By using this file it is possible to create a 360° product viewer while no Actionscript knowledge is required! By inserting a number of images with views from different angles of the product the file automatically adjusts to create an interactive viewer.

How to use

General remark

The main code can be found in the movieclip named "_Controller".

Changing the object itself

You can insert an infinite amount of images with different angles from an object, the file will automatically adjust to it. In order to do so, replace the images in the movieclip called "_Object Clip". Leave no blank keyframes.

Labeling angles

By labeling the frames containing the front, left, back and right view of the object in the movieclip "Object Clip", it is possible make the object rotate directly to those sides by using the following functions:

```
object.showFront();
object.showLeft();
object.showBack();
object.showRight();
```

Changing properties

It is possible to enable and disable the following properties (by default, all properties are set to 'true'):

Intertia (being able to swing the object around)
object.inertia=true; or object.inertia=false;

Motion blur (the blur when rotating the object)
object.blur=true; or object.blur=false;

Reflection (reflection of the object on the floor)
object.reflection=true; or object.reflection=false;

Relection blur (whether the relection is blurry or not)
object.relectionblur=true; or object.reflectionblur=false;

Show start (whether the object rotates in when initiated or not)
object.showstart=true; or object.showstart=false;

Grabhand cursor (whether or not a custom grabhand cursor is shown when grabbing the object)
object.grabhand=true; or object.grabhand=false;

Draggable (whether the object is draggable or not (you may only want to use playback))
object.draggable=true; or object. draggable=false;

Adjusting speed and delay

It is also possible to adjust some speed and delay:

Speed (the speed with which the object rotates, an integer between 0 and 10)
object.speed=5;

Delay (the amount of inertia the object has, an integer greated than 1)
object.delay=10;

Mousewheel functionality

The functionality of the mousewheel can be as follows:

Zooming in and out (in this case you will have to set the mousewheel property to "zoom")
object.mousewheel="zoom";

No functionality at all (in this case you will have to set the mousewheel property to false)
object.mousewheel=false;

Rotation of the object (default setting of the mousewheel, no need to set the property)

Zoom options

It is required to set a "zoom range" in order to be able to zoom in or out on the object. In this zoom range you will give a minimum and maximum scale the object can zoom in between:

object.setZoomrange(minimal scale, maximal scale);

In the preview file this is done as follows:

object.setZoomrange(100,140);

This therefore means that the minimum zoom is the size of the object itself and the maximum zoom is scale 140. The minimum zoom can also be a value lower than 100 in which the object becomes smaller than the default object.

After setting the zoom range it is possible to set a zoom percentage, this is done by using the setZoom method, requiring a percentage between 0% and 100% zoom. 0% means that the zoom level will be the minimal scale as indicated in the zoom range, 100% will be the maximal zoom as set in the zoom range.

In conclusion, the following example will end up in the object being 1.2 times biggen than default:

```
object.setZoomrange(100,140);
object.setZoom(50);
```

Playback functionality

There are several options for playback functionality:

```
Going one step back of forth in rotation
object.stepRight();
object.stepLeft();
```

But it is also possible to use the setPlayback method, requiring a percentage between 0% and 100% where 0% represents the first keyframe of the rotation movie and 100% represents the last keyframe of the rotation movie. Therefore, going from 0% to 100% gives you one 360° turn of the object:

```
object.setPlayback (n);   // with n being between 0% and 100%
```

Adding hotspots

Adding hotspots is very easy. Just open the movieclip "_Object Clip" with the object's movie in it. There is a layer above it called 'hotspots'. There you can add buttons on the positions that you want hotspots. By calling back to the root where the object is located you can show different content.

A single hotspot is shown in the preview file. Of course it is possible to add more hotspots.

How the loading in the preview works

Preview.fla in de non-default version of this file only contains a loading. The actual object is nested in *360viewer.swf*. 360viewer.swf is also the 'heavy file' (the file with all the pictures nested in the library) and therefore needs to be preloaded. This is done in preview.fla, it generates a preview.swf which loads the 360viewer.swf externally. Therefore, the user immediately sees the loader (since it is in preview.swf which is only a few KBs) and the heavy stuff will be preloaded visually.

In preview.fla you see the background of the object. In keyframe 1 the preview.swf itself is loaded with an easy load script:

```
var t=this.getBytesTotal();
this.loadContent=function(){
        var g=this.getBytesLoaded();
        if(g==t){
                this.onEnterFrame=null;
                this.gotoAndStop(2);
        }
}
this.onEnterFrame=this.loadContent;
this.stop();
```

The loadContent function will be looped until the getBytesLoaded will match the getBytesTotal and then the movie will go to keyframe 2.

