed	medium
XML enable	true

Scale Content: Indicates whether the content scales to fit the changer (“fit changer”), or the changer scales to fit the content (“fit image”). The default value is “fit image”.

Scale Content	fit changer
Visual Effect	fit changer
Visual Effect Quality	fit image
Visual Effect Speed	medium
XML enable	true
XML file Path	images.xml

Keep in mind that setting Scale Content to “fit image” causes the changer scales to fit the *first displayed image*. This is either the *first image* from the “Images Collection” (Display Mode set to “auto-forward” or “manual”), or the *last image* from the “Images Collection” (Display Mode set to “auto-backward”). This matters if the first and the last images in the collection have different size.

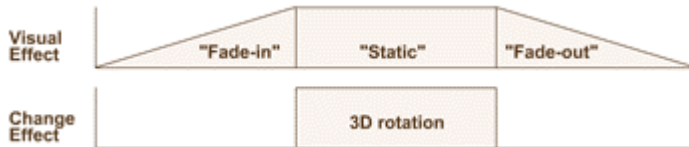
Note: Using the width and height attributes in a configuration XML file (the XML enable parameter is set to “true”) disables Scale Content effect. For more information, see the [XML file Path](#) parameter.

Visual Effect: This applies an additional visual effect to the component. You can select from four predefined values: “none”, “blur”, “grayscale”, “brightness”. The default value is “none” (no visual effect applied to a 3DImageChanger instance).

Visual Effect	none
Visual Effect Quality	none
Visual Effect Speed	blur
XML enable	grayscale
XML file Path	brightness
	images.xml

The visual effect has 3 phases:

- *Fade-in* Increasing – precedes a 3D rotation effect;
- *Static* Constant, maximum – continues while images rotating;
- *Fade-out* Decreasing – follows a 3D rotation effect.



Visual Effect Quality: Determines the depth or intensity of the visual effect. This parameter can be one of three predefined values: “low”, “medium”, “high”; the default value is “medium”.

Visual Effect Quality	medium
Visual Effect Speed	low
XML enable	medium
XML file Path	high
	images.xml

Visual Effect Speed: Determines the speed of increasing and decreasing of the visual effect. In other words, this parameter specifies the duration of “Fade-in” and “Fade-out” phases of the visual effect. You can select from five predefined values: “minimum”, “low”, “medium”, “high”, “maximum”; the default value is “medium”.

Visual Effect Speed	medium
XML enable	minimum
XML file Path	low
	medium
	high
	maximum

XML enable: This parameter defines the method that is used for adding graphic objects to internal “Images Collection”. There are two options available:

false The user adds and edits the items in the Values dialog box (opened from Images Collection text box within the Parameters tab for 3DImageChanger component).

true The user specifies the name of a configuration XML file that contains the list of graphic files to be loaded into 3DImageChanger.

If you choose one method, the other is ignored by the component, and vice versa.

XML enable	false
XML file Path	false
	true

The default value is “false”.

XML file Path: The name of configuration XML file. The default value is “images.xml” (in this case, the XML file must be saved in the same directory as the SWF file that contains a 3DImageChanger component).

XML enable	true
XML file Path	images.xml

XML is the glue between 3DImageChanger and your images. This is the basic template of your configuration XML file:

```
<?xml version="1.0" encoding="UTF-8"?>
<collection path="pictures/" width="300" height="150">
  
  
  
  .....
</collection>
```

The XML document can contain an unlimited number of img nodes. Each img node represents an image object and should have one required attribute:

src An attribute that defines the file name of an image.

Note: You can also add your own attributes in the img node that can be then easily accessible through the component’s `xml` property.

For every additional image you wish to add to the “Images Collection”, create another img node and fill it with the file name.

The collection element has the following three optional attributes:

path (Optional) Specifies the relative path to the folder that contains images. Not required if images are saved in the same directory as the SWF file that contains a 3DImageChanger component.

width (Optional) Sets the base width for images while displaying them in 3DImageChanger.

height (Optional) Sets the base height for images while displaying them in 3DImageChanger.

Note: You must specify both width and height attributes for the settings to take effect. Using these attributes disables Scale Content effect.

Sizing 3DImageChanger component

You can transform a 3DImageChanger component horizontally and vertically while authoring and at runtime. While authoring, select the component on the Stage and use the Free Transform tool or any of the Modify > Transform commands. At runtime, use the setSize() method. The sizing behavior of the 3DImageChanger component is controlled by the scaleContent property. When scaleContent is “fit changer”, the content is scaled to fit within the bounds of the changer (and is rescaled when setSize() is called). The “base dimensions” of the component, in this case, are determined by its size on the Stage while authoring. When scaleContent is “fit image”, the size of the component is fixed to the size of the content and setSize() has no effect. The “base dimensions”, in this case, are determined by the size of the first displayed image.

If you specify the width and height attributes in configuration XML file, the “base dimensions” are set to the values of these attributes, and not depend on scaleContent value.

The images in collection can be different in size. However, when displaying in 3DImageChanger, they are scaled larger or smaller, if needed, to fit the “base dimensions”.

The hit area of a 3DImageChanger component is designated by its “base dimensions”.

Creating an application with 3DImageChanger component

The following procedure explains how to add 3DImageChanger component to an application while authoring.

Example 1

In this example, 3DImageChanger is used to create a simple banner for a web application.

Before creating an application, download the images required for this example from the following URLs and save them to your hard disk:

<http://e-merald.com/pic/components/emerald01.jpg>

<http://e-merald.com/pic/components/emerald02.jpg>

<http://e-merald.com/pic/components/emerald03.jpg>

<http://e-merald.com/pic/components/emerald04.jpg>

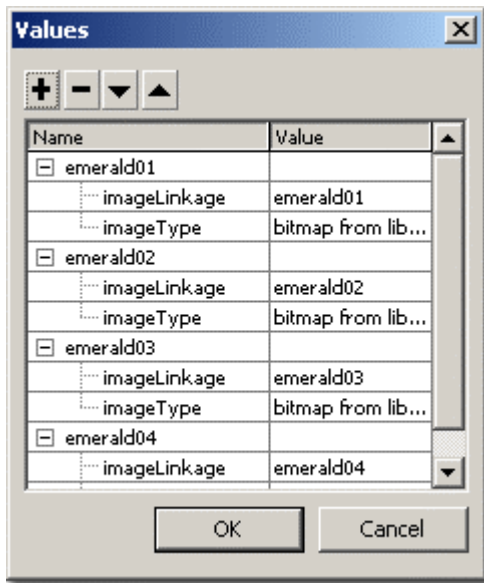
To create an application with 3DImageChanger component:

1. Set the Stage size to 200 x 195 pixels and leave white as background color. For frame rate enter any appropriate value – for example, 30 fps.
2. Drag one 3DImageChanger component from the Components panel to the Stage and set its coordinates to (20, 20).

Note: As the visual area of a 3DImageChanger instance increases while 3D rotation effect, leave enough space around the component on the Stage to avoid trimming the edges of images in the published SWF file.

3. Import the saved JPEG files into the library for the current Flash document by selecting File > Import to Library.
4. For each of the four imported bitmaps, do the following: select it in the Library panel, open the Linkage dialog box from the Library pop-up menu, and enter the corresponding linkage identifier from the following – “emerald01”, “emerald02”, “emerald03”, and “emerald04”.
5. In the Component inspector, do the following:

For Change Effect parameter, set: “prismatic triangular”
 For Delay (seconds) parameter, set: 2
 For Display Mode parameter, set: “auto-forward”
 For Granularity (pixels) parameter, set: 5
 For Rotation Angle parameter, set: 9
 For Visual Effect parameter, set: “blur”
 For Visual Effect Quality parameter, set: “low”
 For Visual Effect Speed parameter, set: “low”
 For Images Collection parameter as the object, add the following values by clicking the Add (+) button in the Values dialog box:



Leave the other parameters without changes.

Note: The “Images Collection” parameter is enabled in Flash Professional 8 only (because of the Collection interface). To add the above images to the “Images Collection” in your Flash Basic 8 application, do the following: (a) in the Property inspector, enter appropriate value for the Instance Name property of the component instance (for example, “changer”); (b) write the following code on Frame 1 of the Timeline:

```
changer.addImage("emerald01", "bitmap");
changer.addImage("emerald02", "bitmap");
changer.addImage("emerald03", "bitmap");
changer.addImage("emerald04", "bitmap");
```

6. Select Control > Test Movie.

Example 2

In this example, 3DImageChanger is used to create a slide show with a “concave surface” effect. The images are loaded as external JPEG files listed in configuration XML file.

To create an application with 3DImageChanger component:

1. Set the Stage size according to the size of images in your collection (in this case, 500 x 300 pixels) and choose the background color. For frame rate enter any appropriate value – for example, 30 fps.

2. Drag one 3DImageChanger component from the Components panel to the Stage and set appropriate coordinates for it.
3. In the Component inspector, do the following:
 - For Change Effect parameter, set:“cylindrical concave”
 - For XML enable parameter, set:.....“true”
 - For XML file Path parameter, set:.....“images.xml”

Change the values of other parameters depending on your needs.

4. In the Property inspector, enter appropriate value for the Instance Name property of the component instance (for example, “changer”).
5. Create two buttons – “Next Image” and “Previous Image” – and place them on the Stage. In the Property inspector, enter appropriate values for the Instance Name property of the component instances (for example, “next_button” and “prev_button”).
6. Write the following code on Frame 1 of the Timeline:

```
next_button.onRelease = function() {
    changer.gotoNextImage();
}
prev_button.onRelease = function() {
    changer.gotoPreviousImage();
}
```

7. Using your favorite text editor (Dreamweaver, UltraEdit, or even Notepad is fine), create a new document and enter the following code (in this case):

```
<?xml version="1.0" encoding="UTF-8"?>
<collection path="Nature_images/">
  
  
  
  
  
  
  
  
  
  
</collection>
```

This is your configuration XML document. Save the XML file in the same directory as the SWF file that contains a 3DImageChanger component and name it “images.xml”.

Now for the images – they must be saved in “Nature_images” directory, in this case, which in turn must be alongside the XML document.

For more information on structure of configuration XML file, see the [XML file Path](#) parameter.

8. Publish the Flash document with appropriate name.

ThreeDimensionalChanger class

Inheritance MovieClip > UIObject class > ThreeDimensionalChanger class

Setting a property of the ThreeDimensionalChanger class with ActionScript overrides the parameter of the same name set in the Property inspector or Component inspector.

Method summary for the ThreeDimensionalChanger class

The following table lists methods of the ThreeDimensionalChanger class.

Method	Description
<code>ThreeDimensionalChanger.addImage()</code>	Adds a new image to the end of the “Images Collection” or inserts it after the specified index in the array.
<code>ThreeDimensionalChanger.applyVisualEffect()</code>	Applies a specified visual effect to images in 3DImageChanger.
<code>ThreeDimensionalChanger.clear()</code>	Removes all images from the “Images Collection”.
<code>ThreeDimensionalChanger.getImageAt()</code>	Returns an image within the “Images Collection” by using its index.
<code>ThreeDimensionalChanger.getImageIndex()</code>	Finds an image within the “Images Collection” and returns its index.
<code>ThreeDimensionalChanger.gotoFirstImage()</code>	Navigates to the first image in the “Images Collection”.
<code>ThreeDimensionalChanger.gotoImageAt()</code>	Navigates to the image in the “Images Collection” with the specified index.
<code>ThreeDimensionalChanger.gotoLastImage()</code>	Navigates to the last image in the “Images Collection”.
<code>ThreeDimensionalChanger.gotoNextImage()</code>	Navigates to the next image in the “Images Collection”.
<code>ThreeDimensionalChanger.gotoPreviousImage()</code>	Navigates to the previous image in the “Images Collection”.
<code>ThreeDimensionalChanger.removeImageAt()</code>	Removes a single image from the “Images Collection” by using its index.
<code>ThreeDimensionalChanger.reset()</code>	Resets a 3DImageChanger instance to its initial state.
<code>ThreeDimensionalChanger.reverse()</code>	Reverses the direction of rotation while rotating images.
<code>ThreeDimensionalChanger.toggleDisplayMode()</code>	Toggles the displayMode property from "manual" to "auto-forward" / "auto-backward" and vice versa.
<code>ThreeDimensionalChanger.togglePlayMode()</code>	Toggles the play mode of a 3DImageChanger instance from “play” to “stop” and vice versa.