In keyframe 2 the 360viewer.swf will be loaded:

```
this.attachObject=function(){
	var a=this.createEmptyMovieClip("container",10);
	a.loadMovie("360viewer.swf");
	this.onEnterFrame=this.loadObject;
}
this.loadObject=function(){
	var t=this.container.getBytesTotal();
	if(t>3){
		var g=this.container.getBytesLoaded();
		this.perc.text=Math.round((g/t)*100)+"%";
		if(g==t){
			this.perc.text="";
			this.onEnterFrame=null;
		}
	}
}
this.attachObject();
```

First the attachObject function is executed. In this function an empty movieclip is created in which the 360viewer.swf is going to be loaded with the loadMovie method. Then we start a looping event checking whether loading status (this.onEnterFrame=this.loadObject;).

In the loadObject function the status of loading the object is checked. First *t* is set to the total amount of bytes of the 360viewer.swf file. When this amount is higher than 3 (which means that the swf is found), *g* will represent the amount actually loaded. Then we will set the loaded percentage in the dynamic text field on stage which is called "perc" by saying:

```
this.perc.text=Math.round((g/t)*100)+"%";
```

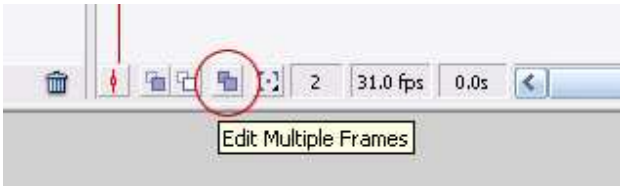
Which basically just divides the loaded amount by the total amount to calculate the percentage loaded. When the amount loaded equals the total amount to be loaded we know that the 360viewer.swf is loaded and "perc" can be se to "". Also, the onEnterFrame event has to be set to null; which stops looping the loadObject function.

Setting the imported images to the right location in an easy way

To import several images make sure your images are numbered such as image1.png, image2.png, image3.png and so on. Then select import to stage and choose the first frame in your sequence. Flash should pick up the sequence and ask if you would like to import all the images. This should import all the images in your sequence and place each image in the timeline. One image per frame.

Now to relocate the images to the correct x and y coordinates:

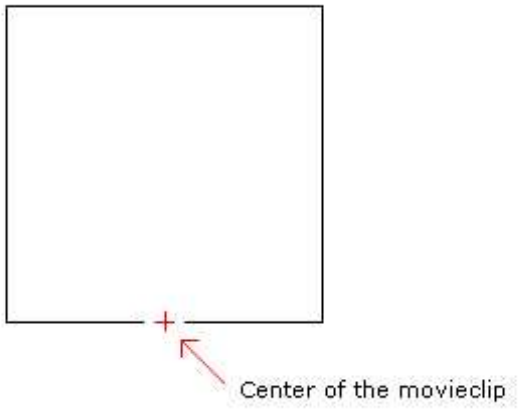
In the timeline you will notice 5 buttons down at the bottom of the timeline window next to the fps. Choose the 4th button "edit multiple images" the 2 shaded box icon.:



This will pull up a slider bracket in the timeline. Adjust the slider to highlight all your images then select all your keyframes. Once you have all the frames selected move the images to the correct location. Uncheck the edit multiple image icon and check your images. All the images should now be in the same location! *(Thanks to daveh113 for posting this approach on FlashDen!).*

Location of the images

In order for the object to work properly (especially with the reflection) the position of the images should be horizontally aligned with the center of the movieclip and vertically aligned with the bottom of the images on the center of the movieclip. This image illustrates this position:



About the reflection

Sometimes it may appear that the reflection is rotating in the opposite direction of the object itself. This is why: the reflection is not a real reflection of the object itself but a reflection of the image. Therefore, the reflection modus of this object will only work when the pictures are taken from a straight angle. The reflection will seem to rotate in the opposite direction when the object in the image is in perspective. These illustration shows what happens:

In the left image you see that the picture is taken from a straight angle, therefore the direct reflection of the image (just flipping it vertically) is correct. However, when the object is in perspective like in the picture on the right the reflection does not match the object itself.

There is nothing to do about this except for:

- Taking the pictures in a straight angle
- Make sure that the reflection is already in the image itself

Thank you for your purchase and good luck using the file!
Check out the rest of Webmarbles' portfolio on Webmarbles.com or FlashDen!