Methods inherited from the UIObject class

The following table lists the methods the ThreeDimensionalChanger class inherits from the UIObject class. When calling these methods from the 3DImageChanger object, use the form *3DImageChangerInstance.methodName*.

Method	Description
UIObject.getStyle()	Gets the style property from the style declaration or object.
UIObject.move()	Moves the object to the requested position.
UIObject.setSize()	Resizes the object to the requested size.
UIObject.setStyle()	Sets the style property on the style declaration or object.

Property summary for the ThreeDimensionalChanger class

The following table lists properties of the ThreeDimensionalChanger class.

Property	Description
ThreeDimensionalChanger.changeEffect	Specifies the type of a 3D rotation effect that is used for changing an image.
ThreeDimensionalChanger.currentImage	Read-only; returns the current image in the “Images Collection”.
ThreeDimensionalChanger.delay	A number that indicates (in seconds) how long each image “holds” before automatically displaying the next image in que.
ThreeDimensionalChanger.direction	Specifies the direction of rotation while changing images.
ThreeDimensionalChanger.displayMode	Controls the automatic playback of the “Images Collection”.
ThreeDimensionalChanger.firstImage	Read-only; returns the first image in the “Images Collection”.
ThreeDimensionalChanger.granularity	Defines the degree of detail and the rendering quality while rotating an image.
ThreeDimensionalChanger.keyboardControl	Determines whether 3DImageChanger uses default keyboard handling to navigate through the images in collection when the component instance has focus.
ThreeDimensionalChanger.lastImage	Read-only; returns the last image in the “Images Collection”.
ThreeDimensionalChanger.length	Read-only; the number of images in the “Images Collection”.
ThreeDimensionalChanger.lockMode	Determines whether 3DImageChanger is enabled or disabled to “store” user interactions.
ThreeDimensionalChanger.nextImage	Read-only; returns the next image in the “Images Collection”.
ThreeDimensionalChanger.preloaderAlpha	A number that indicates the transparency percentage for the preloader animation.

<code>ThreeDimensionalChanger.preloaderColor</code>	The color of the preloader animation.
<code>ThreeDimensionalChanger.preloaderLocation</code>	Specifies the location of the preloader in relation to the static image borders.
<code>ThreeDimensionalChanger.preloaderMode</code>	Determines whether the preloader is visible or hidden when loading images.
<code>ThreeDimensionalChanger.preloaderType</code>	Specifies the preloader appearance.
<code>ThreeDimensionalChanger.previousImage</code>	Read-only; returns the previous image in the "Images Collection".
<code>ThreeDimensionalChanger.repeatMode</code>	Instructs 3DImageChanger how to behave when reaching the final image in the "Images Collection".
<code>ThreeDimensionalChanger.rotation</code>	A number that indicates the degree of rotation of images while their 3D rotation.
<code>ThreeDimensionalChanger.rotationPlane</code>	Defines the plane of rotation while changing images.
<code>ThreeDimensionalChanger.scaleContent</code>	Determines whether the content scales to fit the changer, or the changer scales to fit the content.
<code>ThreeDimensionalChanger.visualEffect</code>	Specifies the type of a visual effect to be applied to a component instance.
<code>ThreeDimensionalChanger.visualEffectQuality</code>	Determines the depth or intensity of the visual effect.
<code>ThreeDimensionalChanger.visualEffectSpeed</code>	Determines the speed of increasing and decreasing of the visual effect.
<code>ThreeDimensionalChanger.xml</code>	Read-only; returns the XML object that represents the content of the downloaded configuration XML file.
<code>ThreeDimensionalChanger.xmlEnable</code>	Defines the method that is used for adding graphic objects to internal "Images Collection".
<code>ThreeDimensionalChanger.xmlFilePath</code>	The name of configuration XML file.

Properties inherited from the UIObject class

The following table lists the properties the ThreeDimensionalChanger class inherits from the UIObject class. When accessing these properties from the 3DImageChanger object, use the form *3DImageChangerInstance.propertyName*.

Property	Description
<code>UIObject.height</code>	The height of the object, in pixels. Read-only. Enabled at runtime after publishing SWF file containing the component.
<code>UIObject.left</code>	The left edge of the object, in pixels. Read-only.
<code>UIObject.scaleX</code>	A number indicating the scaling factor in the x direction of the object, relative to its parent. Enabled at runtime after publishing SWF file containing the component.
<code>UIObject.scaleY</code>	A number indicating the scaling factor in the y direction of the object, relative to its parent. Enabled at runtime after publishing SWF file containing the component.

UIObject.top	The position of the top edge of the object, relative to its parent. Read-only.
UIObject.visible	A Boolean value indicating whether the object is visible (“true”) or not (“false”).
UIObject.width	The width of the object, in pixels. Read-only. Enabled at runtime after publishing SWF file containing the component.
UIObject.x	The left edge of the object, in pixels. Read-only.
UIObject.y	The top edge of the object, in pixels. Read-only.

Event summary for the ThreeDimensionalChanger class

The following table lists the event of the ThreeDimensionalChanger class.

Event	Description
ThreeDimensionalChanger.imageClick	Broadcast when a changer is clicked.
ThreeDimensionalChanger.imageData	Broadcast when an image is loaded.
ThreeDimensionalChanger.imageMove	Generated continuously while an image rotation.
ThreeDimensionalChanger.imageRoll	Broadcast when the pointer rolls over (off) a changer.
ThreeDimensionalChanger.onLoadXML	Broadcast when a changer attempts to load an XML file.

Events inherited from the UIObject class

The following table lists the events the ThreeDimensionalChanger class inherits from the UIObject class.

Event	Description
UIObject.hide	Broadcast when an object’s state changes from visible to invisible.
UIObject.reveal	Broadcast when an object’s state changes from invisible to visible.

ThreeDimensionalChanger.addImage()

Usage

```
myChanger.addImage(imageLinkage, imageType[, startIndex])
```

Parameters

imageLinkage A string that indicates either (a) the linkage identifier for an asset in the library (bitmap or symbol); or (b) the absolute or relative URL of the JPEG, GIF, PNG, or SWF file to be loaded. A relative path must be relative to the SWF file at level 0. Absolute URLs must include the protocol reference, such as `http://`.

imageType A string that specifies the type and location of the image you want to add to your collection. This parameter can be one of three values (the default value is “bitmap”):

- *bitmap* bitmap from library;
- *symbol* symbol from library;
- *file* external file.

index An integer that specifies the index of the image in the “Images Collection” where the insertion begins. You can specify a negative integer to specify a position relative to the end of the collection (for example, -1 is the last image of the “Images Collection”). This parameter is optional.

Returns

A Boolean value of “true” if the collection was changed as a result of the operation.

Description

Method; adds a new image to the end of the “Images Collection” or inserts it after the specified index in the array. The “Images Collection” must contain at least one image for this method to take effect at runtime.

Example

The following example adds a new external image (flower08.jpg) to the end of the “Images Collection”:

```
myChanger.addImage("flower08.jpg", "file");
```

ThreeDimensionalChanger.applyVisualEffect()

Usage

```
myChanger.applyVisualEffect(name, direction, quality, speed)
```

Parameters

name A string that specifies the type of a visual effect. The following values are acceptable for this parameter:

- *blur* softens the details of an image;
- *grayscale* converts the RGB image to grayscale image and vice versa;
- *brightness* changes the image brightness.

direction A string that specifies the direction of a visual effect. This parameter can be one of two values:

- *in* increasing the strength of the effect;
- *out* decreasing the strength of the effect.

quality A string that defines the depth or intensity of a visual effect. This parameter can be one of three values:

- *low*
- *medium*
- *high*

speed A string that defines the speed of increasing and decreasing of a visual effect. This parameter can be one of five values:

- *minimum*
- *low*
- *medium*
- *high*
- *maximum*

Returns

Nothing.

Description

Method; applies a specified visual effect to images in a 3DImageChanger component instance. The Visual Effect parameter must be set to “none” for this method to take effect.

Example

The following example applies a grayscale effect to the changer depending on mouse interaction:

```
myChanger.onRollOver = function() {
    myChanger.applyVisualEffect("grayscale", "in", "medium", "low");
}
myChanger.onRollOut = function() {
    myChanger.applyVisualEffect("grayscale", "out", "medium", "low");
}
```

ThreeDimensionalChanger.changeEffect

Usage

```
myChanger.changeEffect
```

Description

Property; a string that specifies the type of a 3D rotation effect that is used for changing an image. This property can be one of five predefined values: “prismatic quadrangular”, “prismatic triangular”, “cylindrical convex”, “cylindrical concave”, or “plane rectangular”. The default value is “prismatic quadrangular”.

Example

The following example sets the changeEffect property to “prismatic triangular”:

```
myChanger.changeEffect = "prismatic triangular";
```

ThreeDimensionalChanger.clear()

Usage

```
myChanger.clear()
```

Returns

Nothing.

Description

Method; removes all of the images from the “Images Collection”.

Example

The following example calls clear():

```
on (click) {
    myChanger.clear();
}
```

ThreeDimensionalChanger.currentImage

Usage

```
myChanger.currentImage
```

Description

Property (read-only); returns the current image in the “Images Collection”.

Example

The following example sets the value of image_num to the index of the current image:

```
var image_num: Number;
image_num = myChanger.getImageIndex(myChanger.currentImage);
```

ThreeDimensionalChanger.delay

Usage

```
myChanger.delay
```

Description

Property; a number that indicates (in seconds) how long each image “holds” before automatically displaying the next image in que. This property accepts any positive number or 0 (including fractional values) and is enabled at displayMode property set to “auto-forward” or “auto-backward”. If delay is equal to 0, the images are displayed continuously. The default value is 3.

Example

The following example sets the delay property to 5:

```
myChanger.delay = 5;
```

ThreeDimensionalChanger.direction

Usage

```
myChanger.direction
```

Description

Property; a string that specifies the direction of rotation while changing images. There are two values available (the default value is “CWW”):

CCW (Counterclockwise): top-down (for Rotation Plane set to “vertical”) or from left to right (for Rotation Plane set to “horizontal”).

CW (Clockwise): bottom-up (for Rotation Plane set to “vertical”) or from right to left (for Rotation Plane set to “horizontal”).

This is valid for Display Mode parameter set to “auto-forward” or “manual” when moving forward through the “Images Collection” (to every next image). If Display Mode parameter is set to “auto-backward”, the directions of rotation are opposite of listed above. Also, if moving backward through the “Images Collection” (to every previous image), the directions of rotation are opposite of listed above.

Example

The following example sets the direction property to “CW”:

```
myChanger.direction = "CW";
```

ThreeDimensionalChanger.displayMode

Usage

```
myChanger.displayMode
```

Description

Property; a string that specifies the playback mode of the “Images Collection”. By default, it is set to “manual,” which means 3DImageChanger will display the first image, stop and await user input. Changing this to “auto-forward” or “auto-backward” will instruct 3DImageChanger to automatically display the next image in que after the delay timer has expired.

Example

The following example sets the displayMode property to “auto-forward”:

```
myChanger.displayMode = "auto-forward";
```

ThreeDimensionalChanger.firstImage

Usage

```
myChanger.firstImage
```

Description

Property (read-only); returns the first image in the “Images Collection”.

Example

The following example compares the current image with the first image, and if they are equal – sets the value of `changer_stop` to “true”:

```
if (myChanger.currentImage == myChanger.firstImage) {  
    changer_stop = true;  
}
```

ThreeDimensionalChanger.getImageAt()

Usage

```
myChanger.getImageAt(index)
```

Parameters

index A number that indicates the location of image within the collection. This is a zero-based index, so 0 retrieves the first image, 1 retrieves the second image, and so on.

Returns

An image object.

Description

Method; returns an image within the “Images Collection” by using its index.

Example

The following example compares the image at index 5 with the current image, and if they are equal – toggles the display mode:

```
if (myChanger.getImageAt(5) == myChanger.currentImage) {  
    myChanger.toggleDisplayMode();  
}
```

ThreeDimensionalChanger.getImageIndex()

Usage

```
myChanger.getImageIndex(image)
```

Parameters

image An object that represents an image within the “Images Collection”.

Returns

The index of a specified image within the “Images Collection”.

Description

Method; finds an image within the “Images Collection” and returns its index.

Example

The following example checks whether the index of the current image is less than 10 and, if so, sets the value of `group_name` to “animals”:

```
if (myChanger.getImageIndex(myChanger.currentImage) < 10) {  
    group_name = "animals";  
}
```

ThreeDimensionalChanger.gotoFirstImage()

Usage

```
myChanger.gotoFirstImage()
```

Returns

Nothing.

Description

Method; navigates to the first image in the “Images Collection”.

Example

In the following example the changer navigates to the first image in the collection:

```
myChanger.gotoFirstImage();
```

ThreeDimensionalChanger.gotoImageAt()

Usage

```
myChanger.gotoImageAt(index)
```

Parameters

index A number that indicates the location of image within the collection. This is a zero-based index, so 0 retrieves the first image, 1 retrieves the second image, and so on.

Returns

Nothing.

Description

Method; navigates to the image in the “Images Collection” with the specified index.

Example

In the following example the changer navigates to the image with a random index between 0 and 100:

```
var num:Number = Math.round(Math.random()*100);  
myChanger.gotoImageAt(num);
```

ThreeDimensionalChanger.gotoLastImage()

Usage

```
myChanger.gotoLastImage()
```

Returns

Nothing.

Description

Method; navigates to the last image in the “Images Collection”.

Example

In the following example the changer navigates to the last image in the collection:

```
myChanger.gotoLastImage();
```

ThreeDimensionalChanger.gotoNextImage()

Usage

```
myChanger.gotoNextImage()
```

Returns

Nothing.

Description

Method; navigates to the next image in the “Images Collection”.

Example

In the following example the changer navigates to the next image in the collection:

```
myChanger.gotoNextImage();
```

ThreeDimensionalChanger.gotoPreviousImage()

Usage

```
myChanger.gotoPreviousImage()
```

Returns

Nothing.

Description

Method; navigates to the previous image in the “Images Collection”.

Example

In the following example the changer navigates to the previous image in the collection:

```
myChanger.gotoPreviousImage();
```

ThreeDimensionalChanger.granularity

Usage

```
myChanger.granularity
```

Description

Property; defines the degree of detail and the rendering quality while rotating an image. In numerical expression, it is the height or the width (depending on whether the Rotation Plane parameter is set to “vertical” or “horizontal”) of segments into which the image is split when rotating. The smaller the segments, the more the image is smoothed and anti-aliased, and vice versa. At the same time, reducing the value of granularity, you slow down the speed of rotation by increasing the time of rendering. This property can be an integer between 1 and 100 pixels. The default value is 10.

Example

The following example sets the granularity property to 8:

```
myChanger.granularity = 8;
```

ThreeDimensionalChanger.imageClick

Usage

Usage 1:

```
listenerObject = new Object();
```

```

listenerObject.imageClick = function(eventObject) {
    // Your code here
}
myChanger.addEventListener("imageClick", listenerObject);

```

Usage 2:

```

on (imageClick) {
    // Your code here
}

```

Description

Event; broadcast to all registered listeners when the mouse is clicked (pressed and released) over the changer.

The first usage example uses a dispatcher/listener event model. A component instance (*myChanger*) dispatches an event (in this case, `imageClick`) and the event is handled by a function, also called a “handler”, on a listener object (*listenerObject*) that you create. You define a method with the same name as the event on the listener object; the method is called when the event is triggered. When the event is triggered, it automatically passes an event object (*eventObject*) to the listener object method. The event object has properties that contain information about the event. You can use these properties to write code that handles the event. Finally, you call the `EventDispatcher.addEventListener()` method on the component instance that broadcasts the event to register the listener with the instance. When the instance dispatches the event, the listener is called.

For more information, see “EventDispatcher class” in *Macromedia Flash 8 Components Language Reference Help*.

The second usage example uses an `on()` handler and must be attached directly to a changer instance. The keyword `this`, used inside an `on()` handler attached to a component, refers to the component instance. For example, the following code, attached to a `3DImageChanger` instance *myChanger*, sends “_level0.myChanger” to the Output panel:

```

on (imageClick) {
    trace(this);
}

```

Example

This example, written on a frame of the Timeline, navigates to the next image in the “Images Collection” when the instance *myChanger* is clicked. The target property of an event object is the component that generated the event. You can access instance properties and methods from the target property (in this example, the `gotoNextImage()` method is accessed).

```

changerListener = new Object();
changerListener.imageClick = function(eventObj) {
    eventObj.target.gotoNextImage();
}
myChanger.addEventListener("imageClick", changerListener);

```

The following code sends a message to the Output panel when a changer is clicked. The `on()` handler must be attached directly to a changer instance.

```

on (imageClick) {
    trace("changer was clicked");
}

```

ThreeDimensionalChanger.imageData

Usage

```
listenerObject = new Object();
listenerObject.imageData = function(eventObject) {
    // Your code here
}
myChanger.addEventListener("imageData", listenerObject);
```

Description

Event; broadcast to all registered listeners when an image is loaded.

The usage example uses a dispatcher/listener event model. A component instance (*myChanger*) dispatches an event (in this case, *imageData*) and the event is handled by a function, also called a “handler”, on a listener object (*listenerObject*) that you create. You define a method with the same name as the event on the listener object; the method is called when the event is triggered. When the event is triggered, it automatically passes an event object (*eventObject*) to the listener object method. The event object has properties that contain information about the event. You can use these properties to write code that handles the event. Finally, you call the `EventDispatcher.addEventListener()` method on the component instance that broadcasts the event to register the listener with the instance. When the instance dispatches the event, the listener is called. For more information, see “EventDispatcher class” in *Macromedia Flash 8 Components Language Reference Help*.

Event object

Contains the following three properties:

linkage A string that indicates either the linkage identifier for an asset in the library (if the image is a bitmap or a symbol); or the absolute or relative URL (if the image is a loaded JPEG, GIF, PNG, or SWF file).

type A string that indicates the type of an image. There are three possible values of this property: “bitmap from library”, “symbol from library”, “external file”.

number A number (index) that indicates the location of an image within the collection.

Example

This example, written on a frame of the Timeline, sends a message to the Output panel when the image at index 10 is loaded.

```
changerListener = new Object();
changerListener.imageData = function(eventObj) {
    if (eventObj.data.number == 10) trace("image #10");
}
myChanger.addEventListener("imageData", changerListener);
```

ThreeDimensionalChanger.imageMove

Usage

```
listenerObject = new Object();
listenerObject.imageMove = function(eventObject) {
    // Your code here
}
myChanger.addEventListener("imageMove", listenerObject);
```

Description

Event; is generated continuously while an image rotation.

The usage example uses a dispatcher/listener event model. A component instance (*myChanger*) dispatches an event (in this case, *imageMove*) and the event is handled by a function, also called a “handler”, on a listener object (*listenerObject*) that you create. You define a method with the same name as the event on the listener object; the method is called when the event is triggered. When the event is triggered, it automatically passes an event object (*eventObject*) to the listener object method. The event object has properties that contain information about the event. You can use these properties to write code that handles the event. Finally, you call the `EventDispatcher.addEventListener()` method on the component instance that broadcasts the event to register the listener with the instance. When the instance dispatches the event, the listener is called. For more information, see “EventDispatcher class” in *Macromedia Flash 8 Components Language Reference Help*.

Event object

Contains one property that can retrieve the following values:

- *start* (String) Triggers at the beginning of an image rotation.
- (*number*) (Integer) Triggers continuously while an image rotation. Indicates the angle of an image turning (in degrees).
- *stop* (String) Triggers at the end of an image rotation.

Example

This example, written on a frame of the Timeline, sets the message variable to a corresponding value at the beginning and at the end of an image rotation.

```
changerListener = new Object();
changerListener.imageMove = function(eventObj) {
    if (eventObj.data == "start") message = "The image is loaded";
    if (eventObj.data == "stop") message = "New image loading...";
}
myChanger.addEventListener("imageMove", changerListener);
```

ThreeDimensionalChanger.imageRoll

Usage

```
listenerObject = new Object();
listenerObject.imageRoll = function(eventObject) {
    // Your code here
}
myChanger.addEventListener("imageRoll", listenerObject);
```

Description

Event; broadcast when the pointer rolls over a changer, or the pointer rolls off a changer.

The usage example uses a dispatcher/listener event model. A component instance (*myChanger*) dispatches an event (in this case, *imageRoll*) and the event is handled by a function, also called a “handler”, on a listener object (*listenerObject*) that you create. You define a method with the same name as the event on the listener object; the method is called when the event is triggered. When the event is triggered, it automatically passes an event object (*eventObject*) to the listener object method. The event object has properties that contain information about the event. You can use these properties to write code that handles the event. Finally, you call the `EventDispatcher.addEventListener()` method on the component instance that broadcasts the event to register the listener with the instance. When the instance dispatches the event, the listener is called. For more information, see “EventDispatcher class” in *Macromedia Flash 8 Components Language Reference Help*.

Event object

Contains one property that can retrieve the following values:

- *over* (String) Triggers when the pointer rolls over a changer.
- *out* (String) Triggers when the pointer rolls off a changer.

Example

This example, written on a frame of the Timeline, applies a visual effect (“brightness”) in direction of increasing its strength – when the pointer rolls over the instance `myChanger`, and in direction of decreasing its strength – when the pointer rolls off the instance `myChanger`. The target property of an event object is the component that generated the event. You can access instance properties and methods from the target property (in this example, the `applyVisualEffect()` method is accessed).

```
changerListener = new Object();
changerListener.imageRoll = function(eventObj) {
    if (eventObj.data == "over") eventObj.target.applyVisualEffect("brightness", "in", "low", "medium");
    if (eventObj.data == "out") eventObj.target.applyVisualEffect("brightness", "out", "low", "medium");
}
myChanger.addEventListener("imageRoll", changerListener);
```

ThreeDimensionalChanger.keyboardControl

Usage

```
myChanger.keyboardControl
```

Description

Property; determines whether `3DImageChanger` uses default keyboard handling to navigate through the images in collection when the component instance has focus. This property can be either "true" or "false". The default value is "false".

Example

The following example sets the `keyboardControl` property to "true":

```
myChanger.keyboardControl = true;
```

ThreeDimensionalChanger.lastImage

Usage

```
myChanger.lastImage
```

Description

Property (read-only); returns the last image in the "Images Collection".

Example

The following example compares the current image with the last image, and if they are equal – sets the value of `changer_stop` to "true":

```
if (myChanger.currentImage == myChanger.lastImage) {
    changer_stop = true;
}
```

ThreeDimensionalChanger.length

Usage

```
myChanger.length
```

Description

Property (read-only); the number of images in the “Images Collection”.

Example

The following example displays the number of images currently in the “Images Collection”:

```
trace("The total number of images: " + myChanger.length);
```

ThreeDimensionalChanger.lockMode

Usage

```
myChanger.lockMode
```

Description

Property; determines whether 3DImageChanger is enabled or disabled to “store” user interactions. That means 3DImageChanger can react differently to navigation commands (such as “gotoNextImage”, etc.) while changing (rotating) images. This property can be either “true” or “false” (the default value is “false”):

true 3DImageChanger ignores any navigation command.

false 3DImageChanger stores up to 3 navigation commands running every next of them after the previous one has been completed. If the commands are opposite to each other in direction (for example, “gotoNextImage” – “gotoPreviousImage”), 3DImageChanger reverses rotation immediately not waiting for the end of the movement.

Example

The following example sets the lockMode property to “true”:

```
myChanger.lockMode = true;
```

ThreeDimensionalChanger.nextImage

Usage

```
myChanger.nextImage
```

Description

Property (read-only); returns the next image in the “Images Collection”.

Example

The following example compares the next image with the image at index 10, and if they are equal – sets the displayMode property to “manual”:

```
if (myChanger.nextImage == myChanger.getImageAt(10)) {  
    myChanger.displayMode = "manual";  
}
```

ThreeDimensionalChanger.onLoadXML

Usage

```
listenerObject = new Object();  
listenerObject.onLoadXML = function(eventObject) {  
    // Your code here  
}  
myChanger.addEventListener("onLoadXML", listenerObject);
```

Description

Event; broadcast when a changer attempts to load an XML file.

The usage example uses a dispatcher/listener event model. A component instance (*myChanger*) dispatches an event (in this case, `onLoadXML`) and the event is handled by a function, also called a “handler”, on a listener object (*listenerObject*) that you create. You define a method with the same name as the event on the listener object; the method is called when the event is triggered. When the event is triggered, it automatically passes an event object (*eventObject*) to the listener object method. The event object has properties that contain information about the event. You can use these properties to write code that handles the event. Finally, you call the `EventDispatcher.addEventListener()` method on the component instance that broadcasts the event to register the listener with the instance. When the instance dispatches the event, the listener is called. For more information, see “EventDispatcher class” in *Macromedia Flash 8 Components Language Reference Help*.

Event object

Contains a Boolean value (“false” or “true”) to indicate whether an XML file was found.

Example

This example, written on a frame of the Timeline, uses the event object to set up a conditional statement that checks to see if an XML file was found and loaded:

```
changerListener = new Object();
changerListener.onLoadXML = function(eventObj) {
    if (eventObj.data == true) {
        // show success alert
    } else {
        // show error alert
    }
}
myChanger.addEventListener("onLoadXML", changerListener);
```

ThreeDimensionalChanger.preloaderAlpha

Usage

```
myChanger.preloaderAlpha
```

Description

Property; a number that indicates the transparency percentage for the preloader animation. This property accepts any number at or between 0 and 100. The default value is 100.

Example

The following example sets the `preloaderAlpha` property to 70:

```
myChanger.preloaderAlpha = 70;
```

ThreeDimensionalChanger.preloaderColor

Usage

```
myChanger.preloaderColor
```

Description

Property; the color of the preloader animation. The property accepts hex values. The default value is “FFFFFF”.

Example

The following example sets the `preloaderColor` property to “FF6600” (orange):

```
myChanger.preloaderColor = "FF6600";
```

ThreeDimensionalChanger.preloaderLocation

Usage

```
myChanger.preloaderLocation
```

Description

Property; a string that specifies the location of the preloader in relation to the static image borders. This property can be one of five predefined values: “center”, “top-left”, “top-right”, “bottom-left”, or “bottom-right”; the default value is “center”.

Example

The following example sets the `preloaderLocation` property to “top-left”:

```
myChanger.preloaderLocation = "top-left";
```

ThreeDimensionalChanger.preloaderMode

Usage

```
myChanger.preloaderMode
```

Description

Property; determines whether the preloader is visible or hidden when loading images. This property can be one of two predefined values: “always visible” or “always hidden”. The default value is “always visible”.

Example

The following example sets the `preloaderMode` property to “always hidden”:

```
myChanger.preloaderMode = "always hidden";
```

ThreeDimensionalChanger.preloaderType

Usage

```
myChanger.preloaderType
```

Description

Property; a string that specifies the preloader appearance. This property can be one of four predefined values: “spiral”, “progress bar”, “progress circle”, “text string”. The default value is “progress circle”.

Example

The following example sets the `preloaderType` property to “progress bar”:

```
myChanger.preloaderType = "progress bar";
```

ThreeDimensionalChanger.previousImage

Usage

```
myChanger.previousImage
```

Description

Property (read-only); returns the previous image in the “Images Collection”.

Example

The following example compares the previous image with the image at index 10, and if they are equal – sets the `displayMode` property to “manual”:

```
if (myChanger.previousImage == myChanger.getImageAt(10)) {  
    myChanger.displayMode = "manual";  
}
```

ThreeDimensionalChanger.removeImageAt()

Usage

```
myChanger.removeImageAt(index)
```

Parameters

index A number that indicates the location of image within the collection. This is a zero-based index, so 0 retrieves the first image, 1 retrieves the second image, and so on. You can specify a negative integer to specify a position relative to the end of the collection (for example, -1 is the last image of the “Images Collection”).

Returns

A Boolean value of “true” if an image was removed successfully; returns “false” if the image is not found.

Description

Method; removes a single image from the “Images Collection” by using its index.

Example

The following example removes the image at index 5 from the collection:

```
myChanger.removeImageAt(5);
```

ThreeDimensionalChanger.repeatMode

Usage

```
myChanger.repeatMode
```

Description

Property; instructs 3DImageChanger how to behave when reaching the final image in the “Images Collection”. There are two values available:

false Auto-playback stops and Display Mode changes to “manual” (if previously set to “auto-forward” or “auto-backward”).

true The collection restarts with either the first image (Display Mode set to “auto-forward” or “manual”), or the last image (Display Mode set to “auto-backward”).

The default value is “true”.

Example

The following example sets the `repeatMode` property to “false”:

```
myChanger.repeatMode = "false";
```

ThreeDimensionalChanger.reset()

Usage

```
myChanger.reset()
```

Returns

Nothing.

Description

Method; resets a 3DImageChanger instance to its initial state (to the time of the first image loading) and clears it from the visual effect created by applyVisualEffect() method. It's desirable to use the reset() method, for example, if you apply a visual effect to a changer multiple times at runtime, to avoid unexpected behavior.

Example

The following example calls reset():

```
on (click) {  
    myChanger.reset();  
}
```

ThreeDimensionalChanger.reverse()

Usage

```
myChanger.reverse()
```

Returns

Nothing.

Description

Method; reverses the direction of rotation immediately while rotating images. This method has no effect on static image (when no rotation occurs). There are two possible cases of a changer behavior when reverse() is invoked, depending on a component's settings:

- If Display Mode parameter is not "manual", and delay is set to 0 – the direction of rotation reverses and the value of displayMode property changes to the opposite direction (for example, from "auto-forward" to "auto-backward").
- In other cases – the direction of rotation reverses but for the current turn only; the value of displayMode property remains the same.

Example

The following example calls reverse():

```
on (click) {  
    myChanger.reverse();  
}
```

ThreeDimensionalChanger.rotation

Usage

```
myChanger.rotation
```

Description

Property; a number that indicates the degree of rotation of images while their 3D rotation. The smaller the angle, the more the rotation is smooth, and vice versa. At the same time, reducing the value of rotation, you slow down the speed of rotation by increasing the time of rendering.

You can select any integer between 1 and 180 degrees; but the real value of rotation angle will be the one from the [table](#) above that is the closest to the selected number for correspondent Change Effect.

This property overrides the Rotation Angle parameter value set during authoring. The default value for rotation is 10.

Example

The following example sets the rotation property to 15:

```
myChanger.rotation = 15;
```

ThreeDimensionalChanger.rotationPlane

Usage

```
myChanger.rotationPlane
```

Description

Property; a string that defines the plane of rotation while changing images. This property can be either “horizontal” or “vertical”. The default value is “vertical”.

Example

The following example sets the rotationPlane property to “horizontal”:

```
myChanger.rotationPlane = "horizontal";
```

ThreeDimensionalChanger.scaleContent

Usage

```
myChanger.scaleContent
```

Description

Property; determines whether the content scales to fit the changer, or the changer scales to fit the content. This property can be one of two predefined values: “fit changer” or “fit image”. The default value is “fit image”. For more information, see the [Scale Content](#) parameter.

Example

The following example sets the scaleContent property to “fit changer”:

```
myChanger.scaleContent = "fit changer";
```

ThreeDimensionalChanger.toggleDisplayMode()

Usage

```
myChanger.toggleDisplayMode()
```

Returns

Nothing.

Description

Method; toggles the displayMode property from "manual" to "auto-forward" / "auto-backward" and vice versa.

Example

The following example calls toggleDisplayMode():

```
on (click) {  
    myChanger.toggleDisplayMode();  
}
```

ThreeDimensionalChanger.togglePlayMode()

Usage

```
myChanger.togglePlayMode()
```

Returns

Nothing.

Description

Method; toggles the play mode of a 3DImageChanger instance from “play” to “stop” and vice versa. Calling this method causes the images in changer to stop or resume the rotation from a stopped position instantly.

Example

The following example calls togglePlayMode():

```
on (click) {  
    myChanger.togglePlayMode();  
}
```

ThreeDimensionalChanger.visualEffect

Usage

```
myChanger.visualEffect
```

Description

Property; a string that specifies the type of a visual effect to be applied to a 3DImageChanger instance. This property can be one of four predefined values: “none”, “blur”, “grayscale”, “brightness”. The default value is “none” (no visual effect applied to a 3DImageChanger instance). For more information, see the [Visual Effect](#) parameter.

Example

The following example sets the visualEffect property to “blur”:

```
myChanger.visualEffect = "blur";
```

ThreeDimensionalChanger.visualEffectQuality

Usage

```
myChanger.visualEffectQuality
```

Description

Property; determines the depth or intensity of the visual effect. This property can be one of three predefined values: “low”, “medium”, “high”. The default value is “medium”.

Example

The following example sets the visualEffectQuality property to “low”:

```
myChanger.visualEffectQuality = "low";
```

ThreeDimensionalChanger.visualEffectSpeed

Usage

```
myChanger.visualEffectSpeed
```

Description

Property; determines the speed of increasing and decreasing of the visual effect. This property can be one of five predefined values: “minimum”, “low”, “medium”, “high”, “maximum”. The default value is “medium”.

Example

The following example sets the `visualEffectSpeed` property to “high”:

```
myChanger.visualEffectSpeed = "high";
```

ThreeDimensionalChanger.xml

Usage

myChanger.xml

Description

Property (read-only); returns the XML object that represents the content of the downloaded configuration XML file.

Example

This is the configuration XML file for a `3DImageChanger` instance named “myChanger”. The file contains the list of graphic files to be loaded by the component, and additional attributes for each element. The code below shows how to access the attribute values looping through the XML node's child nodes.

```
<?xml version="1.0" encoding="UTF-8"?>
<zodiac path="pictures/zodiac/">
  
  
  
  
  
  
  
  
  
  
  
  
</zodiac>

for (var aNode:XMLNode = myChanger.xml.firstChild.firstChild; aNode != null; aNode = aNode.nextSibling) {
  trace("Name: " + aNode.attributes.name);
  trace("Ruler: " + aNode.attributes.ruler);
  trace("Element: " + aNode.attributes.element);
  trace("Pole: " + aNode.attributes.pole);
}
```

ThreeDimensionalChanger.xmlEnable

Usage

myChanger.xmlEnable

Description

Property; defines the method that is used for adding graphic objects to internal “Images Collection”. There are two options available:

false The user adds and edits the items in the Values dialog box (opened from Images Collection text box within the Parameters tab for 3DImageChanger component).

true The user specifies the name of configuration XML file that contains the list of graphic files to be loaded into 3DImageChanger.

The default value is “false”.

Example

The following example sets the `xmlEnable` property to “false”:

```
myChanger.xmlEnable = "false";
```

ThreeDimensionalChanger.xmlFilePath

Usage

```
myChanger.xmlFilePath
```

Description

Property; the name of configuration XML file. The default value is “images.xml” (in this case, the XML file must be saved in the same directory as the SWF file that contains a 3DImageChanger component).

Example

The following example sets the `xmlFilePath` property to “flowers/tulips.xml”:

```
myChanger.xmlFilePath = "flowers/tulips.xml";
```